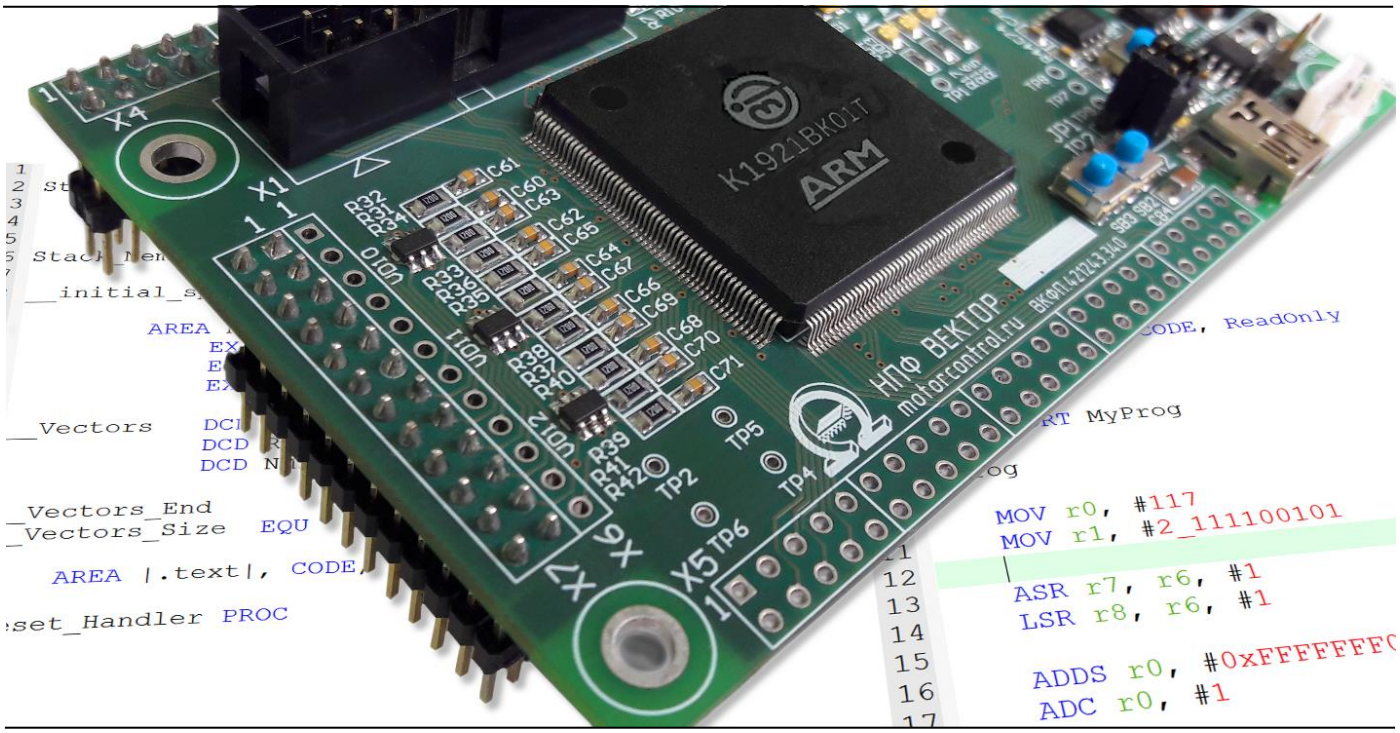
Козаченко В.Ф., Алямкин Д.И., Анучин А.С.,
Жарков А.А., Лашкевич М.М.,
Савкин Д.И., Шпак Д.М.



Практический курс

микропроцессорной техники на базе процессорных ядер ARM-Cortex-M3/M4/M4F

ПРИЛОЖЕНИЕ 1. ИЛЛЮСТРИРОВАННЫЙ СПРАВОЧНИК ПО СИСТЕМЕ КОМАНД ПРОЦЕССОРНЫХ ЯДЕР CORTEX-M3/M4/M4F

СОДЕРЖАНИЕ

1. Команды пересылки данных.....2
2. Арифметические команды.....26
3. Логические команды.....47
4. Команды умножения, умножения с накоплением, деления.....58
5. Команды насыщения.....91
6. Команды упаковки и распаковки данных.....109
7. Команды работы с битовыми полями.....119
8. Команды сопроцессора (обработки чисел в формате с плавающей точкой).....122
9. Команды передачи управления.....162
10. Команды специального назначения.....170

1 КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

Оглавление

1.1. Межрегистровая пересылка. Пересылка с побитовым инвертированием. 3

1.2. Загрузка констант в старшее полуслово 5

1.3. Реверсирование порядка расположения байт и бит в слове....6

1.4. Загрузка регистра-указателя адресом расположения данных (адресом метки) 8

1.5. Загрузка регистров ЦПУ из памяти и сохранение в памяти по содержимому базового регистра и непосредственному смещению 9

1.6. Загрузка регистров ЦПУ из памяти и сохранение в памяти по содержимому базового регистра и смещению в индексном регистре 13

1.7. Загрузка регистров из памяти и сохранение в памяти по содержимому базового регистра и непосредственному смещению с непривилегированным доступом 15

1.8. Загрузка регистра/двух регистров из памяти с использованием относительной адресации по содержимому счетчика команд PC....17

1.9. Команды множественной загрузки регистров из памяти и сохранения в памяти 19

1.10. Сохранение содержимого регистров в стеке и восстановление из стека....21

1.11. Эксклюзивная загрузка и сохранение регистра....23

1.12. Отмена режима эксклюзивности 25

1.1 Межрегистровая пересылка. Пересылка с побитовым инвертированием

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MOV и **MVN**	Move and Move NOT - Пересылка и пересылка с побитовом инвертированием	v	v	v

Синтаксис

```
MOV{S}{cond} Rd, Operand2
MOV{cond} Rd, #imm16
MVN{S}{cond} Rd, Operand2
```

Где:	**MOV** Пересылка данных **MVN** Пересылка побитово-проинвертированных данных	**cond** код условного выполнения команды (опция)
	S Опциональный суффикс (Set Flags). Если он специфицирован, то флаги кода условного выполнения обновляются по результату операции.	
	Rd – Регистр назначения.	
	Operand2 – Гибкий второй операнд: константа, регистр ЦПУ, регистр ЦПУ с опцией предварительного сдвига.	
	imm16 – 16-разрядный непосредственный операнд в диапазоне 0-65535	

Операция

Команда пересылки **MOV** копирует данные из операнда 2 Operand2 в регистр назначения Rd. Если операнд 2 в команде пересылки является регистром с логическим сдвигом влево на число разрядов, отличное от 0 (LSL #0), то предпочтительным синтаксисом является соответствующая ей команда сдвига:

Команда пересылки	Предпочтительный синтаксис с использованием команды сдвига
MOV{S}{cond} Rd, Rm, ASR #n	ASR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, LSL #n (if n != 0)	LSL{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, LSR #n	LSR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, ROR #n	ROR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, RRX	RRX{S}{cond} Rd, Rm

Таким образом, команда пересылки с опцией сдвига второго операнда выполняет ту же самую функцию, что и соответствующая команда сдвига. В этом случае рекомендуется вместо команды пересылки использовать соответствующую команду сдвига ASR, LSL, LSR, ROR или RRX.
Если величина сдвига второго операнда задается содержимым регистра Rs, то вместо соответствующей команды сдвига можно использовать команду пересылки в следующем формате:

Команда сдвига	Синоним – Команда пересылки
ASR{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, ASR Rs
LSL{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, LSL Rs
ROR{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, ROR Rs

Команда **MVN** (пересылка с предварительной побитовой инверсией) вначале выполняет побитовое логическое НЕ содержимого второго операнда, а затем помещает результат в регистр назначения Rd.

Замечание:
Команда **MOVW** выполняет те же самые функции, что и команда MOV, однако ее использование ограничено: допустим только 16-разрядный непосредственный операнд imm16. При этом непосредственные данные загружаются в младшее полуслово регистра назначения и автоматически расширяются нулями в область старшего слова.

Ограничения

Вы можете использовать указатель стека SP и счетчик команд PC в командах пересылки данных при следующих ограничениях:

- Второй операнд должен быть регистром без сдвига
- Вы не должны специфицировать в команде суффикс установки флагов S.

Если счетчик команд PC является регистром-приемником в команде MOV, то:

- Bit[0] в значении, которое записывается в PC, игнорируется
- Переход происходит по адресу, в котором нулевой бит принудительно сброшен в 0: bit[0]=0.

Замечание:
Хотя Вы можете использовать команду MOV в качестве команды передачи управления (branch instruction) по косвенному адресу, все же настоятельно рекомендуется, по возможности, использовать команды перехода BX или BLX для обеспечения гарантии переместимости программного обеспечения для ARM-процессоров с системой команд Cortex-M3/M4F.

Выработка флагов

Если опция установки флагов S специфицирована:

- Флаги N и Z обновляются в соответствии с результатом операции.
- Флаг C может обновиться в соответствии с выполненной операцией сдвига второго операнда.
- Команда не влияет на состояние флага переполнения V.

Таким образом, команда MOV может модифицировать флаги, если используется с опцией установки флагов S.

Примеры

```
MOVS R11, #0x000B     ; Записать число 0x000B в регистр R11,
                      ; Установить флаги.
MOV R1, #0xFA05       ; Записать число 0xFA05 в регистр R1,
                      ; Флаги не обновлять.
MOVS R10, R12         ; Переслать данные из регистра R12 в R10,
                      ; Установить флаги.
MOV R3, #23           ; Записать константу 23 в регистр R3.
MOV R8, SP            ; Записать содержимое указателя стека в R8.
MVNS R2, #0xF         ; Записать побитово-инверсное значение
                      ; непосредственного операнда 0xF, а именно
                      ; число 0xFFFFFFF0 в регистр R2
                      ; Установить флаги
```

1.2 Загрузка констант в старшее полуслово

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MOVT	Move Top – Пересылка вверх (Загрузка старшего полуслова)		V	V

Синтаксис

```
MOVT{cond} Rd, #imm16
```

Где:	
	cond код условного выполнения команды (опция)
	Rd - Регистр-приемник данных
	imm16 - 16-разрядный непосредственный операнд (константа)

Операция

Команда **MOVT** записывает 16-разрядное непосредственное значение imm16 в старшее полуслово регистра назначения Rd[31:16], не изменяя младшее полу-слово Rd[15:0].

Внимание!

Использование последовательно двух команд пересылки MOV (или MOVW) и MOVT с непосредственными данными (16-разрядными константами imm16) позволяет загружать в регистры ЦПУ любые 32-разрядные константы.

Ограничения

Регистр назначения не может быть счетчиком команд PC или указателем стека SP.

Выработка флагов

Эта команда не вырабатывает флаги

Примеры

```
MOVT R3, #0xF123  ; Записать число 0xF123 в старшее полуслово
                  ; регистра R3, сохранив младшее полуслово
                  ; и состояние флагов неизменным
MOVW R4, #0x5555  ; Последовательно загрузить в регистр R4
MOVT R4, #0xAABB  ; 32-разрядную константу 0xAABB5555,
                  ; разбив ее на младшее и старшее полуслова
```

1.3 Реверсирование порядка расположения байт и бит в слове

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
REV , REV16, REVSH, and RBIT	Reverse bytes and Reverse bits – Реверсирование байт и реверсирование бит	v	v	v

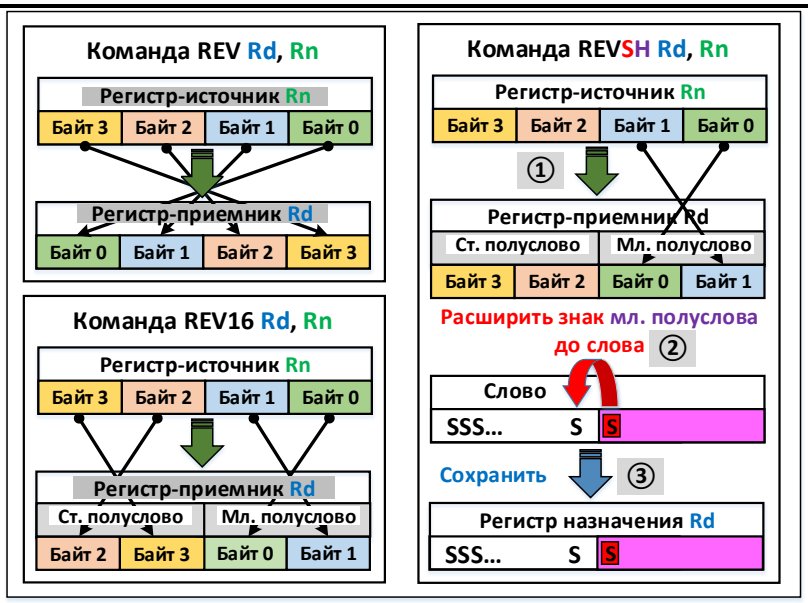
Синтаксис

```
op{cond} Rd, Rn
```

Где:	**op** оптокод один из: **REV** - Reverse byte order in a word – Реверсирование порядка байт в слове **REV16** - Reverse byte order in each halfword independently – Реверсирование порядка байт в каждом полуслове независимо **REVSH** - Reverse byte order in the bottom halfword, and sign extend to 32 bits – Реверсирование порядка байт в младшем полуслове и знаковое расширение до 32 бит. **RBIT** - Reverse the bit order in a 32-bit word – Реверсирование порядка бит в 32-разрядном слове	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий исходный операнд	

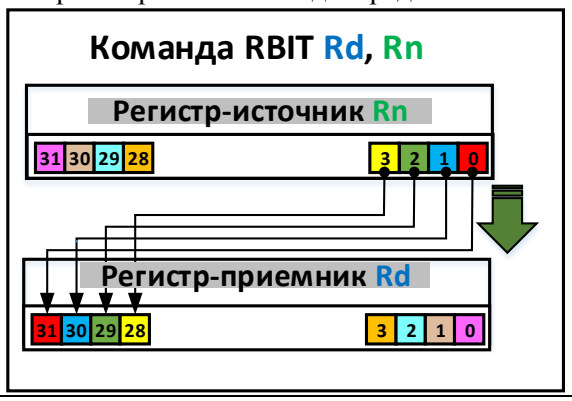
Операция

Эти команды изменяют порядок размещения данных в слове:
REV – Реверсирует порядок расположения байт в слове: преобразует 32- разрядное число с обратным порядком расположения байт (32-bit big-endian data) в 32-разрядное число с прямым порядком расположения байт (32-bit little-endian data) и обратно.
REV16 – Реверсирует порядок расположения байт в каждом полуслове независимо.
REVSH – Реверсирует порядок байт в младшем полуслове и расширяет результат, как число со знаком в дополнительном коде (полуслово) до 32-разрядного слова
Графическая иллюстрация алгоритма работы этих команд представлена ниже:



RBIT – Реверсирует порядок расположения бит в слове

Графическая иллюстрация алгоритма работы команды представлена ниже:



Ограничения

■ Нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Флаги не вырабатываются.

Примеры

```
REV R3, R7        ; Реверсирование порядка байт в R7 и
                  ; запись результата в R3
REV16 R0, R0      ; Реверсирование порядка байт в каждом полуслове
                  ; регистра R0 и запись туда же
REVSH R0, R5      ; Reverse Signed Halfword - Реверсирование байт в
                  ; младшем полуслове и знаковое расширение до
                  ; слова
REVHS R3, R7      ; Реверсирование байт  в слове по условию
                  ; Higher or Same (Выше или то же самое).
RBIT R7, R8       ; Реверсирование бит в регистре R8 и запись в R7
```

1.4 Загрузка регистра-указателя адресом расположения данных (адресом метки)

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
ADR	Generate PC-relative address. Загрузка регистра адресом метки, с использованием относительной адресации по счетчику команд PC	v	v	v

Синтаксис

`ADR{cond} Rd, label`

Где:

`cond`	Опциональный код условного выполнения команды
`Rd`	Регистр назначения
`label`	Выражение относительно текущего значения PC счетчика команд

Операция

Команда **ADR** определяет адрес по смещению (непосредственному значению) относительно текущего значения счетчика команд PC до места расположения данных label и записывает результат [(PC) +offset] в регистр назначения. Смещение автоматически рассчитывается на этапе трансляции программы и сохраняется в коде команды.
Команда ADR генерирует позиционно независимый код, так как адреса являются относительными – относительно текущего значения счетчика команд PC.
Если Вы используете команду ADR для генерации адреса для команд косвенной передачи управления BX и BLX, Вы должны быть уверены, что bit[0] адреса, который Вы генерируете, установлен в 1 для дальнейшего корректного выполнения программы. Это будет означать, что текущий режим работы с набором команд Thumb-2 подтверждается при переходе.
Значение метки должно быть в диапазоне от -4095 до +4095. (13-разрядное знаковое смещение относительно текущего значения в PC).

Замечание:
Вы можете использовать суффикс «.W» для генерации максимально большого диапазона смещений или генерировать адреса, который не являются выравненным по слову.

Ограничения

Rd не может быть указателем стека SP и не может быть счетчиком команд PC.

Выработка флагов

Состояние флагов не изменяется

Примеры

```
ADR R1, Text     ; Загрузить в регистр R1 значение адреса
                 ; области памяти, помеченное меткой Text
ADR R2, Var_1    ; Загрузить регистр-указатель R2 адресом
                 ; расположения в памяти переменной Var_1
```

1.5 Загрузка регистров ЦПУ из памяти и сохранение в памяти по содержимому базового регистра и непосредственному смещению

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDR and **STR** (Immediate Offset)	Загрузка и сохранение регистра ЦПУ с использованием базового регистра (указателя) и непосредственного смещения	v	v	v

Синтаксис:

```
op{type}{cond} Rt, [Rn {, #offset}]    ; immediate offset
                                       ; непосредственное смещение
op{type}{cond} Rt, [Rn, #offset]!      ; pre-indexed
                                       ; пре-индексирование указателя
op{type}{cond} Rt, [Rn], #offset           ; post-indexed
                                       ; пост-индексирование указателя
opD{cond} Rt, Rt2, [Rn {, #offset}]    ; immediate offset, two words
                                       ; непосредственное смещение
                                       ; пересылка двух слов (D)
opD{cond} Rt, Rt2, [Rn, #offset]!      ; pre-indexed, two words
                                       ; пре-индексирование указателя
                                       ; пересылка двух слов(D)
opD{cond} Rt, Rt2, [Rn], #offset           ; post-indexed, two words
                                       ; пост-индексирование указателя,
                                       ; пересылка двух слов(D)
```

Где:

op оптокод один из:
- **LDR** - Load Register (Загрузка регистра)
- **STR** - Store Registe (Сохранение регистра)

cond код условного выполнения команды (опция)

type тип операции,один из:
- **B** - Unsigned byte, zero extend to 32 bits on loads
  Без знаковый байт, расширенный нулем до 32 бит в процессе загрузки;
- **SB** - Signed byte, sign extend to 32 bits (LDR only)
  Байт со знаком, расширенный знаком до 32 бит (только для команды LDR);
- **H** - Unsigned halfword, zero extend to 32 bits on loads
  Полуслово без знака, расширенное нулем до 32 бит в процессе загрузки;
- **HS** - Signed halfword, sign extend to 32 bits (LDR only)
  Полуслово со знаком, расширенное знаком до 32 бит (только для команды LDR);
- \- Omit, for word
  Отсутствует, если производится операция со словом (32-х разрядным.

Rt - Is the register to load or store
Регистр назначения, для загрузки или сохранения данных

Rn - Is the register on which the memory address is based
Базовый регистр, в котором содержится адрес памяти

offset - Is an offset from Rn. If offset is omitted, the address is the contents of Rn.
Смещение относительно значения в базовом регистре Rn.

		Если смещение не указано, то адрес доступа к памяти есть содержимое базового регистра
	Rt2 -	Is the additional register to load or store for two-word operations Второй регистр назначения(для загрузки или сохранения) для операций с двумя словами одновременно

Операция

LDR - загрузка одного или двух регистров значениями из памяти.
STR - сохранение содержимого одного или двух регистров в памяти.

Команды загрузки и сохранения с использованием непосредственного смещения содержимого базового регистра (указателя) могут использовать следующие режимы адресации:

Offset addressing - Базовая адресация со смещением

Значение смещения добавляется или вычитается (в зависимости от знака смещения) к текущему значению в базовом регистре Rn. Результат используется как адрес доступа к памяти. Значение в базовом регистре Rn не изменяется. В ассемблере для задания базовой адресации со смещением используется следующий синтаксис:

```
[Rn, #offset]
```

Pre-indexed addressing – Базовая адресация с пре-индексированием

Значение смещения добавляется или вычитается (в зависимости от знака смещения) к текущему значению в базовом регистре Rn. Результат используется как адрес доступа к памяти и *автоматически сохраняется в базовом регистре Rn*. В ассемблере для задания базовой адресации с пре-индексированием используется следующий синтаксис:

```
[Rn, #offset]!
```

Post-indexed addressing – Базовая адресация с пост-индексированием

Значение в базовом регистре Rn используется в качестве адреса для доступа к памяти. Значение смещения добавляется или вычитается (в зависимости от знака смещения) из значения в базовом регистре после выполнения операции пересылки и записывается в базовый регистр Rn. В ассемблере для этого способа адресации используется следующий синтаксис:

```
[Rn], #offset
```

Значение, которое используется при загрузке или сохранении данных может быть байтом (byte), полусловом (halfword), словом (word) или двойным словом (two words). Байты и полуслова могут быть числами без знака или со знаком в дополнительном коде.
В таблице ниже показан диапазон возможных величин смещений для трех способов адресации: со смещением содержимого базового адреса; с пре-индексированием; с пост-индексированием.

Тип команды	**Непосредственное смещение относительно базы**	**Пре-индексирование**	**Пост-индексирование**
Пересылка слова, полуслова, полуслова со знаком, байта, байта со знаком	-255 ÷ 4095	-255 ÷ 255	-255 ÷ 255
Пересылка двух слов (двойного слова)	Кратно 4-м в диапазоне -1020 ÷ 1020	Кратно 4-м в диапазоне -1020 ÷ 1020	Кратно 4-м в диапазоне -1020 ÷ 1020

Ограничения
Для операций загрузки: ■ Rt (регистр приемник данных) может быть указателем стека SP или счетчиком команд PC только для операций загрузки слов. Эти регистры SP и PC нельзя загружать байтами или полусловами. ■ Rt должен отличаться от Rt2 в операциях пересылки двух слов одновременно. ■ Базовый регистр Rn должен отличаться от регистров назначения Rt и Rt2 в операциях с пре- и пост-индексированием. Если в операции загрузки слова в качестве регистра-приемника Rt используется счетчик команд PC: ■ Бит bit[0] в загружаемом значении должен быть равен 1 для дальнейшего корректного выполнения программы (подтверждение режима работы с набором команд Thumb-2). ■ Переход автоматически произойдет на правильный адрес, который образуется путем очистки нулевого бита bit[0]=0. ■ Если команда является командой условного выполнения, то она должна быть последней в блоке условного выполнения IT. Для команд сохранения: ■ Регистр-источник Rt может быть счетчиком команд SP только при операции сохранения слов (не байт и полуслов). ■ Регистр-источник Rt не может быть счетчиком команд PC. ■ Регистр базы Rn не должен быть счетчиком команд PC. ■ Регистр базы Rn должен отличаться от регистров назначения Rt и Rt2 в операциях с пре- и пост-индексированием.
Выработка флагов
Эти команды не изменяют состояние флагов

Примеры

```
LDR R8, [R10]              ; Загрузка регистра R8 из памяти по
                           ; адресу в регистре R10 – стандартная
                           ; косвенная адресация или базовая с нулевым
                           ; смещением.
LDRNE R2, [R5, #960]!      ; Загрузка (условная, по условию NE)
                           ; регистра R2 словом из памяти по адресу
                           ; в регистре R5, смещенному на 960.
                           ; Адрес в R5 инкрементируется на 960.
                           ; Косвенная адресация с пре-индексированием
                           ; указателя.
                           ; Восклицательный знак «!» – отличительный
                           ; признак базовой адресации с пре-
                           ; индексированием.
STR R2, [R9,#const-str]    ; Сохранить содержимое регистра R2
                           ; в памяти по адресу в регистре R9,
                           ; смещенному на величину выражения
                           ; const-str, результат вычисления
                           ; которого - константа в диапазоне
                           ; 0-4095.
                           ; Базовая адресация со смещением
                           ; указателя на значение выражения
                           ; (вычисляется на стадии трансляции)
STRH R3, [R4], #4          ; Сохранить содержимое регистра R3 как
                           ; полуслово (halfword)в памяти по
                           ; адресу в регистре R4, а затем
                           ; пост-инкрементировать указатель на 4.
                           ; Базовая адресация с пост-смещением
                           ; (пост-инкрементированием) указателя
LDRD R8, R9, [R3, #0x20]   ; Двойная загрузка регистров (суффикс D):
                           ; регистра R8 из памяти по адресу в
                           ; регистре R3, смещенному на 32(0x20)
                           ; и регистра R9 следующим словом из
                           ; памяти по адресу, смещенному на 36
                           ; (36= 32+4) - смещение второго слова.
                           ; Базовая адресация со смещением
STRD R0, R1, [R8], #-16    ; Двойное сохранение содержимого
                           ; регистров R0 и R1. Содержимое R0
                           ; сохраняется по указателю в регистре R8,
                           ; а содержимое R1 по адресу следующего
                           ; слова, т.е. со смещением на 4 байта.
                           ; После выполнения этих двух операций
                           ; выполняется пост-декрементирование
                           ; указателя (R8) на 16.
                           ; Базовая адресация с пост-смещением
                           ; (в данном случае с
                           ; пост-декрементированием) указателя
```

1.6 Загрузка регистров ЦПУ из памяти и сохранение в памяти по содержимому базового регистра и смещению в индексном регистре

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDR and **STR** (Register Offset)	Загрузка и сохранение регистра с использованием базового и индексного регистров	v	v	v

Синтаксис

```
op{type}{cond} Rt, [Rn, Rm {, LSL #n}]
```

Где:

op оптокод один из:
- **LDR** - Load Register (Загрузка регистра)
- **STR** - Store Registe (Сохранение регистра)

cond код условного выполнения команды (опция)

type тип операции один из:
- **B** - Unsigned byte, zero extend to 32 bits on loads
  Без знаковый байт, расширенный нулем до 32 бит в процессе загрузки;
- **SB** - Signed byte, sign extend to 32 bits (LDR only)
  Байт со знаком, расширенный знаком до 32 бит (только для команды LDR);
- **H** - Unsigned halfword, zero extend to 32 bits on loads
  Полуслово без знака, расширенное нулем до 32 бит в процессе загрузки;
- **SH** - Signed halfword, sign extend to 32 bits (LDR only)
  Полуслово зо знаком, расширенное знаком до 32 бит (только для команды LDR);
- \- Omit, for word
  Отсутствует, если производится операция со словом (32-х разрядным).

Rt - Is the register to load or store
Регистр назначения, для загрузки или сохранения данных.

Rn - Is the register on which the memory address is based
Регистр базы, в котором содержится базовый адрес памяти.

Rm - Is a register containing a value to be used as the offset
Индексный регистр, содержащий значение смещения (или индекс данных в массиве).

LSL #n - Is an optional shift, with n in the range 0 to 3
Опционально заданный сдвиг, где n в диапазоне от 0 до 3
Обычно задается, если в Rm – индекс данных в массиве

Операция

LDR - загрузка регистра значением из памяти.
STR - сохранение содержимого регистра в памяти.

Адрес памяти для загрузки или сохранения данных определяется содержимым регистра базы Rn, который модифицируется на величину смещения (или индекса), заданного в регистре Rm.
Если задается индекс расположения данных в массиве, то, для автоматического получения смещения, он может быть сдвинут влево на заданное число бит (от 0 до 3) с помощью операции логического сдвига влево LSL. При этом происходит автоматическое попутное вычисление смещения offset= i x 2^n (умножение индекса на 1, 2, 4 или 8 в зависимости от длины пересылаемых

данных: байт, полуслово, слово, двойное слово). Полученное смещение добавляется к значению базового адреса.
Для команд пересылки байта или полуслова возможны версии работы с числами без знака или со знаком в дополнительном коде.
В этом режиме адресации работа с двойными словами не поддерживается.

Ограничения

■ Регистр базы Rn не должен быть счетчиком команд PC.
■ Регистр смещения (индексный) Rm не должен быть указателем стека SP или счетчиком команд PC.
■ Регистр назначения Rt может быть указателем стека SP только при пересылке слов (не байт или полуслов).
■ Регистр назначения Rt может быть счетчиком команд PC только при загрузке слов.
В этом случае:
■ Bit[0] в загружаемом значении должен быть равен 1 для корректности последующих операций. При этом переход произойдет с автоматической очисткой этого бита, т.е. адрес перехода будет автоматически выровнен по полуслову.
■ Если команда имеет суффикс условного выполнения, то она должна быть последней командой в IT-блоке.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
STR R0, [R5, R1]      ; Сохранить содержимое регистра R0 в памяти
                      ; по адресу, равному сумме базового адреса
                      ; в регистре R5 и смещения в регистре R1.
LDRSH R0, [R5, R1, LSL #1]      ; Загрузить регистр R0 значением
                                ; полуслова со знаком (SH)с
                                ; автоматическим расширением знака
                                ; на старшие разряды.
                                ; Базовый адрес массива знаковых
                                ; полуслов - в регистре базы R5,
                                ; индекс полуслова - в регистре R1.
                                ; Смещение определяется как удвоенное
                                ; значение индекса (R1, LSL #1):
                                ; offset= (R1) x 2^1.
STR R0, [R1, R2, LSL #2]        ; Сохранить содержимое регистра R0 в
                                ; i-м слове массива слов в памяти
                                ; с начальным адресом массива в
                                ; регистре базы R1 и индексом в
                                ; регистре индекса R2.
                                ; Значение смещения вычисляется
                                ; автоматически (R2, LSL #2) так:
                                ; offset= (R2) x 2^2.
```

1.7 Загрузка регистров из памяти и сохранение в памяти по содержимому базового регистра и непосредственному смещению с непривилегированным доступом

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDR and **STR** (Unprivileged Access	Загрузка и сохранение регистра (Непривилегированный доступ)	v	v	v

Синтаксис

```
op{type}T{cond} Rt, [Rn {, #offset}] ; непосредственное смещение
                                     ; суффикс «T» -
                                     ; непривилегированный доступ
```

Где:

op оптокод один из:
- **LDR** - Load Register (Загрузка регистра)
- **STR** - Store Registe (Сохранение регистра)

cond код условного выполнения команды (опция)

type тип операции один из:
- **B** - Unsigned byte, zero extend to 32 bits on loads
  Без знаковый байт, расширенный нулем до 32 бит в процессе загрузки;
- **SB** - Signed byte, sign extend to 32 bits (LDR only)
  Байт со знаком, расширенный знаком до 32 бит (только для команды LDR);
- **H** - Unsigned halfword, zero extend to 32 bits on loads
  Полуслово без знака, расширенное нулем до 32 бит в процессе загрузки;
- **SH** - Signed halfword, sign extend to 32 bits (LDR only)
  Полуслов зо знаком, расширенное знаком до 32 бит (только для команды LDR);
- **-** Omit, for word
  Опускается, если производится операция со словом (32-х разрядным).

Rt - Is the register to load or store
Регистр назначения, для загрузки или сохранения данных

Rn - Is the register on which the memory address is based
Регистр базы, в котором содержится базовый адрес памяти

offset – смещение относительно базового адреса в Rn, которое может быть в диапазоне от 0 до 255. Если смещение не указано, то используется базовый адрес (классическая косвенная адресация)

Операция

LDR - загрузка регистра значением из памяти.
STR - сохранение содержимого регистра в памяти.
Это точно такие же операции загрузки и сохранения, как операции с непосредственным смещением «LDR and STR (Immediate Offset)». Отличие состоит только в том, что доступ к памяти непривилегированный (unprivileged access), даже если команда используется в привилегированном программном обеспечении.
Если эти команды используются в непривилегированном программном обеспечении, то они выполнятся точно так же, как обычные инструкции доступа к памяти с использованием непосредственного смещения.
Особенность – блокирование доступа к системным ресурсам процессора во избежание некорректных изменений режима работы процессора.

Ограничения

■ Регистр базы Rn не должен быть счетчиком команд PC.
■ Регистр назначения (приемник или передатчик) Rt не должен быть указателем стека SP или счетчиком команд PC

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
STRBTEQ R4, [R7]      ; Команда условного сохранения (если
                      ; эквивалентно EQ)младшего байта (B) слова,
                      ; находящегося в регистре R4 по адресу в
                      ; базовом регистре R7, с непривилегированным
                      ; доступом (T).
LDRHT R2, [R2, #8]    ; Загрузка полуслова из памяти по адресу
                      ; в базовом регистре R2, смещенному на 8
                      ; с непривилегированным доступом (T).
```

1.8 Загрузка регистра/двух регистров из памяти с использованием относительной адресации по содержимому счетчика команд PC

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDR (PC-Relative)	Загрузка регистра с использованием относительной адресации по содержимому счетчика команд PC	V	V	V

Синтаксис

```
LDR{type}{cond} Rt, label
LDRD{cond} Rt, Rt2, label ; Загрузка двух слов одновременно
```

Где:

op оптокод один из:
- **LDR** - Load Register (Загрузка регистра)
- **LDRD** – Загрузка двух регистров

cond код условного выполнения команды (опция)

type тип операции один из:
- **B** - Unsigned byte, zero extend to 32 bits on loads
  Без знаковый байт, расширенный нулем до 32 бит в процессе загрузки
- **SB** - Signed byte, sign extend to 32 bits (LDR only)
  Байт со знаком, расширенный знаком до 32 бит (только LDR)
- **H** - Unsigned halfword, zero extend to 32 bits on loads
  Половина слова без знака, расширенная нулем до 32 бит в процессе загрузки
- **SH** - Signed halfword, sign extend to 32 bits (LDR only)
  Половина слова со знаком, расширенная знаком до 32 бит (только LDR)
- **-** Omit, for word
  Не указывается, если производится операция со словом (32-х разрядным)

Rt - Регистр назначения, приемник данных

Rt2 – Второй регистр-приемник при загрузке двух слов одновременно

Label – Метка. Относительное выражение по содержимому счетчика команд PC.

Операция

Команда **LDR** выполняет загрузку регистра данными из памяти по адресу относительно текущего значения в счетчике команд PC и заданному в коде операции смещению. Адрес может быть специфицирован в виде метки или в виде смещения относительно текущего значения в счетчике команд PC. В любом случае смещение автоматически рассчитывается транслятором на этапе Ассемблирования программы и сохраняется в формате команды.
Значение для загрузки может быть байтом, полусловом или словом (32 бита). Инструкции загрузки байта и полуслова могут быть как для чисел без знака (unsigned), так и со знаком в дополнительном коде (signed).
Метка должна быть расположена внутри ограниченного диапазона адресов по отношению к текущей инструкции. Ниже показан возможный диапазон смещений между меткой и текущим значением счетчика команд PC:

Тип команды	Диапазон возможных смещений
Слово, полуслово, полуслово со знаком, байт, байт со знаком	-4095 ÷ 4095
Два слова одновременно	-1020 ÷ 1020

Замечание:
Вы можете использовать дополнительный суффикс ".W» для того, чтобы получить максимально возможный диапазон смещений.

Внимание!
При использовании псевдокоманды Ассемблера для загрузки непосредственных данных в регистры ЦПУ (LDR Rd,=Const) транслятор автоматически размещает 32-разрядные константы в свободной области ПЗУ (в виде «литеральных пулов») и генерирует команды доступа к ним с использованием относительной адресации по содержимому счетчика команд, описанные выше. Смещение до места расположения констант также автоматически рассчитывается транслятором.

Ограничения

■ Регистр назначения (приемник) Rt может быть указателем стека SP или счетчиком команд PC только при загрузке слов (32-разрядных).
■ Второй регистр приемник Rt2 не может быть указателем стека SP или счетчиком команд PC.
■ Регистры Rt и Rt2 должны отличаться (при загрузке сразу двух слов).
Если регистр назначения Rt является счетчиком команд PC при загрузке слов, то:
■ Bit[0] в загружаемом значении должен быть равен 1 для последующего корректного выполнения программы. При этом, переход автоматически произойдет по адресу, выравненному по полуслову (т.е. с очищенным младшим битом).
■ Если команда является условной, то она должна быть последней командой в блоке IT.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
LDR R0, Table          ; Загрузка регистра R0 словом данных по
                       ; адресу, отмеченному меткой Table
LDRSB R7, Data_1       ; Загрузка регистра R7 байтом данных,
                       ; расширенным знаком до 32-х разрядного
                       ; слова из области памяти, помеченной
                       ; меткой Data_1
```

1.9 Команды множественной загрузки регистров из памяти и сохранения в памяти

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDM и **STM**	Загрузка и сохранение нескольких регистров (множественная)	v	v	v

Синтаксис

```
op{addr_mode}{cond} Rn{!}, reglist
```

Где:	**op** оптокод один из: **LDM** - Load Multiple registers – Загрузка нескольких регистров **STM** - Store Multiple registers – Сохранение нескольких регистров	**cond** код условного выполнения команды (опция)
	addr_mode – режим адресации, один из: **IA** - Increment address After each acces – Инкрементирование адреса после каждого доступа. Это режим по умолчанию. **DB** - Decrement address Before each access – Декрементирование адреса перед каждым доступом.	
	Rn – Базовый регистр, содержащий базовый адрес области памяти	
	! – Опция – суффикс обратной записи в базовый регистр (writeback suffix). Если опция «!» присутствует, то окончательный адрес, который получается при множественной загрузке или сохранении данных (при пост-инкрементировании или пре-декрементаровании), записывается обратно в базовый регистр Rn.	
	reglist – список из одного или нескольких регистров для загрузки или сохранения, заключенный в фигурные скобки. Список может содержать диапазон регистров. Имена регистров или диапазонов регистров должны быть отделены друг от друга запятой.	

Операция

Команда **LDM** загружает регистры, указанные в листинге *reglist*, словами из памяти начиная с адреса в базовом регистре Rn.
Команда **STM** сохраняет содержимое регистров, указанных в листинге *reglist* в памяти, начиная с адреса в базовом регистре Rn.

Для команд LDM, LDMIA, LDMFD, STM, STMIA и STMEA (с пост-инкрементированием) диапазон адресов, используемый для доступа к памяти, представляет собой 4-х байтовый интервал от Rn до Rn + 4 * (n-1), где n – число регистров в листинге.
Доступ к памяти производится в порядке возрастания номеров регистров: регистр с самым младшим номером используется для доступа к ячейке памяти с наименьшим адресом, регистр с самым большим номером используется для доступа к ячейке памяти с наибольшим адресом.
Если специфицируется суффикс обратной записи {!}, то значение последнего адреса доступа к памяти Rn + 4 * (n-1) записывается в базовый регистр Rn.

Для команд LDMDB, LDMEA, STMDB, and STMFD (с пре-декрементированием), диапазон адресов, используемых для доступа к памяти, представляет собой 4-х байтовый интервал от Rn до Rn - 4 * (n-1), где n – число регистров в листинге.
Доступ производится в порядке уменьшения номеров регистров: регистр с наибольшим номером используется для доступа к самому старшему адресу памяти, а регистр с наименьшим номером, - для доступа к ячейке памяти по наименьшему адресу.
Если специфицируется суффикс обратной записи {!}, то значение последнего адреса доступа к памяти Rn - 4 * (n-1) записывается в базовый регистр Rn.

Команды сохранения в стеке PUSH или извлечения из стека POP могут быть выражены в формате этой команды.

Ограничения

■ Регистр базы Rn не может быть счетчиком команд PC.
■ Листинг регистров *reglist* не должен содержать указателя стека SP.
■ Для любой команды сохранения STM, листинг регистров не должен содержать счетчик команд PC
■ Для любой команды загрузки LDM, листинг регистров не должен содержать счетчик команд PC, если он содержит LR – регистр адреса возврата из подпрограммы.
■ Листинг регистров не должен содержать базовый регистр Rn, если Вы специфицировали суффикс обратной записи {!}.

Если счетчик команд PC включен в листинг регистров при операции загрузки LDM, то:
■ Bit[0] значения которое загружается в PC должен быть равен 1 для корректности операции. При этом переход будет происходить на адрес, выравненный по половине слова.
■ Если команда является условной, то она должна быть последней командой в IT-блоке.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры правильные

```
LDM R8,{R0,R2,R9}          ; LDMIA синоним команды LDM.
                           ; Загрузить регистры R0, R2, R9
                           ; словами из памяти начиная с базового
                           ; адреса в R8. Авто-инкрементирование
                           ; адреса после каждого доступа (+4)
STMDB R1!,{R3-R6,R11,R12}  ; Сохранить значения специфицированных
                           ; регистров в памяти по базовому
                           ; адресу  в R1 с пре-
                           ; декрементированием. Запись данных в
                           ; порядке от старшего регистра к
                           ; младшему. В конце обновить значение
                           ; базового регистра R1
```

Примеры неправильные

```
STM R5!,{R5,R4,R9}    ; Значение, которое сохраняется в базовом
                      ; регистре R5 является непредсказуемым.
                      ; Листинг регистров не должен содержать
                      ; имя базового регистра
LDM R2, {}            ; В листинге регистров для загрузки нет ни
                      ; одного имени
```

1.10 Сохранение содержимого регистров в стеке и восстановление из стека

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
PUSH и **POP**	Сохранить в стеке содержимое регистров с пре-декрементированием и восстановить содержимое регистров из стека с пост-инкрементированием	V	V	V

Синтаксис

```
PUSH{cond} reglist
POP{cond} reglist
```

Где:	***Reglist*** – Не пустой листинг регистров, заключенный в фигурные скобки. Он может содержать диапазон регистров. Если список содержит больше чем один регистр или диапазон регистров, то имена регистров, или диапазонов должны быть разделены запятой.	**cond** код условного выполнения команды (опция)
	Команды **PUSH** и **POP** являются синонимами команд **STMDB** (сохранить в памяти несколько регистров с пре-декрементированием), **LDM** (загрузить несколько регистров из памяти с пост-инкрементированием) или **LDMIA** – загрузить из памяти несколько регистров с пост-инкрементированием адреса. При этом в качестве базового регистра используется указатель стека SP, который при записи в стек автоматически пре-декрементируется, а при восстановлении данных из стека пост-инкрементируется. Последний адрес доступа всегда сохраняется в указателе стека. Указатель стека SP всегда содержит адрес последней заполненной данными ячейки памяти – вершины стека. При сохранении данных в стеке они записываются в сторону меньших адресов. Мнемоника команд PUSH и POP является предпочтительной при сохранении/восстановлении данных в стеке/из стека.	

Операция

По команде **PUSH** содержимое специфицированных в листинге регистров сохраняется в стеке в порядке уменьшения номеров регистров так, что самый старший регистр сохраняется в памяти по наибольшему адресу, а самый младший регистр – по наименьшему адресу.

По команде **POP** содержимое специфицированных в листинге регистров восстанавливается из стека в порядке возрастания номеров регистров так, что регистр с наименьшим номером загружается из ячейки памяти с наименьшим адресом, а регистр с наибольшим номером – из ячейки памяти с наибольшим адресом.

Это аналоги команд LDM и STM с автоматическим использованием в качестве базового регистра указателя стека SP и авто-обновлением указателя при каждом доступе в память – при сохранении данных: пре-декрементирование; при восстановлении: пост-инкрементирование.

Ограничения

- Листинг регистров не должен содержать указателя стека SP
- Для команды PUSH листинг регистров не должен содержать счетчик команд PC.
- Для команды POP листинг регистров не должен содержать счетчик команд PC, если он содержит LR – регистр адреса возврата из подпрограммы.

Если все же счетчик команд PC включен в листинг регистров команды POP:

■ Bit[0] в значении, которое загружается в счетчик команд PC должно быть равным 1 для дальнейшего корректного выполнения программы, хотя переход происходит адресу, выравненному по полуслову. Это необходимо для подтверждения работы процессора с набором команд Thumb-2.
■ Если команда является условной, то она должна быть последней командой в IT-блоке.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
PUSH {R0,R4-R7}       ; Сохранить в стеке регистры R0 и R4-R7
PUSH {R2,LR}          ; Сохранить в стеке регистры R2 и LR
POP {R0,R10,PC}       ; Восстановить из стека содержимое регистров
                      ; R0, R10 и счетчика команд PC
```

1.11 Эксклюзивная загрузка и сохранение регистра

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
LDREX и STREX	Эксклюзивная загрузка и сохранение регистра	v	v	v

Синтаксис

```
LDREX{cond} Rt, [Rn {, #offset}]
STREX{cond} Rd, Rt, [Rn {, #offset}]
LDREXB{cond} Rt, [Rn]
STREXB{cond} Rd, Rt, [Rn]
LDREXH{cond} Rt, [Rn]
STREXH{cond} Rd, Rt, [Rn]
```

Где:

LDREX – эксклюзивная загрузка слова **STREX** – эксклюзивное сохранение слова **LDREXB** – эксклюзивная загрузка байта **STREXB** – эксклюзивное сохранение байта **LDREXH** – эксклюзивная загрузка полуслова **STREXH** – эксклюзивное сохранение полуслова	**cond** – код условного выполнения команды (опция)
Rd – регистр назначения для возврата статуса эксклюзивного сохранения (только для команд эксклюзивного сохранения)	
Rt – Регистр для загрузки или сохранения данных	
Rn – Регистр базы, в котором содержится базовый адрес памяти	
offset – Опционально заданное смещение адреса в базовом регистре Rn. Если смещение не указано, то для доступа к памяти используется адрес в базовом регистре Rn (обычная косвенная адресация). Базовая адресация с использованием непосредственного смещения возможна только для эксклюзивной загрузки/сохранения полных 32-разрядных слов. Для обеих команд смещение должно быть одинаковым (доступ к одному и тому же слову). При работе с байтами или полусловами возможна только обычная косвенная адресация.	

Операция

Команды **LDREX**, **LDREXB** и **LDREXH** выполняют загрузку слова, байта или полуслова, соответственно, из памяти в регистр назначения Rt.
Команды **STREX**, **STREXB** и **STREXH** выполняют сохранение слова, байта или полуслова, соответственно, из регистра Rt в память.
Адрес ячейки памяти, который используется в какой-либо команде эксклюзивного сохранения (Store-Exclusive), должен быть тем же самым адресом, который использовался в последней выполненной команде эксклюзивной загрузки (Load-exclusive). Содержимое базового регистра и величина смещения не должны изменяться в промежутке между командой загрузки и сохранения. Это означает, что в программе всегда должна быть команда эксклюзивной загрузки (Load-exclusive) и соответствующая ей команда эксклюзивного сохранения (Store-Exclusive) – должна выполняться синхронизация операций пере-инициализации парой команд.
Если команда эксклюзивного сохранения (Store-Exclusive) выполняет сохранение данных в памяти, то она записывает 0 в свой регистр назначения Rd, гарантируя тем самым, что никакой другой процесс в системе (прямой доступ в память или прерывание) не имел доступа к ячейке памяти между операцией эксклюзивной загрузки (Load-exclusive) и операцией эксклюзивного сохранения (Store-Exclusive).
Если она не выполняет сохранение данных, то она записывает 1 в свой регистр назначения Rd. Это означает, что был доступ к ячейке памяти от стороннего процесса (например, по прерыванию) и запись не состоялась, - нужно ее повторить.
Внимание!

Пара команд эксклюзивной загрузки/сохранения используется для того, чтобы гарантированно поменять значение какой-либо «важной» переменной в памяти или в периферийном регистре, без права доступа к ней любого другого процесса (например, процедуры обслуживания прерывания) в промежутке времени между считыванием этой переменной, ее модификацией и последующей перезаписью (пере-инициализацией).
Это типичная процедура для пере-инициализации важных регистров периферийных устройств по технологии «Чтение-Модификация-Запись».
Для повышения производительности уменьшайте число команд между командой эксклюзивной загрузки Load-Exclusive и соответствующей ей командой эксклюзивного сохранения Store-Exclusive до минимума.

Замечание: Результат выполнения команды эксклюзивного сохранения Store-Exclusive по адресу, который отличается адреса, используемого в команде эксклюзивной загрузки Load-Exclusive, является непредсказуемым.

Ограничения

■ Не используйте счетчик команд PC.
■ Не используйте указатель стека SP для регистра возврата статуса операции Rd и регистра для записи или считывания Rt.
■ Для команды эксклюзивного сохранения STREX, регистр возврата статуса Rd должен отличаться от регистра источника данных Rt и базового регистра Rn.
■ Значение смещения должно быть кратно 4 в диапазоне 0-1020.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
      MOV R1, #0x1                ; Гарантированная инициализация
                                  ; «важной» переменной Var5 значением 1
      LDR R5, Var5                ; Загрузить адрес переменной в регистр
                                  ; указатель R5
Loop                              ; Цикл записи при необходимости
      LDREX R0, [R5]              ; Загрузка исходного значения Var5
                                  ; в регистр R0
                                  ; После этой команды доступ к
                                  ; ячейке памяти Var5 других процессов
                                  ; начинает контролироваться
      CMP R0, #0                  ; Ячейка пуста? Только в этом случае
                                  ; будет выполняться запись
      ITT EQ                      ; IT-блок для команд STREXEQ и CMPEQ
                                  ; Проверяется одно условие EQ
      STREXEQ R0, R1, [R5]        ; Пере-инициализировать переменную
                                  ; VAR5 содержимым регистра R1, «1»
                                  ; Вернуть статус операции
                                  ; эксклюзивного сохранения в регистр
                                  ; R0
      CMPEQ R0, #0                ; Операция сохранения прошла успешно?
                                  ; В регистре статуса операции
                                  ; эксклюзивного сохранения R0 0?
      BNE Loop                    ; Нет – еще раз повторить запись
      ...                         ; Да – операция записи прошла успешно
```

1.12 Отмена режима эксклюзивности

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
CLREX	Отмена режима эксклюзивности. Clear Exclusive.	v	v	v

Синтаксис

CLREX{cond}

Где:	**CLREX** – сброс режима эксклюзивности	**cond** – код условного выполнения команды (опция)

Операция

Используйте команду **CLREX** для того, чтобы заставить следующую за ней команду эксклюзивного сохранения STREX, STREXB или STREXH записать 1 в свой регистр статуса операции, что будет свидетельствовать о том, что операция сохранения не выполнена. Рекомендуется использовать эту команду в кодах обработки исключений для принудительного ошибочного эксклюзивного сохранения – тогда, когда исключение возникает между операцией эксклюзивной загрузки и соответствующей ей операцией эксклюзивного сохранения при выполнении операций синхронизации.

Выработка флагов

Эта команда не изменяет состояние флагов

Примеры

```
        CLREX       ; Сбросить режим эксклюзивности
```

2 АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Оглавление

2.1. Сложение слов, сложение слов с учетом переноса, вычитание слов, вычитание слов с учетом заема, реверсивное вычитание слов 27

2.2. Команды сравнения слов 30

2.3. Параллельное сложение 2-х пар полуслов со знаком или 4-х пар байт со знаком 31

2.4. Параллельное сложение 2-х пар полуслов без знака или 4-х пар байт без знака 32

2.5. Параллельное сложение 2-х пар полуслов со знаком или 4-х пар байт со знаком с делением сумм на два 34

2.6. Параллельное вычитание 2-х пар полуслов со знаком или 4-х пар байт со знаком с делением разностей на два 35

2.7. Параллельное перекрестное сложение и вычитание двух пар полуслов со знаком 36

2.8. Параллельное перекрестное сложение и вычитание двух пар полуслов без знака 37

2.9. Параллельное перекрестное сложение и вычитание двух пар полуслов со знаком с делением результатов на два 39

2.10. Параллельное перекрестное сложение и вычитание двух пар полуслов без знака с делением результатов на два 41

2.11. Селектирование байт или полуслов по состоянию флагов GE 43

2.12. Сложение абсолютных разностей 4-х пар байт без знака 44

2.13. Сложение абсолютных разностей 4-х пар байт без знака и накопление суммы 45

2.14. Подсчет числа лидирующих нулей 46

2.1 Сложение слов, сложение слов с учетом переноса, вычитание слов, вычитание слов с учетом заема, реверсивное вычитание

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
ADD, ADC, SUB, SBC и RSB	Add, Add with carry , Subtract, Subtract with carry , and Reverse Subtract. Сложение, Сложение с переносом, Вычитание, Вычитание с переносом (точнее – «заемом») и Реверсное вычитание	v	v	v

Синтаксис

```
op{S}{cond} {Rd,} Rn, Operand2
op{cond} {Rd,} Rn, #imm12              ; Только для ADD и SUB
```

Где:

op оптокод один из: **ADD** Add - Сложение **ADC** Add with Carry - Сложение с учетом переноса **SUB** Subtract- Вычитание **SBC** Subtract with Carry - Вычитание с учетом флага переноса («заема») **RSB** Reverse Subtract - Реверсное вычитание	**cond** код условного выполнения команды (опция)
S Опциональный суффикс (Set Flags). Если он специфицирован, то флаги кодов условного выполнения обновляются по результату операции.	
Rd - Регистр назначения. Если он не указан, то регистром назначения будет регистр-источник Rn.	
Rn - Регистр, содержащий первый операнд.	
Operand2 - Гибкий второй операнд ("Flexible Second Operand"): константа, регистр ЦПУ, регистр с опцией сдвига.	
imm12 - Непосредственный 12-разрядный операнд в диапазоне 0-4095	

Операция

ADD	добавляет к первому операнду в регистре Rn значение второго операнда Operand2 или непосредственной константы imm12
ADC	добавляет к первому операнду в регистре Rn значение второго операнда Operand2 и значение флага переноса C
SUB	вычитает из первого операнда в регистре Rn значение второго операнда Operand2 или непосредственной константы imm12
SBC	вычитает из первого операнда в регистре Rn значение второго операнда Operand2 и значение флага «заема» (инверсное значение флага переноса C). Если флаг переноса был очищен, то это означает, что в предыдущей операции возник «заем».
RSB	Реверсное вычитание: из значения второго операнда Operand2 вычитается значение первого операнда. Команда применима к широкому диапазону опций для второго операнда.

Команды ADC и SBC (сложения и вычитания с учетом переноса/заема) применяются при работе с числами из нескольких слов – в *мульти-словной арифметике* (multiword arithmetic).

Замечания:
Команда **ADDW** эквивалентна по синтаксису команде **ADD**, которая использует непосредственный 12-битный операнд imm12.
Команда **SUBW** эквивалентна по синтаксису команде **SUB**, которая использует непосредственный 12-битный операнд imm12.

Ограничения

■ Второй операнд Operand2 не должен быть указателем стека SP или счетчиком команд PC.
■ Регистр назначения Rd может быть указателем стека SP только в командах ADD и SUB и только при следующих дополнительных ограничениях:
– Первый операнд Rn также должен быть указателем стека SP.
– Величина сдвига второго гибкого операнда Operand2 должна быть ограничена максимум 3-мя битами при использовании логического сдвига влево LSL.
■ Первый операнд Rn может быть указателем стека SP только в командах ADD и SUB.
■ Регистр-приемник Rd может быть счетчиком команд PC только в команде **ADD{cond} PC, PC, Rm** при дополнительных условиях:
– Вы не должны специфицировать суффикс S установки флагов.
– Второй регистровый операнд Rm не должен быть PC или SP.
– Если команда является условной, то она должна быть последней в блоке условного выполнения IT-блоке.
■ За исключением команды **ADD{cond} PC, PC, Rm**, первый операнд Rn может быть счетчиком команд PC только в командах ADD и SUB, причем при следующих дополнительных ограничениях:
– Вы не должны специфицировать суффикс S установки флагов.
– Второй операнд должен быть константой в диапазоне от 0 до 4095.

Замечания:
- Если Вы используете счетчик команд PC в операциях сложения или вычитания, то два младших бита bits[1:0] счетчика команд перед выполнением операции сложения или вычитания обнуляются, т.к. все адреса, используемые в вычислениях, выравниваются по слову (word-aligned).

- Если Вы хотите сгенерировать адрес команды, то Вы должны отъюстировать константу, базируясь на значении в счетчике команд PC. Фирма ARM рекомендует вместо команды ADD или SUB с первым операндом Rn, равным PC, использовать команду ADR. В этом случае ассемблер автоматически вычислит правильное значение константы для команды ADR.

- Если регистр назначения Rd является счетчиком команд PC в инструкции **ADD{cond} PC, PC, Rm** то:
■ Bit[0] в значении, которое загружается в PC, игнорируется.
■ Переход выполняется по адресу, с принудительно очищенным нулевым битом bit[0]=0.
Это связано с тем, что процессоры Cortex-M3/M4/M4F поддерживают только один набор команд Thumb-2, объединяющий наборы ARM и Thumb. Нулевой бит в операциях передачи управления в предыдущих поколениях процессоров использовался для переключения процессора в режим использования нужного набора команд. В процессорах Cortex-M подтверждается работа только с набором команд Thumb-2. Нулевой бит должен быть сброшен всегда.

Выработка флагов

Если суффикс S специфицирован, то эти команды обновляют значения флагов N, Z, C и V в соответствии с результатом операции.

Примеры

```
ADD R2, R1, R3        ; Сложить содержимое регистров R1 и R3
                      ; Записать результат в R2
SUBS R8, R6, #240     ; Вычесть из содержимого регистра R6 число
                      ; 240, обновить флаги
```

```
      RSB R4, R4, #1280      ; Реверсное вычитание: Из второго операнда
                             ; (1280) вычесть содержимое регистра R4
                             ; Результат сохранить в R4
      ADCHI R11, R0, R3      ; Условная команда: Если HI (Флаг C
                             ; установлен и флаг Z сброшен), то значения
                             ; регистрах R0 и R3 суммируются с учетом
                             ; флага переноса и сохраняются в R11
```

Замечание: Если при вычитании возникает «заем», то флаг C обнуляется, в противном случае – устанавливается.

Примеры многословной арифметики

Пример 1.
Сложить два целых 64-разрядных числа (integer) в регистрах R1, R0 и в регистрах R3, R2. Результат сохранить в R5, R4:

64-р. 1-й операнд	R1 (ст. слово)	R0 (мл. слово)
64-р. 2-1 операнд	R3 (ст. слово)	R2 (мл. слово)
64-р. результат	R5 (ст. слово)	R4 (мл. слово)

```
      ADDS R4, R0, R2 ; Сложение младших слов в регистрах R0,R2
                      ; Выработка флага переноса
      ADC R5, R1, R3  ; Сложение старших слов в регистрах R1, R3 с
                      ; учетом возможного переноса
```

Пример 2.
Длинные слова не обязательно должны содержаться в последовательных регистрах. В примере ниже, первое 96-разрядное слово содержится в регистрах R8, R2, R6, а второе 96-разрядное слово в регистрах R11, R1, R9. Результат вычитания сохраняется в регистрах R2, R9, R6.

96-р. 1-й операнд	R8 (ст.)	R2 (ср.)	R6 (мл.)
96-р. 2-й операнд	R11 (ст.)	R1 (ср.)	R9 (мл.)
96-р. результат	R2 (ст.)	R9 (ср.)	R6 (мл.)

```
      SUBS R6, R6, R9 ; Вычитание младших значащих слов
                      ; Выработка флага «заема» (инверсный C)
      SBCS R9, R2, R1 ; Вычитание средних слов с учетом «заема»
                      ; И еще одна выработка флагов
      SBC R2, R8, R11 ; Вычитание старших слов с учетом «заема»
```

2.2 Команды сравнения слов

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
CMP и **CMN**	Compare and Compare Negative – Сравнение и Сравнение с отрицательным значением	v	v	v

Синтаксис

```
CMP{cond} Rn, Operand2
CMN{cond} Rn, Operand2
```

Где:	**cond** код условного выполнения команды (опция)
	Rn - Регистр, содержащий первый операнд
	Operand2 – Гибкий второй операнд: константа, регистр ЦПУ, регистр ЦПУ с предварительными сдвигом.

Операция

Команда **CMP** производит сравнение содержимого регистра Rn со вторым операндом Operand2. Она обновляет значения флагов, но никуда не записывает результат.
Команда CMP вычитает из содержимого регистра Rn (первый операнд) значение операнда 2 Operand2. Она эквивалентна команде **SUBS**, за исключением того, что результат вычитания не сохраняется (отбрасывается).

Команда **CMN** вычитает из содержимого регистра Rn (первый операнд) отрицательное значение операнда 2 Operand2, то есть прибавляет к первому операнду значение второго и выставляет флаги результата операции. Она эквивалентна команде **ADDS**, за исключением того, что результат суммирования не сохраняется (отбрасывается).

Ограничения

- Не использовать счетчик команд PC.
- Второй операнд Operand2 не должен быть указателем стека SP (первый операнд может).

Выработка флагов

Эта команда обновляет состояние флагов N, Z, C и V в соответствии с результатом операции

Примеры

```
CMP   R2, R9          ; Сравнить содержимое регистров R2 и R9
CMN   R0, #6400       ; Сравнить содержимое R0 с числом 6400
CMPGT SP, R7, LSL #2  ; Условная команда сравнения по GT
                      ; («Строго больше»)
                      ; Сравнивается содержимое указателя стека SP
                      ; с числом в R7, логически сдвинутым влево
                      ; на два разряда (умноженным на 4)
                      ; в R7 – число возможных слов в стеке
                      ; При умножении на 4 – число байт в стеке
```

2.3 Параллельное сложение 2-х пар полуслов со знаком или 4-х пар байт со знаком

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SADD16 and **SADD8**	Signed Add 16 and Signed Add 8 – Знаковое сложение 16-разрядных чисел и знаковое сложение 8-разрядных чисел		V	V

Синтаксис

```
op{cond}{Rd,} Rn, Rm
```

Где:		
	op оптокод один из: **SADD16** - Performs two 16-bit signed integer additions. Выполняет сложение двух пар целых 16-разрядных чисел со знаком **SADD8** - Performs four 8-bit signed integer ad. Выполняет сложение 4-х целых 8-разрядных чисел со знаком.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

Операция

Эти команды выполняют сложение 2-х полуслов или 4-х байт параллельно как бы в двух 16-разрядных или 4-х 8-разрядных АЛУ, используя обычное двоичное сложение. Сложение выполняется по модулю 2^{16}=65536 (для полуслов) и 2^{8}=256 (для байт), то есть переполнения игнорируются.

Команда **SADD16**:

1. Знаковое сложение каждого полуслова первого операнда с соответствующим полусловом второго операнда.
2. Запись результата в соответствующее полуслово регистра-назначения.

Команда **SADD8**:

1. Знаковое сложение каждого байта первого операнда с соответствующим байтом второго операнда.
2. Запись результата в соответствующий байт регистра-назначения.

Внимание!

Дополнительно по результатам параллельных операций в регистре статуса программы пользователя APSR выставляются специальные флаги GE[3:0] («Больше или эквивалентно»). Флаг этого условия полностью соответствует операции **ADDS**, выполняемой в 16-разрядном или 8-разрядном АЛУ. Флаги GE[3:0] можно использовать в команде селектирования байт/полуслов SEL для выбора нужных значений из каждой пары при загрузке данных в регистры (см. команду **SEL**).

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния основных флагов, так как выполняется параллельно несколько (две или 4-е) операции. Устанавливаются только дополнительные флаги GE[3:0] в регистре статуса программы пользователя APSR по результатам параллельно выполненных операций.

Примеры

```
SADD16 R1, R0     ; Знаковое сложение полуслов в регистре R1
                  ; с соответствующими полусловами в регистре
                  ; R0 и сохранение результатов в соответствующих
                  ; полусловах регистра R1
SADD8 R4, R0, R5  ; Знаковое сложение байт в регистре R0 и
                  ; соответствующих байт в регистре R5 и запись
                  ; результатов в соответствующие байты регистра
                  ; R4
```

2.4 Параллельное сложение 2-х пар полуслов без знака или 4-х пар байт без знака

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
UADD16 and **UADD8**	Unsigned Add 16 – Без знаковое сложение 16-разрядных полуслов Unsigned Add 8 – Без знаковое сложение 8-разрядных байт		V	V

Синтаксис

```
op{cond}{Rd,} Rn, Rm
```

Где:	**op** оптокод один из: **UADD16** - Performs two 16-bit unsigned integer additions. Выполняет сложение двух 16-разрядных целых чисел без знака. **UADD8** - Performs four 8-bit unsigned integer additions. Выполняет сложение 4-х 8-и разрядных чисел без знака.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

Операция

Эти команды выполняют сложение без знаковых полуслов или без знаковых байт параллельно, используя обычное двоичное сложение как будто бы в двух 16-разрядных или 4-х 8-и разрядных АЛУ. Сложение выполняется по модулю 2^{16}=65536 (для полуслов) и 2^{8}=256 (для байт), то есть переполнения игнорируются.

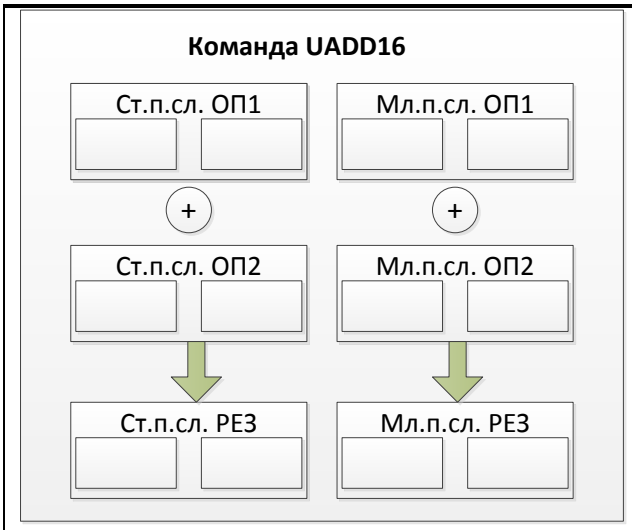
Команда **UADD16**:

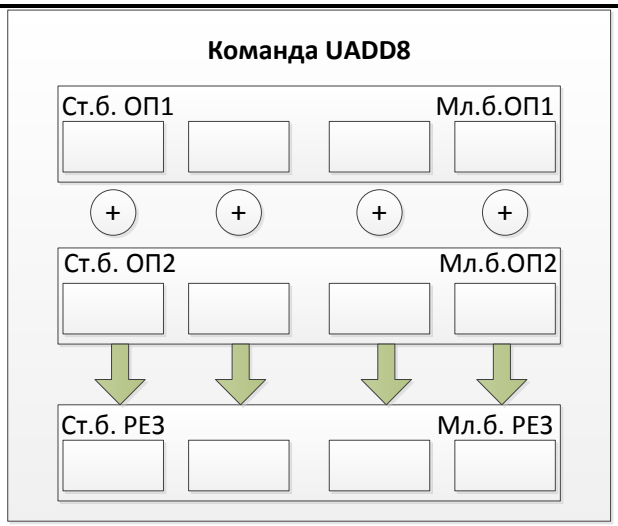
1. Сложение каждого полуслова первого операнда с соответствующим полусловом второго операнда.
2. Запись без знакового результата в соответствующее полуслово регистра-назначения.

Команда **UADD8**:

1. Сложение каждого байта первого операнда с соответствующим байтом второго операнда.
2. Запись без знакового результата в соответствующий байт регистра-назначения.

Графическая иллюстрация алгоритма выполнения команд:





Дополнительно в регистре статуса программы пользователя вырабатываются четыре флага GE[3:0] («Больше или эквивалентно»), аналогичные по смыслу флагам «**Выше или то же самое**» при операциях с числами без знака.
Флаг условия GE полностью соответствует операции **ADDS**, выполняемой в 16-разрядном или 8-разрядном АЛУ. Флаги GE[3:0] можно использовать в команде селектирования байт/полуслов SEL для выбора нужных значений из каждой пары при загрузке данных в регистры (см. команду **SEL**) по состоянию флагов.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние флагов, потому, что одновременно выполняется две или четыре операции, а регистр флагов операций – только один.
Изменяются только дополнительные флаги GE[3:0], отражающие результаты выполненных параллельных операций.

Примеры

```
UADD16 R1, R0     ; Складывает полуслова в регистре R1 с
                  ; соответствующими полусловами в регистре R0 и
                  ; сохраняет результаты в регистре R1
UADD8 R4, R0, R5  ; Складывает байты в регистре R0 с
                  ; соответствующими байтами в регистре R5 и
                  ; записывает в соответствующие байты регистра R4
```

2.5 Параллельное сложение 2-х пар полуслов со знаком или 4-х пар байт со знаком с делением сумм на два

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SHADD16 and SHADD8	Signed Halving Add 16 and Signed Halving Add 8 – Знаковое «уполовинивание» 2-х сумм полуслов и Знаковое «уполовинивание» 4-х сумм байт		V	V

Синтаксис

```
op{cond}{Rd,} Rn, Rm
```

Где:	**op** оптокод один из: **SHADD16** - Signed Halving Add. Знаковое «уполовинивание» (деление на два) 16-разрядных сумм полуслов **SHADD8** - Signed Halving Add 8. Знаковое «уполовинивание» (деление на два) 8-разрядных сумм байт	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

Операция

Используйте эти команды для параллельного знакового сложения 16-разрядных и 8-разрядных чисел, деления результата сложения на 2 и последующей записи «половинок» в регистр назначения.

Команда **SHADD16**:
1. Добавляет к каждому полуслову первого операнда соответствующее полуслово второго операнда.
2. Арифметически (с учетом знака) сдвигает каждый результат на один бит вправо, выполняя деление результатов-полуслов на два.
3. Записывает полученные полуслова в регистр назначения.

Команда **SHADD8**:
1. Добавляет к каждому байту первого операнда соответствующий байт второго операнда.
2. Арифметически (с учетом знака) сдвигает каждый результат-байт на один бит вправо, выполняя деление байтовых результатов на два.
3. Записывает полученные байты в регистр назначения.

Внимание!
Сложение выполняется с учетом расширения разрядной сетки 16-разрядного или 8-разрядного АЛУ на один разряд. Поэтому при арифметическом сдвиге вправо старший разряд суммы не теряется, а итоговый результат никогда не выходит за пределы 16-и или 8-разрядной сетки.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
SHADD16 R1, R0     ; Складывает полуслова в регистре R1 с
                   ; соответствующими полусловами в регистре R0,
                   ; делит полученные результаты на два и сохраняет
                   ; «половинки» в соответствующих полусловах
                   ; регистра R1
SHADD8 R4, R0, R5  ; Складывает байты в регистре R0 с байтами в
                   ; регистре R5, делит полученные результаты на 2
                   ; и сохраняет в соответствующих байтах регистра
                   ; R4
```

2.6 Параллельное вычитание 2-х пар полуслов со знаком или 4-х пар байт со знаком с делением разностей на два

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SHSUB16 and SHSUB8	Signed Halving Subtract 16 and Signed Halving Subtract 8 – Знаковое «уполовинивание» 2-х разностей полуслов и Знаковое «уполовинивание» 4-х разностей байт		v	v

Синтаксис

op{cond}{Rd,} Rn, Rm

Где:		
	op оптокод один из: **SHSUB16** - Signed Halving Sub 16. Знаковое «уполовинивание» (деление на два) 16-разрядных разностей полуслов **SHSUB8** - Signed Halving Sub 8. Знаковое «уполовинивание» (деление на два) 8-разрядных разностей байт	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

Операция

Используйте эти команды для параллельного знакового вычитания 16-разрядных и 8-разрядных чисел, деления результатов вычитания на 2 и последующей записи «половинок» в регистр назначения.

Команда **SHSUB16**:
1. Вычитает из каждого полуслова первого операнда соответствующее полуслово второго операнда.
2. Арифметически (с учетом знака) сдвигает каждый результат на один бит вправо, выполняя деление результатов на два.
3. Записывает полученные полуслова «половинок» в регистр назначения.

Команда **SHSUB8**:
1. Вычитает из каждого байта первого операнда соответствующий байт второго операнда.
2. Арифметически (с учетом знака) сдвигает каждый результат на один бит вправо, выполняя деление результатов на два.
3. Записывает полученные байты «половинок» в регистр назначения.

Внимание!

Вычитание выполняется с учетом расширения разрядной сетки 16-разрядного или 8-разрядного АЛУ на один разряд. Поэтому при арифметическом сдвиге вправо старший разряд разности не теряется, а итоговый результат никогда не выходит за пределы 16-и или 8-разрядной сетки.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
SHSUB16 R1, R0     ; Вычитает полуслова в регистре R0 из
                   ; соответствующих полуслов в регистре R1,
                   ; делит полученные результаты на два и сохраняет
                   ; «половинки» в соответствующих полусловах
                   ; регистра R1
SHSUB8 R4, R0, R5  ; Вычитает байты в регистре R5 из байтов в
                   ; регистре R0, делит полученные результаты на 2
                   ; и сохраняет в соответствующих байтах регистра
                   ; R4
```

2.7 Параллельное перекрестное сложение и вычитание двух пар полуслов со знаком

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SASX and **SSAX**	Signed Add and Subtract with Exchange – Знаковое сложение и вычитание полуслов с обменом (перекрестное) Signed Subtract and Add with Exchange – Знаковое вычитание и сложение полуслов с обменом (перекрестное)		V	V

Синтаксис

op{cond}{Rd,} Rn, Rm

Где:	**op** оптокод один из: **SASX** - Signed Add and Subtract with Exchange – Знаковое сложение и вычитание перекрестное полуслов, с «перекрещиванием» операндов. **SSAX** - Signed Subtract and Add with Exchange – Знаковое вычитание и сложение полуслов перекрестное, с «перекрещиванием» операндов.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

Операция

Команда **SASX**:
1. Добавляет к старшему знаковому полуслову первого операнда младшее знаковое полуслово второго операнда.
2. Записывает полученное знаковое полуслово в старшее полуслово регистра назначения.
3. Вычитает из младшего знакового полуслова первого операнда старшее знаковое полуслово второго операнда.
4. Записывает полученную разность в младшее полуслово регистра назначения.

Команда **SSAX**:
1. Вычитает из старшего знакового полуслова первого операнда младшее знаковое полуслово второго операнда.
2. Записывает полученный результат в старшее полуслово регистра назначения.
3. Добавляет к младшему знаковому полуслову первого операнда старшее знаковое полуслово второго операнда.
4. Сохраняет результат в младшем полуслове регистра назначения.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние основных флагов. Вырабатываются флаги GE[3:0] в регистре статуса программы пользователя APSR по аналогии с операциями **ADDS**, **SUBS**, которые как бы выполняются в двух 16-разрядных АЛУ параллельно.

Примеры

```
SASX R0, R4, R5 ; Добавляет к старшему полуслову в R4
                ; младшее полуслово в R5 и записывает
                ; результат в старшее полуслово регистра R0
                ; Вычитает из младшего полуслова в регистре
                ; R4 старшее полуслово в регистре R5 и
                ; сохраняет результат в
                ; младшем полуслове регистра R0
SSAX R7, R3, R2 ; Вычитает из старшего полуслова в регистре R3
                ; младшее полуслово в регистре R2 и записывает
                ; результат в старшее полуслово регистра R7
                ; Добавляет к младшему полуслову в регистре R3
                ; старшее полуслово в регистре R2 и сохраняет
                ; результат в младшем полуслове регистра R7
```

2.8 Параллельное перекрестное сложение и вычитание двух пар полуслов без знака

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
UASX and **USAX**	Add and Subtract with Exchange – Без знаковое сложение и вычитание с обменом (перекрещиванием) Subtract and Add with Exchange – Без знаковое вычитание и сложение с обменом (перекрещиванием)		V	V

Синтаксис

op{cond}{Rd,} Rn, Rm

Где:	**op** оптокод один из: **UASX** - Add and Subtract with Exchange - Без знаковое сложение и вычитание перекрестное **USAX** - Subtract and Add with Exchange - Без знаковое вычитание и сложение перекрестное	**cond** код условного выполнения команды (опция)
	Rd - Регистр назначения	
	Rn - Регистр, содержащий первый операнд	
	Rm - Регистр, содержащий второй операнд	

Операция

Команда **UASX**:
1. Складывает старшее полуслово первого операнда с младшим полусловом второго операнда.
2. Записывает результат без знакового сложения в старшее полуслово регистра назначения.
3. Вычитает из младшего полуслова первого операнда старшее полуслово второго операнда.
4. Записывает результат без знакового вычитания в младшее полуслово регистра назначения.

Команда **USAX**:
1. Вычитает из старшего полуслова первого операнда младшее полуслово второго операнда.
2. Записывает результат без знакового вычитания в старшее полуслово регистра назначения.
3. Складывает младшее полуслово первого операнда со старшим полусловом второго операнда.
4. Записывает результат без знакового сложения в младшее полуслово регистра назначения.

Ограничения

■ В команде нельзя использовать указатель стека SPи счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние флагов. Вырабатываются флаги GE[3:0] в регистре статуса программы пользователя APSR по аналогии с операциями **ADDS**, **SUBS**, которые как бы выполняются в двух 16-разрядных АЛУ параллельно.

Примеры

```
UASX R0, R4, R5 ; Сложение старшего полуслова первого операнда в
                ; регистре R4 с младшим полусловом второго
                ; операнда в регистре R5. Запись результата в
                ; старшее полуслово регистра назначения R0
                ; Вычитание из младшего полуслова первого
                ; операнда в регистре R4 старшего полуслова
                ; второго операнда в регистре R5. Сохранение в
                ; младшем полуслове регистра назначения R0
USAX R7, R3, R2 ; Вычитание из старшего полуслова первого
                ; операнда в регистре R3 младшего полуслова
                ; второго операнда в регистре R2
                ; Сохранение результата в старшем полуслове
                ; регистра назначения R7
                ; Сложение младшего полуслова первого операнда
                ; в регистре R3 со старшим полусловом второго
                ; операнда в регистре R2.
                ; Сохранение результата в младшем полуслове
                ; регистра назначения
```

2.9 Параллельное перекрестное сложение и вычитание двух пар полуслов со знаком с делением результатов на два

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SHASX and SHSAX	Signed Halving Add and Subtract with Exchange – Знаковое «уполовинивание» суммы полуслов и разности полуслов с обменом (перекрестное) Signed Halving Subtract and Add with Exchange – Знаковое «уполовинивание» разности полуслов и суммы полуслов с обменом (перекрестное)		v	v

Синтаксис

```
op{cond}{Rd,} Rn, Rm
```

Где:

op оптокод один из:

SHASX - Signed Halving Add and Subtract with Exchange - Знаковое «уполовинивание» (деление на два) 16-разрядной суммы и разности полуслов с «перекрещиванием» операндов

SHSAX - Signed Halving Subtract and ADD with Exchange. Знаковое «уполовинивание» (деление на два)16-разрядной разности и суммы полуслов с «перекрещиванием» операндов

cond код условного выполнения команды (опция)

Rd – Регистр назначения.

Rn – Регистр, содержащий первый операнд

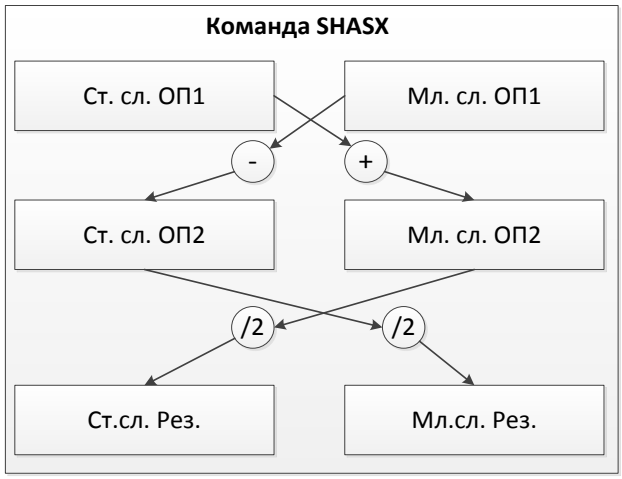
Rm – Регистр, содержащий второй операнд

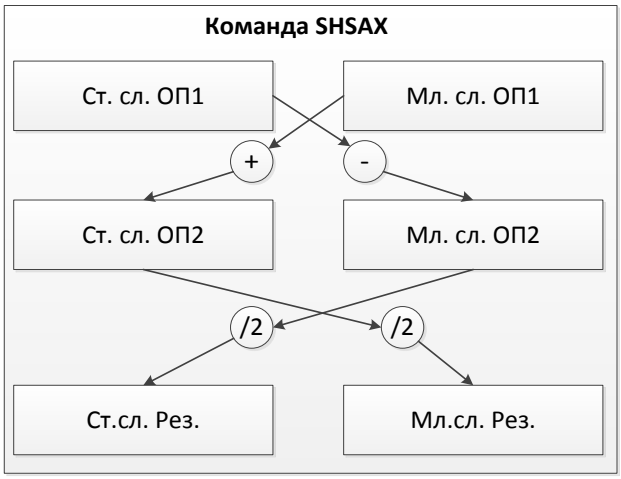
Операция

Команда **SHASX**:
1. Добавляет к старшему полуслову первого операнда младшее полуслово второго операнда.
2. Сдвигает результат арифметически на один бит вправо, обеспечивая знаковое деление на 2 и сохраняет «половинку» в старшем полуслове регистра назначения.
3. Вычитает из младшего полуслова первого операнда старшее полуслово второго операнда.
4. Сдвигает результат арифметически на один бит вправо, обеспечивая знаковое деление на 2 и сохраняет «половинку» в младшем полуслове регистра назначения.

Команда **SHSAX**:
1. Вычитает из старшего полуслова первого операнда младшее полуслово второго операнда.
2. Сдвигает результат арифметически на один бит вправо, обеспечивая знаковое деление на 2 и сохраняет «половинку» в старшем полуслове регистра назначения.
3. Складывает младшее полуслово первого операнда со старшим полусловом второго операнда.
4. Сдвигает результат арифметически на один бит вправо, обеспечивая знаковое деление на 2 и сохраняет «половинку» в младшем полуслове регистра назначения.

Графическая иллюстрация порядка выполнения операций в командах SHASX и SHSAX:





Внимание!
Сложение и вычитание выполняется с учетом расширения разрядной сетки 16-разрядного АЛУ на один разряд. Поэтому, при арифметическом сдвиге вправо старший разряд не теряется, а итоговый результат никогда не выходит за пределы 16-и или 8-разрядной сетки.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
SHASX R7, R4, R2        ; Добавляет к старшему полуслову в R4
                        ; младшее полуслово в R2 и записывает
                        ; «уполовиненный» результат в старшее полу-
                        ; слово регистра R7
                        ; Вычитает из младшего полуслова в регистре
                        ; R4 старшее полуслово в регистре R2 и
                        ; сохраняет «уполовиненный» результат в
                        ; младшем полуслове регистра R7
SHSAX R0, R3, R5        ; Вычитает из старшего полуслова в регистре R3
                        ; младшее полуслово в регистре R5 и записывает
                        ; «уполовиненный» результат в старшее полуслово
                        ; регистра R0
                        ; Добавляет к младшему полуслову в регистре R3
                        ; старшее полуслово в регистре R5 и сохраняет
                        ; «уполовиненный результат» в младшем полуслове
                        ; регистра R0
```

2.10 Параллельное перекрестное сложение и вычитание двух пар полуслов без знака с делением результатов на два

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
UHASX and UHSAX	Unsigned Halving Add and Subtract with Exchange – Без знаковое «уполовинивание» суммы полуслов и разности полуслов с обменом («перекрещиванием») операндов Unsigned Halving Subtract and Add with Exchange – Без знаковое «уполовинивание» разности полуслов и суммы полуслов с обменом («перекрещиванием») операндов		V	V

Синтаксис

```
op{cond}{Rd,} Rn, Rm
```

Где:		
	op оптокод один из: UHASX - Unsigned Halving Add and Subtract with Exchange – Без знаковое «уполовинивание» (деление на два) суммы и разности полуслов с «перекрещиванием» операндов UHSAX - Unsigned Halving Subtract and ADD with Exchange. Без знаковое «уполовинивание» (деление на два) разности и суммы полуслов с «перекрещиванием» операндов	cond код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий первый операнд	
	Rm – Регистр, содержащий второй операнд	

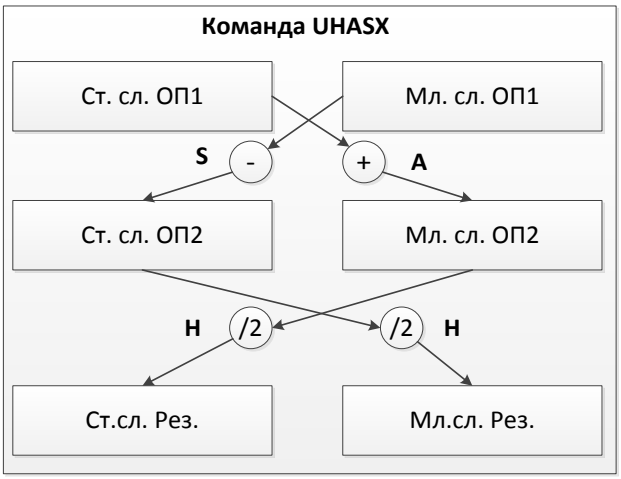
Операция

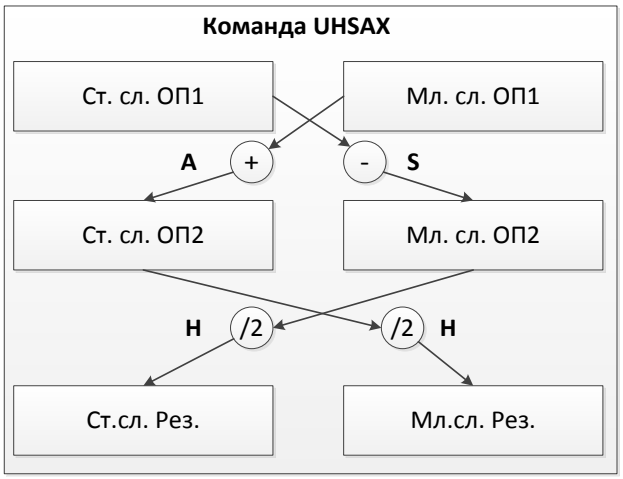
Команда UHASX:
1. Складывает числа без знака - старшее полуслово первого операнда с младшим полусловом второго операнда.
2. Сдвигает логически результат на один бит вправо, обеспечивая деление на 2 или «уполовинивание» результата, и сохраняет «половинку» в старшем полуслове регистра назначения.
3. Вычитает числа без знака - из младшего полуслова первого операнда старшее полуслово второго операнда.
4. Сдвигает логически результат на один бит вправо, обеспечивая деление на 2 или «уполовинивание» результата, и сохраняет «половинку» в младшем полуслове регистра назначения.

Команда UHSAX:
1. Вычитает числа без знака - из старшего полуслова первого операнда младшее полуслово второго операнда.

2. Сдвигает результат логически на один бит вправо, обеспечивая деление на 2 или «уполовинивание» результата, и сохраняет «половинку» в старшем полуслове регистра назначения.
3. Складывает числа без знака - младшее полуслово первого операнда со старшим полусловом второго операнда.
4. Сдвигает результат логически на один бит вправо, обеспечивая деление на 2 или «уполовинивание» результата, и сохраняет «половинку» в младшем полуслове регистра назначения.

Графическая иллюстрация алгоритма выполнения приведенных выше команд показана ниже:





Внимание!
Сложения и вычитания выполняются с учетом расширения разрядной сетки 16-разрядного АЛУ на один разряд. Поэтому, при сдвиге вправо старший разряд суммы или разности не теряется, а итоговый результат никогда не выходит за пределы 16-и или 8-разрядной сетки.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояние флагов

Примеры

```
UHASX R7, R4, R2 ; Добавляет к старшему полуслову в регистре
                 ; R4 младшее полуслово в регистре R2
                 ; Делит результат на 2 и сохраняет
                 ; в старшем полуслове регистра R7
                 ; Вычитает из младшего полуслова в регистре R4
                 ; старшее полуслово в регистре R2
                 ; Делит результат на 2 и сохраняет
                 ; в младшем полуслове регистра R7
UHSAX R0, R3, R5 ; Вычитает из старшего полуслова первого
                 ; операнда в R3 младшее полуслово второго
                 ; операнда в R5. Делит результат на 2 и
                 ; сохраняет в старшем полуслове регистра R0
                 ; Добавляет к младшему полуслову первого
                 ; операнда в регистре R3 старшее полуслово
                 ; второго операнда в регистре R5
                 ; Делит результат на 2 и сохраняет в младшем
                 ; полуслове регистра назначения R0
```

2.11 Селектирование байт или полуслов по состоянию флагов GE

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SEL	Select Bytes – Селектирование байт Selects each byte of its result from either its first operand or its second operand, according to the values of the GE flags. Выбор байта по значению флага GE («Больше или эквивалентно») либо из первого операнда, либо из второго. Используется для поддержки параллельных операций с байтами и полусловами.		V	V

Синтаксис

```
SEL{cond} {Rd,} Rn, Rm
```

Где:	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.
	Rn – Регистр первого операнда
	Rm – Регистр второго операнда

Операция

Команда **SEL**:
1. Считывает значения каждого флага GE[3:0] («Больше или эквивалентно») из регистра статуса приложения APSR, которые были установлены предыдущей операцией параллельного суммирования/вычитания полуслов/байт.
2. При установленном флаге GE для соответствующего байта/полуслова выбирает его из регистра первого операнда, в противном случае – из регистра второго операнда.
3. Выбранные байты/полуслова загружаются в соответствующие позиции регистра назначения.

При выполнении параллельных операций с байтами/полусловами в регистре статуса программы пользователя APSR вырабатываются дополнительные флаги результатов параллельных операций GE. Их значение полностью соответствует условию GE («Больше или эквивалентно»), вырабатываемому командами **ADDS** или **SUBS**, если бы они выполнялись в 8-и или 16-и разрядном АЛУ. Для каждой байтовой параллельной операции имеется отдельный флаг. Для параллельных операций с полусловами используются пары флагов, которые модифицируются одновременно:

При параллельных операциях с 4-мя байтами:			
Биты [31:24]	Биты [23:16]	Биты [15:8]	Биты [7:0]
GE[3]	**GE[2]**	**GE[1]**	**GE[0]**
При параллельных операциях с 2-мя полусловами:			
GE[3:2]		**GE[1:0]**	

Основное назначение команды SEL: После выполнения операции параллельного вычитания байт/полуслов и установки флагов GE[3:0] загрузить в регистр ЦПУ максимальные или минимальные значения байт/полуслов на основе сформированных значений флагов GE[3:0].

Ограничения

■ Нет

Выработка флагов

Эта команда не изменяет состояние флагов

Примеры

```
USUB8    r4, r1, r2  ; Параллельное вычитание 4-х пар байт
                     ; без знака (из байтов в регистре r1
                     ; байтов в регистре r2). Установка
                     ; флагов GE[3:0]
SEL      r4, r1, r2  ; Загрузка в регистр r4 максимальных
                     ; значений из каждой пары байт
SEL      r5, r2, r1  ; Загрузка в регистр r5 минимальных
                     ; значений из каждой пары байт
                     ; (Порядок регистров-источников изменен!)
```

2.12 Сложение абсолютных разностей 4-х пар байт без знака

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
USAD8	Unsigned Sum of Absolute Differences – Без знаковое сложение абсолютных значений разностей 4-х пар байт		V	V

Синтаксис

```
USAD8{cond}{Rd,} Rn, Rm
```

Где:	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.
	Rn – Регистр первого операнда
	Rm – Регистр второго операнда

Операция

Команда **USAD8**:
1. Вычитает из каждого байта первого операнда соответствующий байт второго операнда, получая четыре разности.
2. Суммирует абсолютные значения дифференциальных разностей вместе.
3. Записывает результат в регистр назначения.

Ограничения

■ Нельзя использовать указатель стека SP и счетчик команд PC

Выработка флагов

Эта команда не изменяет состояние флагов

Примеры

```
USAD8 R1, R4, R0      ; Вычитает из каждого байта первого операнда
                      ; R4 соответствующий байт второго операнда
                      ; R0, получая дифференциальные разности
                      ; Складывает все разности по модулю и
                      ; сохраняет результат в регистре R1
USAD8 R0, R5     ; Вычитает из каждого байта первого операнда
                 ; R0 соответствующие байты второго операнда в R5
                 ; Суммирует все разности по модулю и сохраняет
                 ; результат в регистре R0
```

2.13 Сложение абсолютных разностей 4-х пар байт без знака и накопление суммы

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
USADA8	Unsigned Sum of Absolute Differences and Accumulate – Без знаковое суммирование абсолютных значений разностей 4-х байт и накопление (аккумулирование)		V	V

Синтаксис

```
USADA8{cond}{Rd,} Rn, Rm, Ra
```

Где:	**cond** код условного выполнения команды (опция)
	Rd - Регистр назначения, может отсутствовать
	Rn - Регистр первого операнда
	Rm - Регистр второго операнда
	Ra - Регистр-аккумулятор, содержащий накопленное значение

Операция

Команда **USADA8**:
1. Вычитает из каждого байта первого операнда соответствующий байт второго операнда, получая четыре разности.
2. Суммирует абсолютные значения дифференциальных разностей вместе.
3. Добавляет к значению в аккумуляторе сумму абсолютных разностей.
4. Записывает результат в регистр назначения, если он специфицирован.

Ограничения

■ Нельзя использовать указатель стека SP и счетчик команд PC

Выработка флагов

Эта команда не изменяет состояние флагов

Примеры

```
        USADA8 R1, R0, R6       ; Вычитает из байтов первого операнда R1
                                ; соответствующие байты второго операнда R0
                                ; получая разности, складывает абсолютные
                                ; значения разностей и добавляет сумму к
                                ; текущему значению в аккумуляторе R6
                                ; сохраняет результат в регистре назначения
                                ; R1 (так как регистр назначения в команде
                                ; опущен, то его функции выполняет регистр
                                ; первого операнда)
        USADA8 R4, R0, R5, R2   ; Вычитает из байтов первого операнда
                                ; R0 байты второго операнда R5,
                                ; получая четыре разности
                                ; Суммирует разности по модулю и
                                ; добавляет к аккумулятору в регистре
                                ; R2. Сохраняет результат в регистре
                                ; назначения R4.
```

2.14 Подсчет числа лидирующих нулей

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
CLZ	Count Leading Zeros – Подсчет числа лидирующих нулей	v	v	v

Синтаксис

```
CLZ{cond} Rd, Rn
```

Где:	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.
	Rn – Регистр операнда

Операция

Команда **CLZ** подсчитывает число лидирующих нулей в операнде, расположенном в регистре Rn и возвращает результат в регистр назначения Rd.
Если ни один бит в регистре операнда не установлен, то возвращается число 32, если установлен самый старший бит bit[31], то возвращается ноль (0):

D31	D30	D29	D28	D27	D26	…	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x

В этом примере число лидирующих нулей 2.
Команда применяется в процедурах нормализации двоичных чисел.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эта команда не изменяет состояния флагов

Примеры

```
        CLZ         R4,R9       ; Подсчитать число лидирующих нулей в R9
                                ; Результат записать в R4
        CLZNE       R2,R3       ; Подсчитать число лидирующих нулей в R3
                                ; при выполнении условия NE («Не
                                ; эквивалентно»)
                                ; Результат записать в R2
```

3 ЛОГИЧЕСКИЕ КОМАНДЫ

Оглавление

3.1. Команды побитового Логического И, ИЛИ, Исключающего ИЛИ, И-НЕ, ИЛИ-НЕ.....48

3.2. Арифметический сдвиг вправо. Логический сдвиг влево и вправо. Циклический сдвиг вправо. Циклический сдвиг вправо через перенос (расширенный).....50

3.3. Пересылка данных. Пересылка с побитовым инвертированием. 52

3.4. Реверсирование порядка расположения байт и бит в слове.....54

3.5. Тестирование битов и проверка слов на эквивалентность.....56

3.1 Команды побитового Логического И, ИЛИ, Исключающего ИЛИ, И-НЕ, ИЛИ-НЕ

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
AND, ORR, EOR, BIC и ORN	Logical AND –Логическое И, OR – Логическое ИЛИ, Exclusive OR – Логическое Исключающее ИЛИ, Bit Clear – Очистка битов, Логическое И с НЕ OR NOT – Установка битов, Логическое ИЛИ с НЕ	v	v	v

Синтаксис

```
op{S}{cond} {Rd,} Rn, Operand2
```

Где:

op оптокод один из:

AND Logical AND – Логическое И
ORR Logical OR, or bit set – Логическое ИЛИ или Установка битов
EOR Logical Exclusive OR – Логическое Искл. ИЛИ
BIC Logical AND NOT , or bit clear – Логическое И-НЕ или очистка битов
ORN Logical OR NOT – Логическое ИЛИ-НЕ

cond код условного выполнения команды (опция)

S Опциональный суффикс (Set Flags). Если он специфицирован, то флаги кода условного выполнения обновляются по результату операции.

Rd – Регистр назначения. Если он не указан, то регистром назначения будет регистр-источник Rn.

Rn – Регистр-источник, содержащий первый операнд

Operand2 – Гибкий второй операнд: константа, регистр, регистр с опцией предварительного сдвига.

Операция

Команды AND, EOR и ORR выполняют операции побитового И, Исключающего ИЛИ и ИЛИ над первым операндом в Rn и вторым операндом Operand2.

Команда **AND** «Логическое И»– очищает битовые поля в тех разрядах, где в маске (второй операнд) нули (показана половина слова):

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
OP1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0P2	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
AND	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0

Команда **EOR** «Исключающее ИЛИ» инвертирует содержимое тех разрядов, где в маске (второй операнд) единицы (показана половина слова):

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
OP1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0P2	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
EOR	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0

Команда **ORR** «Логическое ИЛИ» устанавливает 1-е значение в тех разрядах, где в маске (второй операнд) единицы (показана половина слова):

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
OP1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0P2	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
OR R	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0

Команда **BIC** «Логическое И-НЕ» выполняет побитовую операцию И над битами первого операнда в Rn и инверсными битами второго операнда Operand2 – очистка тех разрядов, где в маске единицы (показана половина слова):

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
O1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
02	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
/O2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
BIC	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0

Команда **ORN** «Логическое ИЛИ-НЕ» выполняет побитовую операцию ИЛИ над битами первого операнда в Rn и инверсными битами второго операнда Operand2 – установка тех разрядов, где в маске нули (показана половина слова):

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
O1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
02	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
/O2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
BIC	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1

Ограничения

■ В команде нельзя использовать указатель стека SPи счетчик команд PC.

Выработка флагов

Если суффикс S специфицирован, то эти команды:

■ Обновляют флаги N и Z в соответствии с результатом операции.

■ Могут обновить флаг переноса C в процессе расчета операнда 2, например, если используется операция попутного сдвига.

■ Команды не оказывают влияния на флаг знакового переполнения V.

Примеры

```
AND   R9, R2, #0xFF00        ; Очистка младшего байта регистра R2
ORREQ R2, R0, R5             ; «Логическое ИЛИ» R0 и R5 при условии
                             ; EQ – «Эквивалентно»
ANDS R9, R8, #0x19           ; «Логическое И» R8 и 0x19 с
                             ; установкой флагов
EORS R7, R11, #0x18181818    ; «Исключающее ИЛИ» R11 и 0x18181818
                             ; с установкой флагов
BIC   R0, R1, #0xab          ; «Логическое И-НЕ» R1 и 0xab
ORN   R7, R11, R14, ROR #4   ; Маска в R14 циклически сдвигается
                             ; на 4 разряда вправо перед операцией
                             ; «Логического ИЛИ-НЕ»
ORNS R7, R11, R14, ASR #32   ; Маска сдвигается арифметически на 32
                             ; разряда вправо – знак операнда в R14
                             ; расширяется до полного слова, затем
                             ; выполняется операция «Логического
                             ; ИЛИ-НЕ»
                             ; Фактически: если (R14) положительно,
                             ; то все биты R11 устанавливаются
                             ; и сохраняются в R7
```

3.2 Арифметический сдвиг вправо. Логический сдвиг влево и вправо. Циклический сдвиг вправо. Циклический сдвиг вправо через перенос (расширенный)

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
ASR, LSL, LSR, ROR и RRX	Arithmetic Shift Right – Арифметический сдвиг вправо Logical Shift Left – Логический сдвиг влево Logical Shift Right – Логически сдвиг вправо Rotate Right – Циклический сдвиг вправо Rotate Right with Extend – Циклический сдвиг вправо с расширением	v	v	v

Синтаксис

```
op{S}{cond} Rd, Rm, Rs
op{S}{cond} Rd, Rm, #n
RRX{S}{cond} Rd, Rm
```

Где:

op оптокод один из:

ASR Arithmetic Shift Right – Арифметический сдвиг вправо
LSL Logical Shift Left – Логический сдвиг влево
LSR Logical Shift Right- Логический сдвиг вправо
ROR Rotate Right – Циклический сдвиг вправо
RRX Расширенный циклический сдвиг вправо на 1 разряд

cond код условного выполнения команды (опция)

S Опциональный суффикс (Set Flags). Если он специфицирован, то флаги кода условного выполнения обновляются по результату операции.

Rd – Регистр назначения. Если он не указан, то регистром назначения будет регистр-источник Rm.

Rm – Регистр, содержащий значение, которое будет сдвигаться

Rs – Регистр, содержащий величину сдвига, которая должна быть применена к содержимому регистра-операнда Rm. Используется только последний значащий байт, значение которого может быть от 0 до 255

n – величина сдвига в числе бит. Величина сдвига зависит от типа команды:

ASR от 1 до 32
LSL от 1 до 31
LSR от 1 до 32
ROR от 1 до 31

Замечание:
Для команды сдвига на 0 разрядов LSL{S}{cond} Rd, Rm, #0 используется более предпочтительный синтаксис MOV{S}{cond} Rd, Rm

Операция

Команды **ASR**, **LSL**, **LSR** и **ROR** перемещают биты в регистре Rm влево или вправо на число разрядов, заданное константой n или содержимым регистра Rs.

Команда расширенного циклического сдвига RRX рассматривает регистр-источник Rm и флаг переноса C как один 33-разрядный регистр, выполняя циклический сдвиг вправо на один разряд. Во всех командах результат записывается в регистр назначения Rd, а исходное содержимое регистра Rm (исходного операнда) остается неизменным.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Если суффикс S специфицирован, то эти команды:

■ Обновляют флаги N и Z в соответствии с результатом операции.

■ Флаг переноса C обновляется последним вытесненным за пределы 32-разрядной сетки битом, за исключением случая, когда выполняется логический сдвиг влево на 0 разрядов.

Примеры

```
ASR R7, R8, #9  ; Арифметический сдвиг вправо на 9 бит числа в R8
                ; и запись результата в R7
LSLS R1, R2, #3 ; Логический сдвиг влево на 3 разряда содержимого
                ; R2 с записью результата в R1
                ; Обновление флагов по результату операции
LSR R4, R5, #6  ; Логический сдвиг вправо на 6 бит содержимого R5
                ; с записью результата в R4
ROR R4, R5, R6  ; Циклический сдвиг вправо содержимого R5
                ; на число разрядов в младшем байте регистра R6
                ; с записью результата в R4
RRX R4, R5      ; Циклический сдвиг вправо на разряд расширенный
                ; (R5, флаг C) – один 33-разрядный регистр
```

3.3 Пересылка данных. Пересылка с побитовым инвертированием.

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MOV и **MVN**	Move and Move NOT - Пересылка и пересылка с побитовом инвертированием	v	v	v

Синтаксис

```
MOV{S}{cond} Rd, Operand2
MOV{cond} Rd, #imm16
MVN{S}{cond} Rd, Operand2
```

Где:	**MOV** Пересылка данных **MVN** Пересылка побитово-проинвертированных данных	**cond** код условного выполнения команды (опция)
	S Опциональный суффикс (Set Flags). Если он специфицирован, то флаги кода условного выполнения обновляются по результату операции.	
	Rd – Регистр назначения.	
	Operand2 – Гибкий второй операнд: константа, регистр ЦПУ, регистр ЦПУ с опцией предварительного сдвига.	
	imm16 – 16-разрядный непосредственный операнд в диапазоне 0-65535	

Операция

Команда пересылки **MOV** копирует данные из операнда 2 Operand2 в регистр назначения Rd. Если операнд 2 в команде пересылки является регистром с логическим сдвигом влево на число разрядов, отличное от 0 (LSL #0), то предпочтительным синтаксисом является соответствующая ей команда сдвига:

Команда пересылки	Предпочтительный синтаксис с использованием команды сдвига
MOV{S}{cond} Rd, Rm, ASR #n	ASR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, LSL #n (if n != 0)	LSL{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, LSR #n	LSR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, ROR #n	ROR{S}{cond} Rd, Rm, #n
MOV{S}{cond} Rd, Rm, RRX	RRX{S}{cond} Rd, Rm

Таким образом, команда пересылки с опцией сдвига второго операнда выполняет ту же самую функцию, что и соответствующая команда сдвига. В этом случае рекомендуется вместо команды пересылки использовать соответствующую команду сдвига ASR, LSL, LSR, ROR или RRX.
Если величина сдвига второго операнда задается содержимым регистра Rs, то вместо соответствующей команды сдвига можно использовать команду пересылки в следующем формате:

Команда сдвига	Синоним – Команда пересылки
ASR{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, ASR Rs
LSL{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, LSL Rs
ROR{S}{cond} Rd, Rm, Rs	MOV{S}{cond} Rd, Rm, ROR Rs

Команда **MVN** (пересылка с предварительной побитовой инверсией) вначале выполняет побитовое логическое НЕ содержимого второго операнда, а затем помещает результат в регистр назначения Rd.
Замечание:
Команда MOVW выполняет те же самые функции, что и команда MOV, однако ее использование ограничено: допустим только 16-разрядный непосредственный операнд imm16.

Ограничения

Вы можете использовать указатель стека SP и счетчик команд PC в командах пересылки данных при следующих ограничениях:

- Второй операнд должен быть регистром без сдвига
- Вы не должны специфицировать в команде суффикс установки флагов S.

Если счетчик команд PC является регистром-приемником в команде MOV, то:

- Bit[0] в значении, которое записывается в PC, игнорируется
- Переход происходит по адресу, в котором нулевой бит принудительно сброшен в 0: bit[0]=0.

Замечание:
Хотя Вы можете использовать команду MOV в качестве команды передачи управления (branch instruction) по косвенному адресу, все же настоятельно рекомендуется, по возможности, использовать команды перехода BX или BLX для обеспечения гарантии переместимости программного обеспечения для ARM-процессоров с системой команд Cortex-M3/M4F.

Выработка флагов

Если опция установки флагов S специфицирована:

- Флаги N и Z обновляются в соответствии с результатом операции.
- Флаг C может обновиться в соответствии с выполненной операцией сдвига второго операнда.
- Команда не влияет на состояние флага переполнения V.

Таким образом, команда MOV может модифицировать флаги, если используется с опцией установки флагов S.

Примеры

```
MOVS R11, #0x000B     ; Записать число 0x000B в регистр R11,
                      ; Установить флаги.
MOV R1, #0xFA05       ; Записать число 0xFA05 в регистр R1,
                      ; Флаги не обновлять.
MOVS R10, R12         ; Переслать данные из регистра R12 в R10,
                      ; Установить флаги.
MOV R3, #23           ; Записать константу 23 в регистр R3.
MOV R8, SP            ; Записать содержимое указателя стека в R8.
MVNS R2, #0xF         ; Записать побитово-инверсное значение
                      ; непосредственного операнда 0xF, а именно
                      ; число 0xFFFFFFF0 в регистр R2
                      ; Установить флаги
```

3.4 Реверсирование порядка расположения байт и бит в слове

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
REV , REV16, REVSH, and RBIT	Reverse bytes and Reverse bits – Реверсирование байт и реверсирование бит	v	v	v

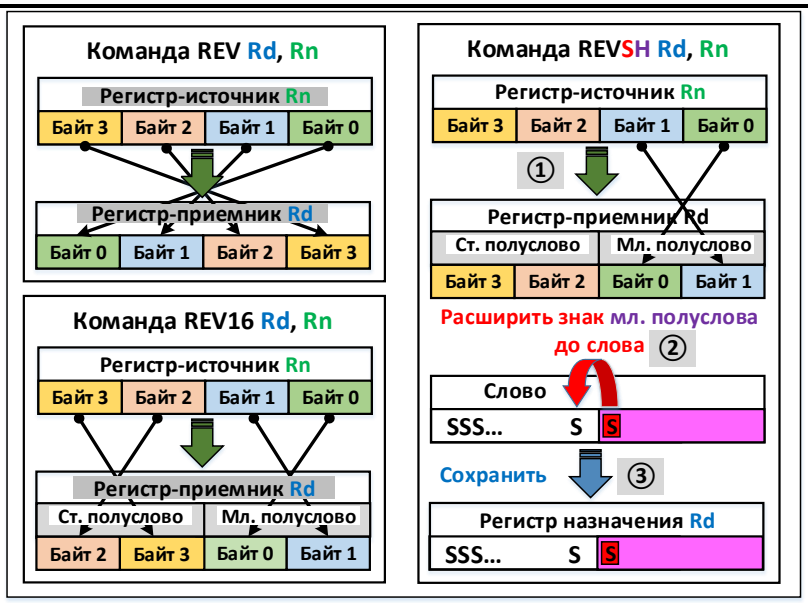
Синтаксис

```
op{cond} Rd, Rn
```

Где:	**op** оптокод один из: **REV** - Reverse byte order in a word – Реверсирование порядка байт в слове **REV16** - Reverse byte order in each halfword independently – Реверсирование порядка байт в каждом полуслове независимо **REVSH** - Reverse byte order in the bottom halfword, and sign extend to 32 bits – Реверсирование порядка байт в младшем полуслове и знаковое расширение до 32 бит. **RBIT** - Reverse the bit order in a 32-bit word – Реверсирование порядка бит в 32-разрядном слове	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn – Регистр, содержащий исходный операнд	

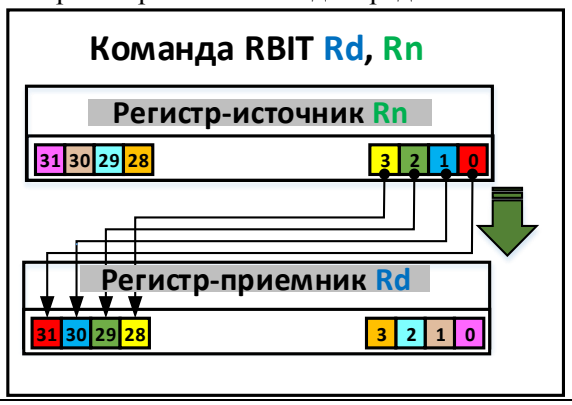
Операция

Эти команды изменяют порядок размещения данных в слове:
REV – Реверсирует порядок расположения байт в слове: преобразует 32- разрядное число с обратным порядком расположения байт (32-bit big-endian data) в 32-разрядное число с прямым порядком расположения байт (32-bit little-endian data) и обратно.
REV16 – Реверсирует порядок расположения байт в каждом полуслове независимо.
REVSH – Реверсирует порядок байт в младшем полуслове и расширяет результат, как число со знаком в дополнительном коде (полуслово) до 32-разрядного слова
Графическая иллюстрация алгоритма работы этих команд представлена ниже:



RBIT – Реверсирует порядок расположения бит в слове

Графическая иллюстрация алгоритма работы команды представлена ниже:



Ограничения

■ Нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Флаги не вырабатываются.

Примеры

```
REV R3, R7        ; Реверсирование порядка байт в R7 и
                  ; запись результата в R3
REV16 R0, R0      ; Реверсирование порядка байт в каждом полуслове
                  ; регистра R0 и запись туда же
REVSH R0, R5      ; Reverse Signed Halfword - Реверсирование байт в
                  ; младшем полуслове и знаковое расширение до
                  ; слова
REVHS R3, R7      ; Реверсирование байт  в слове по условию
                  ; Higher or Same (Выше или то же самое).
RBIT R7, R8       ; Реверсирование бит в регистре R8 и запись в R7
```

3.5 Тестирование битов и проверка слов на эквивалентность

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
TST and **TEQ**	Test bits – Тестирование битов Test Equivalence – Тестирование на эквивалентность (переменных)	v	v	v

Синтаксис

```
op{cond} Rn, Operand2
```

Где:	**op** оптокод один из: **TST** - Test bits - Тестирование бит **TEQ** - Test Equivalence - Тестирование на эквивалентность двух переменных-слов	**cond** код условного выполнения команды (опция)
	Rn - Регистр, содержащий первый операнд	
	Operand2 - Гибкий второй операнд	

Операция

Эти команды тестируют значение в регистре Rn в сравнении с значением операнда 2. Они обновляют флаги в соответствии с операцией тестирования, но не записывают куда-либо результат тестирования (аналогично командам сравнения CMP).

Команда **TST** выполняет побитовое «Логическое И» значения в регистре Rn и значения во втором операнде Operand2. Она выполняет те же самые действия, что и команда **ANDS**, за исключением того, что результат операции отбрасывается, - выставляются только флаги.
Для того, чтобы протестировать какой-либо бит в регистре Rn (на 0 или 1), нужно выполнить команду TST со вторым операндом-константой (маской), которая будет иметь этот бит, установленный в 1 и остальные биты, - сброшенные в 0. Единичному значению бита будет соответствовать код условия NE («Не эквивалентно»), а нулевому – EQ («Эквивалентно»).

Команда **TEQ** (тестирование на эквивалентность) выполняет операцию «Логическое исключающее ИЛИ» (Exclusive OR) над значением в регистре Rn и значением второго операнда Operand2. Это точно такая же операция, как **EORS**, за исключением того, что результат операции отбрасывается, - выставляются только флаги.
Используйте команду TEQ для проверки, что <u>две переменные эквивалентны друг другу</u>. Если все биты переменных будут одинаковы, то установится флаг нуля Z. При этом можно не анализировать флаги V и C.
Используйте также команду TEQ <u>для тестирования знаков переменных</u>. Значение флага N будет «Логическим Исключающим ИЛИ» знаковых битов этих переменных. Если результат тестирования положительный, то знаки операндов совпадают. Если – отрицательный, то знаки операндов отличаются:

OP1	OP2	Exclusive OR	Результат сравнения операндов
+ (S=0)	+ (S=0)	S=0 → N=0	Одинаковые по знаку
- (S=1)	+ (S=0)	S=1 → N=1	Разные по знаку
- (S=1)	- (S=1)	S=0 → N=0	Одинаковые по знаку
+ (S=0)	- (S=1)	S=1 → N=1	Разные по знаку

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

■ Флаг отрицательного результата N и флаг нуля Z обновляются в соответствии с результатом операции.
■ Флаг C может обновляться при выполнении операций сдвига во втором гибком операнде.
■ Состояние флага V не меняется.

Примеры

```
TST R0, #0x3F8   ; Выполняет «Побитовое И» содержимого R0 с маской
                 ; 0x3F8. Обновляются только флаги в регистре
                 ; статуса приложения APSR.
TEQEQ R10, R9    ; Команда условной проверки (по условию EQ)
                 ; на эквивалентность содержимого регистров R10,
                 ; и R9
                 ; Только если все биты совпадают, выставляется
                 ; флаг Z.
```

4 КОМАНДЫ УМНОЖЕНИЯ, УМНОЖЕНИЯ С НАКОПЛЕНИЕМ, ДЕЛЕНИЯ

Оглавление

4.1. Умножение слов, умножение слов с положительным и отрицательным аккумулированием с генерацией младшего слова произведения 59

4.2. Деление слов со знаком и без знака с получением 32-разрядного частного 61

4.3. Знаковое умножение полуслов с получением 32-разрядного произведения. Знаковое умножение слова на полуслово с получением старшего слова 48-разрядного произведения 62

4.4. Знаковое двойное умножение полуслов, сложение или вычитание 32-разрядных произведений 65

4.5. Знаковое умножение полуслов с получением 32-разрядного произведения и аккумулированием. Знаковое умножение слова на полуслово с получением старшего слова 48-разрядного произведения и аккумулированием. 68

4.6. Двойное знаковое умножение полуслов с аккумулированием обоих 32-разрядных произведений 72

4.7. Длинное умножение слов со знаком и без знака с получением 64-разрядного произведения, длинное знаковое и без знаковое умножение с накоплением 64-разрядного произведения, длинное без знаковое умножение с двойным аккумулированием. 74

4.8. Знаковое умножение полуслов с длинным аккумулированием. Двойное знаковое умножение полуслов с длинным аккумулированием 77

4.9. Двойное знаковое умножение полуслов, вычитание произведений и накопление разности в 32-разрядном или 64-разрядном аккумуляторе 81

4.10. Умножение слов со знаком с получением старшего слова произведения 85

4.11. Знаковое умножение слов с положительным или отрицательным аккумулированием старшего значащего слова произведения 87

4.1 Умножение слов, умножение слов с положительным и отрицательным аккумулированием с генерацией младшего слова произведения

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MUL, MLA, and MLS	Multiply, Multiply with Accumulate, and Multiply with Subtract, using 32-bit operands, and producing a 32-bit result. Умножение, Умножение с аккумулированием и Умножение с вычитанием (отрицательным аккумулированием), Использование 32-разрядных операндов и генерация 32-разрядного результата (младшего слова результата).	v	v	v

Синтаксис

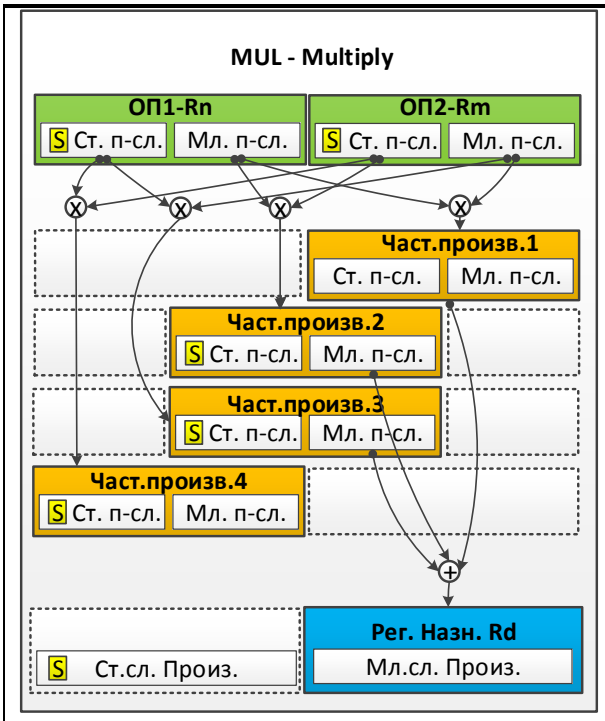
```
MUL{cond} {Rd,} Rn, Rm      ; Multiply
                            ; Умножение 32-разрядных слов
MLA{cond} Rd, Rn, Rm, Ra    ; Multiply with accumulate
                            ; Умножение слов с аккумулированием
MLS{cond} Rd, Rn, Rm, Ra    ; Multiply with subtract
                            ; Умножение слов с вычитанием (отрицательным
                            ; аккумулированием)
```

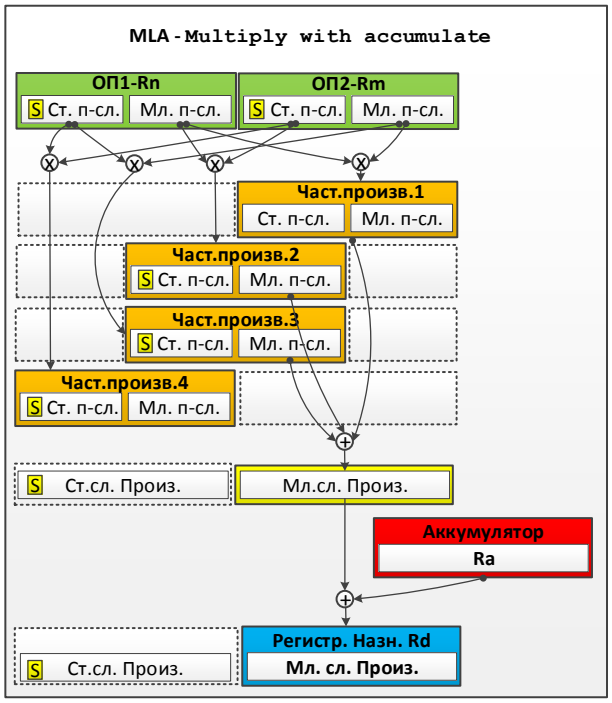
Где:

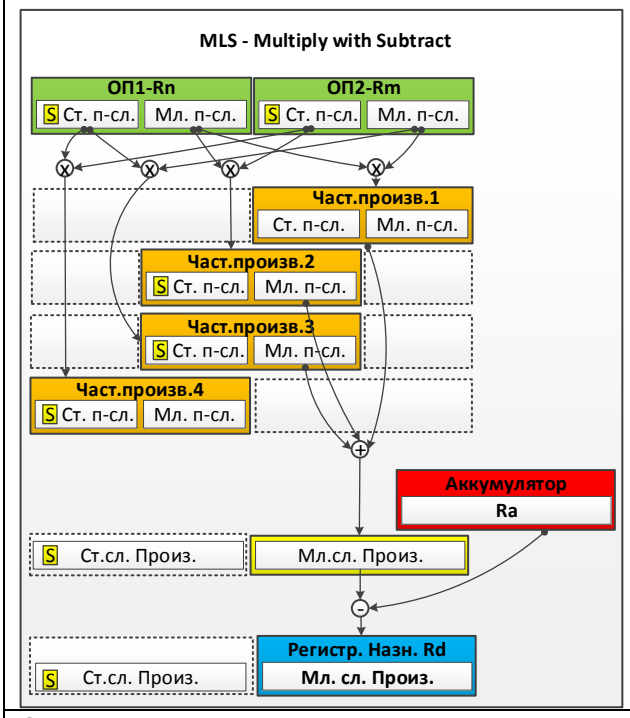
	cond код условного выполнения команды (опция)
	Rd – Регистр назначения. Если регистр назначения опущен, то регистром назначения будет регистр первого операнда **Rn**.
	Rn, Rm – Регистры, содержащие первый и второй операнды – слова, которые должны умножаться (множители).
	Ra – Регистр аккумулятор, к которому добавляется результат умножения или из которого вычитается результат умножения.

Операция

MUL - Умножает значения в регистрах множителей Rn и Rm и сохраняет младшее значащее 32-разрядное слово 64-разрядного произведения в регистре результата Rd.
MLA - Умножает значения в регистрах множителей Rn и Rm и добавляет младшее значащее 32-разрядное слово результата умножения к аккумулятору в регистре Ra, сохраняет полученное накопленное значение в регистре назначения Rd.
MLS - Умножает значения в регистрах множителей Rn и Rm и вычитает младшее значащее 32-разрядное слово результата умножения из значения в аккумуляторе Ra, сохраняет полученное значение в регистре назначения Rd.
Все операции выполняются над числами со знаком в дополнительном коде. Если результат выходит за пределы 32-х разрядов, то старшее слово результата отбрасывается (результат будет недостоверным).
Графическая иллюстрация алгоритма выполнения команд представлена ниже. Обратите внимание на то, что вычисляется только младшее 32-разрядное слово произведения. Старшее 32-разрядное слово произведения не вычисляется.







Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.
Если Вы используете суффикс S вместе с командой умножения MUL (хотите установить флаги по результату операции), то:
■ Регистры операндов (Rn и Rm) и регистр назначения Rd должны быть в диапазоне от R0 до R7.
■ Регистр назначения Rd должен быть тем же самым, что и регистр второго операнда Rm.
■ Вы не должны дополнительно использовать суффикс условного выполнения команды cond.

Выработка флагов

■ Команда не меняет значения флагов C и V. Узнать, выходит ли произведение за 32-разрядный диапазон невозможно! Проявляйте осторожность, так как возможные переполнения не контролируются!

Примеры

```
MUL R10, R2, R5        ; Умножение, R10 = (R2) x (R5). Сохраняются
                       ; только младшие 32 разряда произведения.
MLA R10, R2, R1, R5    ; Умножение с аккумулированием
                       ; R10 = (R5) +(R2 x R1).
MULLT R2, R3, R2       ; Условное умножение по LT (строго меньше)
                       ; R2 = (R3) x (R2).
MLS R4, R5, R6, R7     ; Умножение с вычитанием (отрицательным
                       ; накоплением), R4 = (R7) - (R5) x (R6).
```

4.2 Деление слов со знаком и без знака с получением 32-разрядного частного

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SDIV UDIV	Signed Divide Знаковое деление Unsigned Divide Без знаковое деление	v	v	v

Синтаксис

```
SDIV{cond} {Rd,} Rn, Rm
UDIV{cond} {Rd,} Rn, Rm
```

Где:		**cond** код условного выполнения команды (опция)
	Rd –Регистр назначения. Если он отсутствует, то регистром назначения будет регистр Rn.	
	Rn – Регистр, содержащий делимое	
	Rm – Регистр, содержащий делитель	

Операция

Команда **SDIV** выполняет знаковое деление значения в регистре Rn на значение в регистре Rm. Команда **UDIV** выполняет без знаковое деление значения в регистре Rn на значение в регистре Rm. И в том и в другом случае делить можно только целые числа. Эти команды не работают с вещественными числами в формате с фиксированной точкой.

Для обеих команд, если значение в Rn не делится на значение Rm, результат округляется к нулю.
Время выполнения команд от 2 до 12 циклов и зависит от значений делимого и делителя.
Команды являются прерываемыми. При поступлении запроса прерывания, выполнение команды приостанавливается и автоматически продолжается после обслуживания прерывания.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
SDIV R0, R2, R4        ; Знаковое деление, R0 = R2/R4.
UDIV R8, R8, R1        ; Беззнаковое деление, R8 = R8/R1.
```

4.3 Знаковое умножение полуслов с получением 32-разрядного произведения. Знаковое умножение слова на полуслово с получением старшего слова 48-разрядного произведения

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMUL{X}{Y} SMULW{Y}	Signed Multiply (halfwords) Знаковое умножение полуслов Signed Multiply (word by halfword) Знаковое умножение слова (32 разряда) на полуслово		V	V

Синтаксис

```
SMUL{X}{Y}{cond} Rd,Rn, Rm ; Знаковое умножение полуслов
SMULW{Y}{cond} Rd, Rn, Rm  ; Знаковое умножение слова на полуслово
```

Где:

op:

SMUL{X}{Y}	Signed Multiply (halfwords) Знаковое умножение полуслов. **{X}{Y}** – Суффиксы, которые специфицируют полуслова в первом Rn и втором Rm регистрах-операндах, участвующие в операции умножения. Вначале специфицируется полуслово первого операнда**{X}**, а затем – полуслово второго операнда **{Y}**. Возможные значения X и Y: **B** – Bottom halfword Младшее полуслово регистра-операнда, биты [15:0] **T** – Top halfword Старшее полуслово регистра-операнда, биты [31:16]
SMULW{Y}	Signed Multiply (word by halfword) Знаковое умножение слова первого операнда Rn на полуслово второго операнда Rm, заданное суффиксом **{Y}**. Возможные значения Y: **B** – Bottom halfword Младшее полуслово второго регистра-операнда Rm, биты [15:0] **T** – Top halfword Старшее полуслово второго регистра-операнда Rm, биты [31:16]

cond
код условного выполнения команды (опция)

Rd – Регистр назначения.

Rn, Rm – Регистры первого и второго операндов-источников

Операция

Команды SMULBB, SMULTB, SMULBT и SMULTT интерпретируют значения в регистрах первого и второго операндов Rn и Rm как четыре 16-разрядных полуслова со знаком в дополнительном коде:

■ Выполняют знаковое умножение специфицированных полуслов первого и второго регистров-операндов:

BB – bottom * bottom (младшего полуслова на младшее)

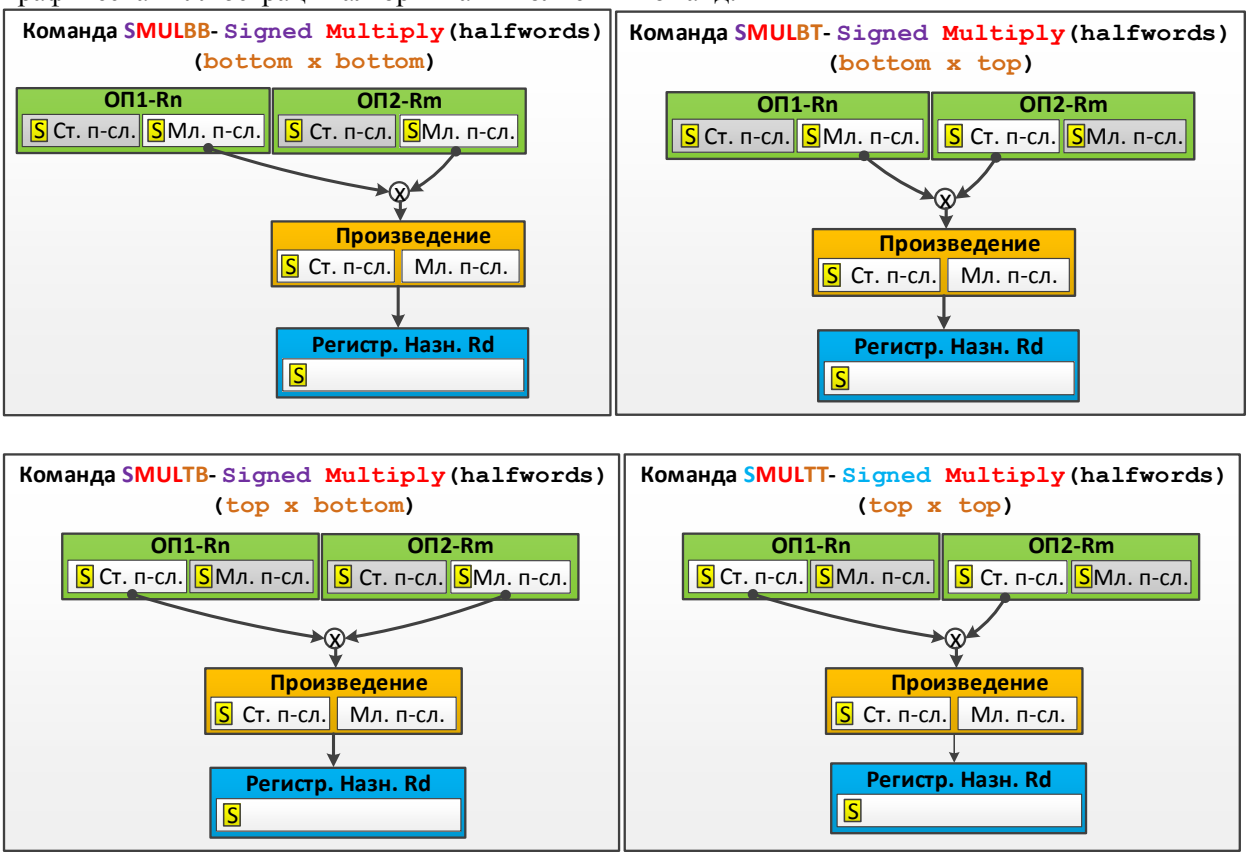
TB – top * bottom (старшего полуслова на младшее)

BT – bottom * top (младшего полуслова на старшее)

TT – top * top (старшего полуслова на старшее)

■ Записывают 32-разрядный результат умножения в регистр-назначения Rd.

Графическая иллюстрация алгоритма выполнения команд:

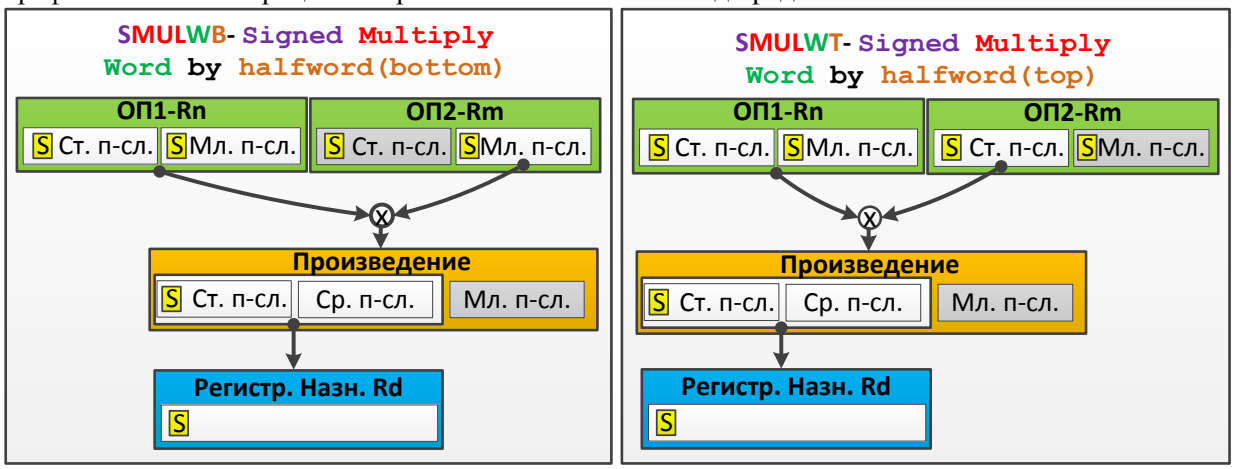


Команды SMULWT и SMULWB интерпретируют значение в первом регистре-операнде Rn как 32-разрядное число со знаком в дополнительном коде (слово), а значение во втором регистре-операнде Rm – как два 16-разрядных числа со знаком в дополнительном коде (два полуслова):

■ Выполняют умножение 32-разрядного слова первого операнда на специфицированное суффиксом Y 16-разрядное полуслово второго операнда с получением 48-разрядного результата.

■ Записывают старшие значащие 32 бита произведения в регистр-назначения Rd. Младшее полуслово произведения теряется.

Графическая иллюстрация алгоритма выполнения команд представлена ниже:



Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
SMULBT R0, R4, R5     ; Умножение младшего полуслова первого
                      ; регистра-операнда R4 на старшее полуслово
                      ; второго регистра-операнда R5 (знаковое)
                      ; Сохранение произведения в регистре-
                      ; назначения R0
SMULBB R0, R4, R5     ; Умножение младшего полуслова первого
                      ; регистра-операнда R4 на младшее полуслово
                      ; второго регистра-операнда R5(знаковое)
                      ; Сохранение произведения в регистре-
                      ; назначения R0
SMULTT R0, R4, R5     ; Умножение старшего полуслова первого
                      ; регистра-операнда R4 на старшее полуслово
                      ; второго регистра-операнда R5(знаковое)
                      ; Сохранение произведения в регистре-
                      ; назначения R0
SMULTB R0, R4, R5     ; Умножение старшего полуслова первого
                      ; регистра-операнда R4 на младшее полуслово
                      ; второго регистра-операнда R5(знаковое)
                      ; Сохранение произведения в регистре-
                      ; назначения R0
SMULWT R4, R5, R3     ; Знаковое умножение 32-разрядного слова
                      ; в первом регистре-операнде R5 на
                      ; старшее 16-разрядное полуслово во втором
                      ; регистре-операнде R3
                      ; Выделение старших 32-бит произведения и
                      ; запись в регистр назначения R4
SMULWB R4, R5, R3     ; Знаковое умножение 32-разрядного слова
                      ; в первом регистре-операнде R5 на
                      ; младшее 16-разрядное полуслово во втором
                      ; регистре-операнде R3
                      ; Выделение старших 32-бит произведения и
                      ; запись в регистр назначения R4
```

4.4 Знаковое двойное умножение полуслов, сложение или вычитание 32-разрядных произведений

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMUAD	Signed Dual Multiply Add Знаковое двойное умножение полуслов, сложение произведений		v	v
SMUSD	Signed Dual Multiply Subtract Знаковое двойное умножение полуслов, вычитание произведений			

Синтаксис

op{X}{cond} Rd, Rn, Rm

Где:

op:

SMUAD	Signed Dual Multiply Add Знаковое двойное умножение полуслов, суммирование произведений
SMUSD	Signed Dual Multiply Subtract Знаковое двойное умножение полуслов, вычитание произведений

X - суффикс обмена местами полуслов второго операнда (перекрещенного умножения)- e**X**change

X отсутствует	Младшее 1-го на младшее 2-го (bottom x bottom) Старшее 1-го на старшее 2-го (top x top)
X присутсвует	Младшее 1-го на старшее 2-го (bottom x top) Старшее 1-го на младшее 2-го (top x bottom)

cond
код условного выполнения команды (опция)

Rd – Регистр назначения.

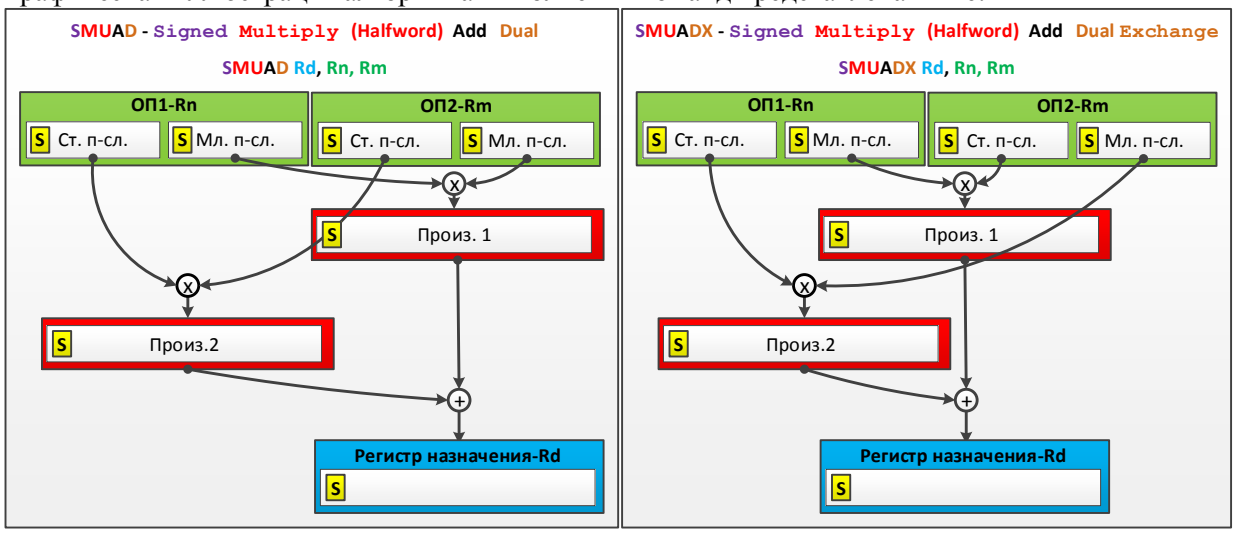
Rn, Rm – Регистры, содержащие первый и второй операнды с полусловами - множителями

Операция

Команда **SMUAD** интерпретирует значения в первом и втором регистрах-операндах как два знаковых полуслова в дополнительном коде (16-разрядных):

- Если указана опция обмена полуслов «X», то меняет местами полуслова второго операнда.
- Выполняет два знаковых умножения соответствующих полуслов первого и второго операндов (младшего на младшее), (старшего на старшее).
- Складывает вместе два 32-разрядных произведения.
- Сохраняет результат сложения в регистре-назначения.

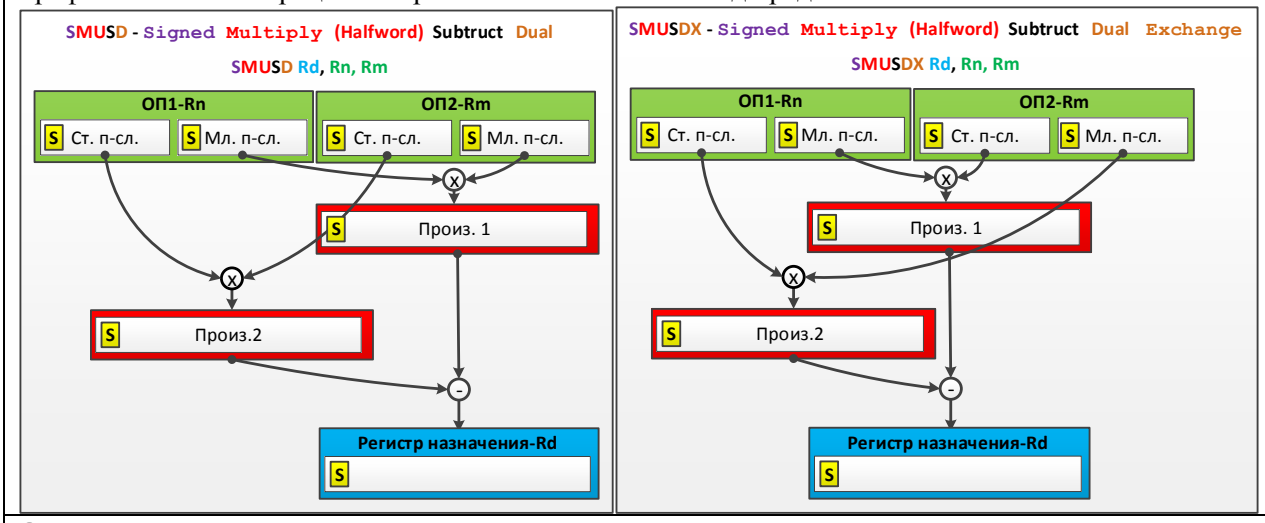
Графическая иллюстрация алгоритма выполнения команд представлена ниже:



Команда **SMUSD** также интерпретирует значения в первом и втором регистрах-операндах как два знаковых полуслова в дополнительном коде (16-разрядных):

■ Если указана опция обмена полуслов «X», то меняет местами полуслова второго операнда.
■ Выполняет два знаковых умножения соответствующих полуслов первого и второго операндов (младшего на младшее), (старшего на старшее).
■ Вычитает из первого произведения второе.
■ Сохраняет результат вычитания в регистре-назначения.

Графическая иллюстрация алгоритма выполнения команд представлена ниже:



Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
        SMUAD R0, R4, R5        ; Двойное умножение знаковых полуслов
                                ; исходных операндов в регистрах R4 и R5
                                ; младшего на младшее (bottom x bottom)
                                ; старшего на старшее (top x top)
                                ; Сложение двух 32-разрядных произведений
                                ; Сохранение результата в регистре-
                                ; назначения R0
```

П1.4 КОМАНДЫ УМНОЖЕНИЯ С НАКОПЛЕНИЕМ, ДЕЛЕНИЯ

```
SMUADX R3, R7, R4      ; Двойное умножение знаковых полуслов
                       ; исходных операндов в регистрах R7 и R4
                       ; перекрестное
                       ; младшего R7 на старшее R4 (bottom x top)
                       ; старшего R7 на младшее R4 (top x bottom)
                       ; Сложение двух 32-разрядных произведений
                       ; Сохранение результата в регистре-
                       ; назначения R3

SMUSD R3, R6, R2       ; Двойное умножение знаковых полуслов
                       ; исходных операндов в регистрах R6 и R2
                       ; младшего на младшее (bottom x bottom)
                       ; старшего на старшее (top x top)
                       ; Вычитание из первого произведения
                       ; второго
                       ; Сохранение результата в регистре-
                       ; назначения R3


SMUSDX R4, R5, R3      ; Двойное умножение знаковых полуслов
                       ; исходных операндов в регистрах R5 и R3
                       ; перекрестное
                       ; младшего R5 на старшее R3 (bottom x top)
                       ; старшего R5 на младшее R3 (top x bottom)
                       ; Вычитание из первого произведения
                       ; второго
                       ; Сохранение результата в регистре-
                       ; назначения R4
```

4.5 Знаковое умножение полуслов с получением 32-разрядного произведения и аккумулированием. Знаковое умножение слова на полуслово с получением старшего слова 48-разрядного произведения и аккумулированием

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMLA	Signed Multiply (halfwords) Accumulate Знаковое умножение (полуслов) с аккумулированием		v	v
SMLAW	Signed Multiply (word by halfword) Accumulate Знаковое умножение слова на полуслово с аккумулированием старшего слова			

Синтаксис

op{Y}{cond} {Rd}, Rn, Rm, Ra

Где:

op одна из операций:

SMLAXY	Signed Multiply(halfwords) Accumulate. Знаковое умножение (полуслов) с аккумулированием. Символами X и Y специфицируется, какие полуслова из регистров первого и второго операндов Rn и Rm. Используются в качестве 16-разрядных множителей: B - bottom halfword – младшее полуслово, биты [15:0]; T - top halfword – старшее полуслово, биты [31:16]. Символ X задает полуслово первого операнда, следующий за ним символ Y, – полуслово второго операнда. Если указан только один из двух возможных символов, то он специфицирует полуслова, как в первом, так и во втором операндах-словах.
SMLAWY	Signed Multiply Accumulate (word by halfword) Знаковое умножение (слова на полуслово) и аккумулирование. Слово, участвующее в умножении берется из первого операнда Rn. Полуслово, участвующее в умножении, берется из второго операнда Rm. Оно специфицируется символом Y:

cond
код условного выполнения команды (опция)

	B - bottom halfword – младшее полуслово Rm, биты [15:0]; T - top halfword – старшее полуслово Rm, биты[31:16].
	Rd – Регистр назначения. Если регистр назначения опущен, то регистром назначения будет регистр первого операнда **Rn**.
	Rn, Rm – Регистры, содержащие первый и второй операнды – источники. Из них извлекаются множители (из первого слово или полуслово, а из второго – полуслово).
	Ra – Регистр аккумулятор, к которому добавляется результат умножения.

Операция

Команды SMLABB, SMLABT, SMLATB, SMLATT:

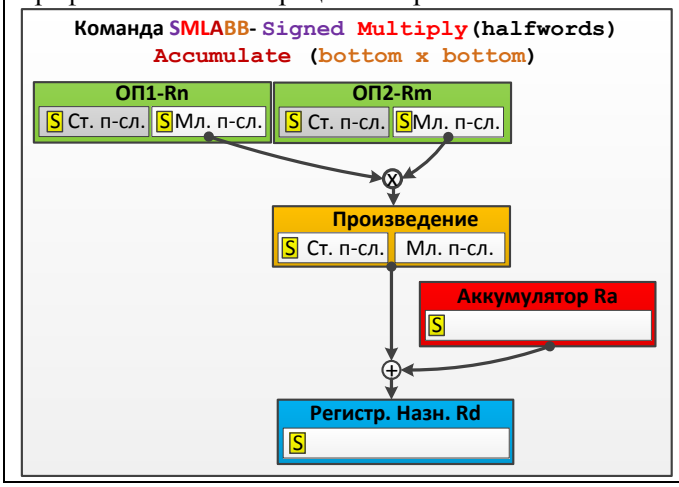
■ Выполняют знаковое умножение специфицированных полуслов из первого и второго регистров-операндов соответственно:

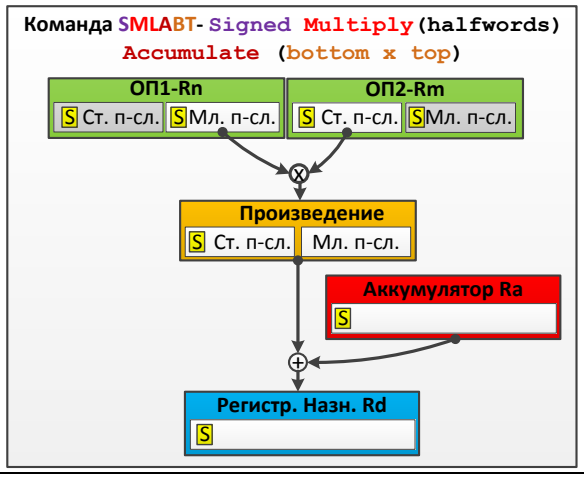
BB - младших полуслов первого и второго регистров-операндов;
BT - младшего полуслова первого регистра-операнда на старшее полуслово второго;
TB - старшего полуслова первого регистра-операнда на младшее полуслово второго;
TT - старших полуслов первого и второго регистров-операндов.

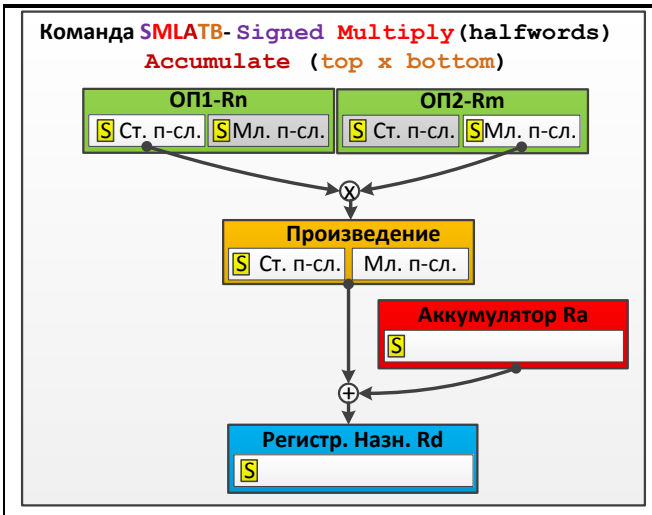
■ Добавляют полученный 32-разрядный результат умножения к значению в регистре-аккумуляторе Ra.

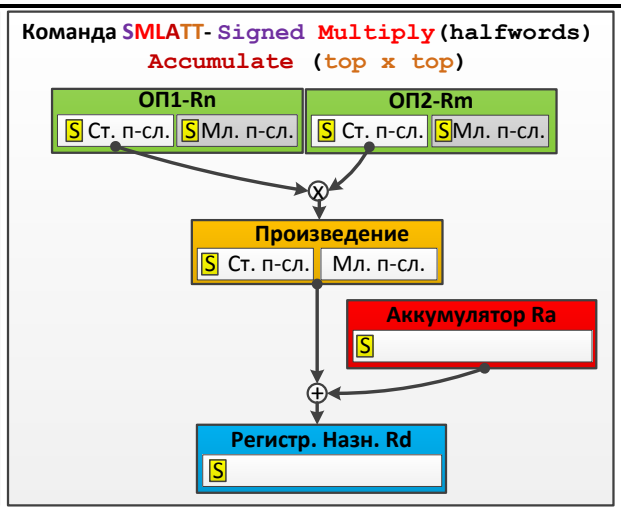
■ Сохраняют результат умножения с накоплением в регистре назначения Rd.

Не специфицированные значения полуслов в регистрах первого и второго операндов игнорируются (не используются в операции умножения).
Знаковые переполнения при аккумулировании *контролируются*. Если они возникают, то устанавливается флаг Q (насыщения).
Графическая иллюстрация алгоритма выполнения команд представлена ниже:









Команды SMLAWB и SMLAWT :

■ Умножают 32-разрядное слово со знаком из первого регистра-операнда Rn на полуслово со знаком из второго регистра-операнда:

B - на младшее полуслово второго операнда Rm;

T - на старшее полуслово второго операнда Rm.

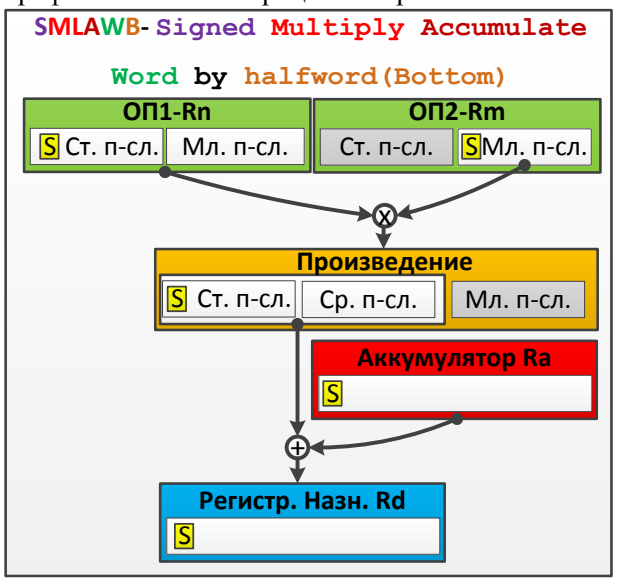
■ Добавляют старшие 32-разряда полученного 48-разрядного произведения S48 = (S32)*(S16) к 32-разрядному значению в аккумуляторе Ra.

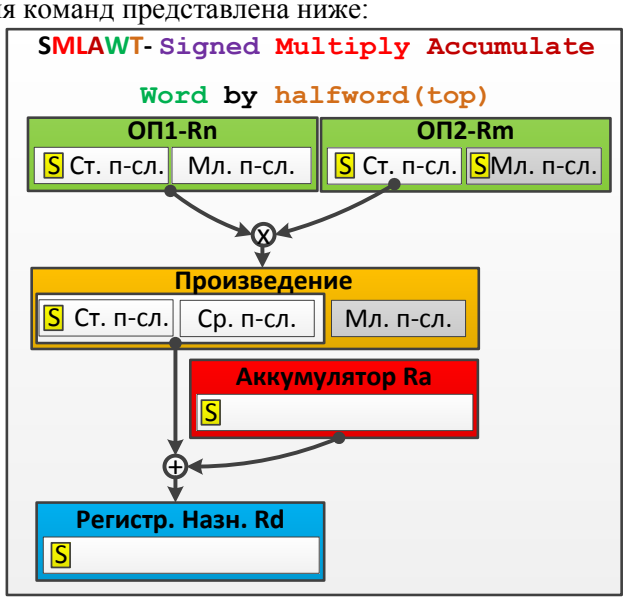
■ Сохраняют результат умножения с накоплением в регистре назначения Rd.

Внимание! Младшие 16 бит 48-разрядного произведения теряются. Предполагается, что форматы исходных множителей таковы, что младшие 16-разрядов произведения можно отбросить без какой-либо потери точности.

Если в результате операции сложения (накопления) в аккумуляторе возникает переполнение overflow (знаковое), то команда устанавливает флаг Q (насыщения) в статусном регистре APSR.

Графическая иллюстрация алгоритма выполнения команд представлена ниже:





Ограничения

В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Если в процессе операции накопления возникает переполнение overflow, то флаг Q выставляется.

Примеры

```
SMLABB R5, R6, R4, R1 ; Умножает младшие полуслова
                      ; регистров первого R6 и второго R4
                      ; операндов, добавляет произведение к
                      ; значению в регистре аккумуляторе R1
                      ; и сохраняет результат в R5

SMLATB R5, R6, R4, R1 ; Умножает старшее полуслово первого
                      ; регистра-операнда R6 на младшее
                      ; полуслово второго R4, добавляет
                      ; полученное произведение к значению в
                      ; регистре-аккумуляторе R1 и
                      ; сохраняет сумму в регистре-приемнике
                      ; R5

SMLATT R5, R6, R4, R1 ; Умножает старшие полуслова
                      ; регистров первого R6 и второго R4
                      ; операндов, добавляет произведение к
                      ; значению в регистре аккумуляторе R1
                      ; и сохраняет результат в регистре-
                      ; приемнике R5

SMLABT R5, R6, R4, R1 ; Умножает младшее полуслово первого
                      ; регистра-операнда R6 на старшее
                      ; полуслово второго R4, добавляет
                      ; полученное произведение к значению в
                      ; регистре-аккумуляторе R1 и
                      ; сохраняет сумму в регистре-приемнике
                      ; R5

SMLABT R4, R3, R2     ; Внимание! Опущен регистр назначения
                      ; Им по умолчанию будет регистр
                      ; первого операнда R4
                      ; Умножает младшее полуслово первого
                      ; регистра-операнда R4 на старшее
                      ; полуслово второго R3, добавляет
                      ; полученное произведение к значению в
                      ; регистре-аккумуляторе R2 и
                      ; сохраняет сумму в регистре-приемнике R4

SMLAWB R10, R2, R5, R3     ; Умножает 32-разрядное слово в первом
                           ; регистре-операнде R2 на младшее
                           ; полуслово во втором регистре
                           ; операнде R5. Добавляет старшее 32-
                           ; разрядное слово произведения к
                           ; значению в регистре-аккумуляторе R3
                           ; и сохраняет результат в R10

SMLAWT R10, R2, R1, R5     ; Умножает 32-разрядное слово в первом
                           ; регистре-операнде R2 на старшее
                           ; полуслово во втором регистре
                           ; операнде R1. Добавляет старшее 32-
                           ; разрядное слово произведения к
                           ; значению в регистре-аккумуляторе R5
                           ; и сохраняет результат в R10
```

4.6 Двойное знаковое умножение полуслов с аккумулированием обоих 32-разрядных произведений

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMLAD	Signed Multiply Accumulate Dual Знаковое умножение полуслов с аккумулированием двойное		v	v

Синтаксис

```
SMLAD{X}{cond} Rd, Rn, Rm, Ra
```

Где:

SMLAD	Signed Multiply Accumulate Dual Двойное знаковое умножение полуслов и аккумулирование обоих произведений
SMLADX	Signed Multiply Accumulate Dual Reverse Двойное знаковое реверсивное умножение (перекрестное)полуслов и аккумулирование обоих произведений

X - Специфицирует, какие полуслова из исходных регистров-операндов Rn и Rm используются в операциях умножения:

X отсутствует	Умножаются младшие и старшие полуслова, соответственно, в регистрах множителей Rn, Rm: bottom x bottom; top x top
X присутсвует	Перекрестное умножение: младшего полуслова первого регистра Rn на старшее полуслово второго регистра; старшего полуслова первого регистра Rn на младшее полуслово второго Rm: bottom x top; top x bottom

cond
код условного выполнения команды (опция)

Rd - Регистр назначения.

Rn, Rm - Регистры, содержащие первый и второй операнды с полусловами - множителями

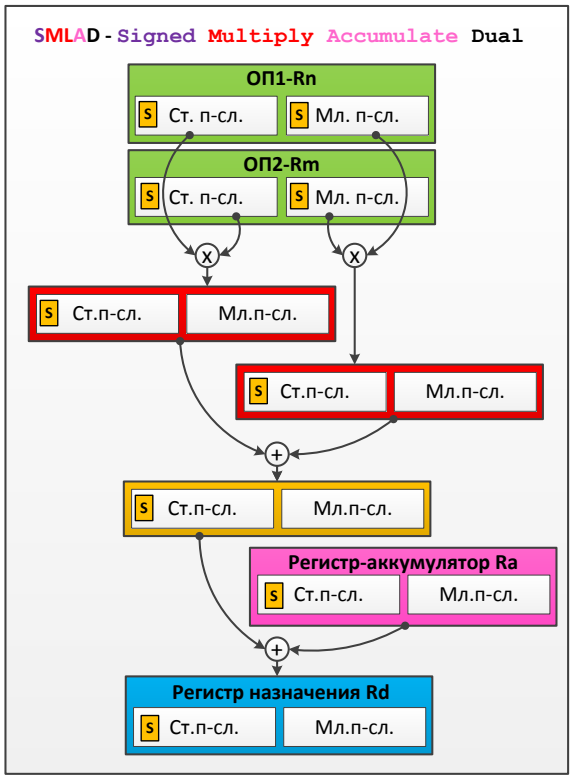
Ra - Регистр-аккумулятор: содержит аккумулируемое значение

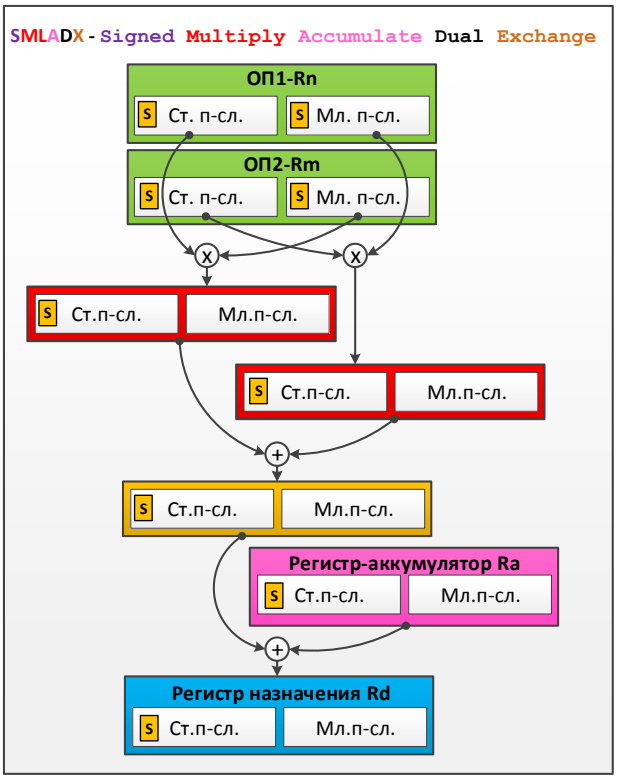
Операция

Команды SMLAD и SMLADX выполняют одновременно две операции умножения полуслов со знаком (16-разрядных), расположенных в регистрах исходных операндов Rn и Rm (32-разрядных).

■ Команда SMLAD выполняет умножение старших и младших полуслов исходных операндов, соответственно, с получением двух 32-разрядных произведений со знаком.
■ Команда SMLADX выполняет перекрестное умножение старших и младших полуслов со знаком исходных операндов (старшее полуслово первого операнда на младшее полуслово второго операнда и младшее полуслово первого операнда на старшее полуслово второго операнда) с получением двух 32-разрядных произведений со знаком.
■ Далее оба полученных произведения добавляются к значению в регистре-аккумуляторе Ra.
■ Результат двойного умножения и аккумулирования в виде 32-разрядного слова со знаком сохраняется в регистре назначения Rd.

Графическая иллюстрация алгоритма выполнения этих команд представлена ниже:





Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команды не изменяют состояние флагов

Примеры

```
SMLAD R10, R2, R1, R5      ; Знаковое умножение двух полуслов
                           ; первого регистра-операнда R2 на
                           ; знаковые полуслова второго
                           ; регистра-операнда R1
                           ; Добавление обоих произведений к
                           ; значению в аккумуляторе R5
                           ; и сохранение результата в регистре
                           ; назначения R10
SMLADX R0, R2, R4, R6      ; Перекрестное умножение старшего
                           ; полуслова R2 на младшее полуслово
                           ; R4 и младшего полуслова R2 на
                           ; старшее полуслово R4
                           ; Добавление обоих произведений к R6
                           ; Сохранение накопленной суммы в R0
```

4.7 Длинное умножение слов со знаком и без знака с получением 64-разрядного произведения, длинное знаковое и без знаковое умножение с накоплением 64-разрядного произведения, длинное без знаковое умножение с двойным аккумулированием.

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMULL, UMULL	Signed and Unsigned Long Multiply, using 32-bit operands and producing a 64-bit result Знаковое и без знаковое длинное умножение 32-разрядных слов с генерацией 64-разрядного произведения	v	v	v
SMLAL, UMLAL	Signed and Unsigned Long Multiply with Accumulate, using 32-bit operands and producing a 64-bit result. Знаковое и без знаковое длинное умножение 32-разрядных слов с получением 64-разрядного произведения и аккумулированием			
UMAAL	Unsigned Long Multiply with Accumulate Accumulate Без знаковое длинное умножение с двойным аккумулированием		v	v

Синтаксис

op{cond} RdLo, RdHi, Rn, Rm

Где:

op:

UMULL	Unsigned Long Multiply Беззнаковое длинное умножение
SMULL	Signed Long Multiply Знаковое длинное умножение
UMLAL	Unsigned Long Multiply, with Accumulate Беззнаковое длинное умножение с аккумулированием
SMLAL	Signed Long Multiply, with Accumulate Знаковое длинное умножение с аккумулированием
UMAAL	Unsigned Long Multiply with Accumulate Accumulate Беззнаковое длинное умножение с двойным аккумулированием

cond
код условного выполнения команды (опция)

RdHi, RdLo – регистры назначения (будут содержать длинное произведение). Для команд длинного умножения с накоплением UMLAL и SMLAL содержат также аккумулируемое значение (64-разрядный аккумулятор)

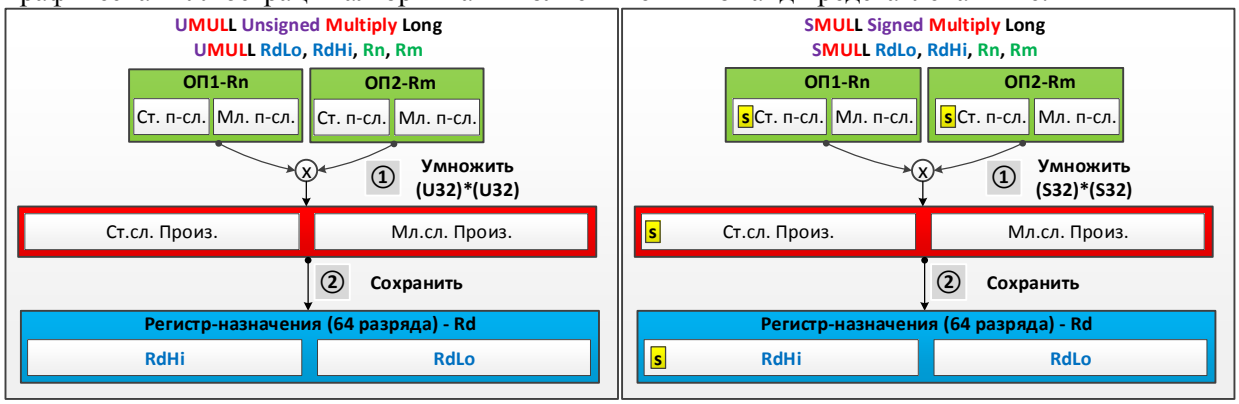
Rn, Rm – Регистры первого и второго операндов

Операция

Команда **UMULL** беззнакового длинного умножения интерпретирует значения в регистрах исходных операндов Rn и Rm как 32-разрядные числа без знака. Умножает их друг на друга (U32) * (U32) и сохраняет 64-разрядное произведение без знака в регистре назначения (RdHi, RdLo).

Команда **SMULL** знакового длинного умножения интерпретирует значения в регистрах исходных операндов Rn и Rm как 32-разрядные числа со знаком в дополнительном коде. Умножает их друг на друга (S32) * (S32) и сохраняет 64-разрядное произведение со знаком в регистре назначения (RdHi, RdLo).

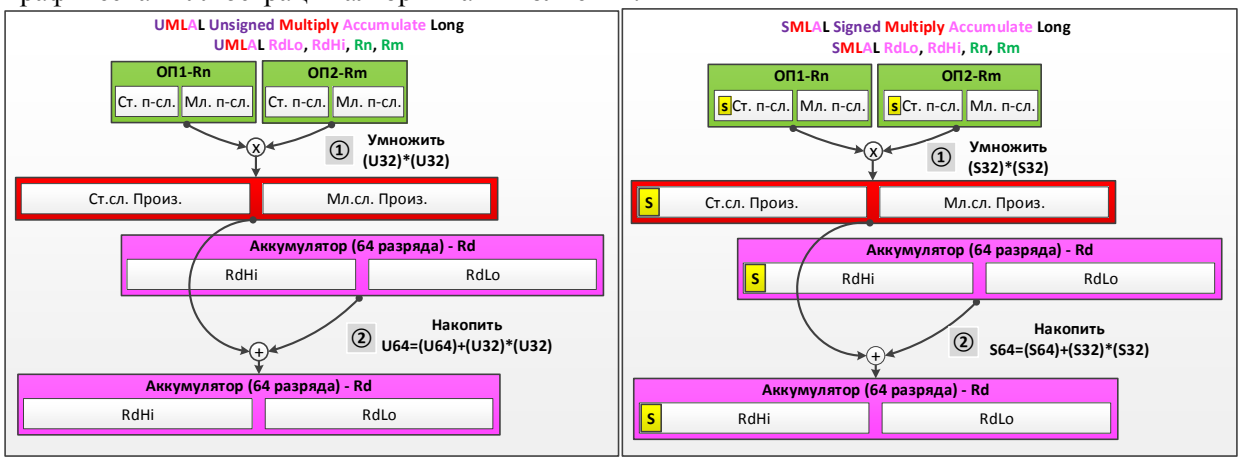
Графическая иллюстрация алгоритма выполнения этих команд представлена ниже:



Команда **UMLAL** беззнакового длинного умножения с аккумулированием интерпретирует значения в регистрах исходных операндов Rn и Rm как 32-разрядные числа без знака. Умножает их друг на друга (U32) * (U32) с получением 64-разрядного произведения без знака. Добавляет это произведение к текущему 64-разрядному числу без знака в аккумуляторе (RdHi, RdLo) и сохраняет результат накопления там же (RdHi, RdLo).

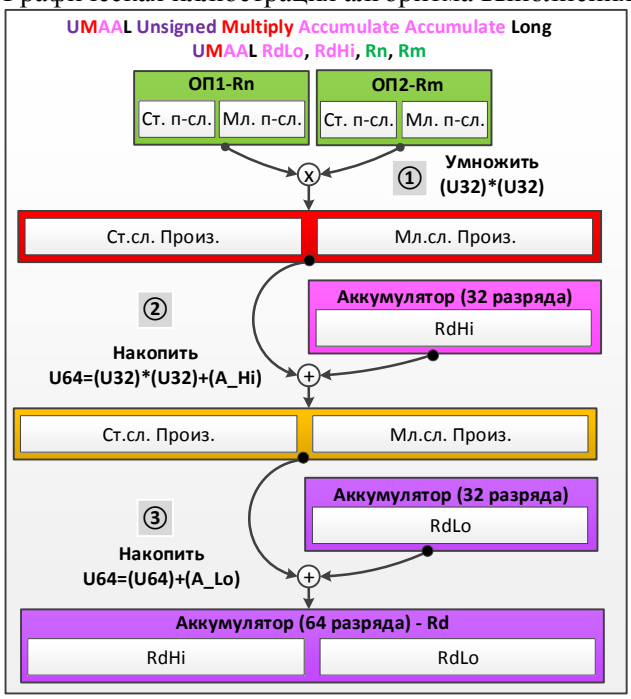
Команда **SMLAL** знакового длинного умножения с аккумулированием интерпретирует значения в регистрах исходных операндов Rn и Rm как 32-разрядные числа со знаком в дополнительном коде. Умножает их друг на друга (S32) * (S32) с получением 64-разрядного произведения со знаком. Добавляет это произведение к текущему 64-разрядному целому числу со знаком в аккумуляторе (RdHi, RdLo) и сохраняет результат накопления там же (RdHi, RdLo).

Графическая иллюстрация алгоритма выполнения:



Команда **UMAAL** без знакового длинного умножения с двойным аккумулированием интерпретирует значения в регистрах исходных операндов Rn и Rm как 32-разрядные целые числа без знака. Умножает их друг на друга (U32) * (U32) с получением 64-разрядного произведения. Добавляет к полученному 64-разрядному произведению 32-разрядное целое число без знака из регистра RdHi, а затем 32-разрядное целое число без знака из регистра RdLo. Сохраняет полученный результат двойного аккумулирования в том же самом регистре аккумуляторе (RdHi, RdLo).

Графическая иллюстрация алгоритма выполнения команды представлена ниже:



Ограничения

- В команде нельзя использовать указатель стека SP и счетчик команд PC.
- Регистры аккумулятора RdHi и RdLo должны быть разными.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
UMULL R0, R4, R5, R6  ; Беззнаковое длинное умножение
                      ; слова в регистре R5 на слово в регистре R6
                      ; Сохранение в длинном 64-разрядном
                      ; регистре назначения (R4,R0) = (R5) x (R6)

SMLAL R4, R5, R3, R8  ; Знаковое длинное умножение
                      ; слова в регистре R3 на слово в регистре R8
                      ; с аккумулированием результата в 64-
                      ; разрядном регистре (R5, R4)
                      ; Signed (R5,R4) = (R5,R4) + (R3) x (R8)

UMAAL R3, R6, R2, R7  ; Без знаковое длинное умножение
                      ; слова в регистре R2 на слово в регистре R7
                      ; Добавление к 64-разрядному произведению
                      ; 32-разрядного числа в R6, а затем в R3
                      ; Сохранение результата в (R6, R3)
```

4.8 Знаковое умножение полуслов с длинным аккумулированием. Двойное знаковое умножение полуслов с длинным аккумулированием

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMLAL SMLALD	Signed Multiply (halfwords) Accumulate Long Dual Знаковое умножение полуслов с длинным аккумулированием (64 разряда) Двойное знаковое умножение полуслов с длинным аккумулированием (64 разряда)		V	V

Синтаксис

```
SMLAL{X}{Y}{cond} RdLo, RdHi, Rn, Rm   ; Умножение знаковых полуслов
                                       ; операндов X и Y
                                       ; 64-разрядное накопление
SMLALD{X}{cond} RdLo, RdHi, Rn, Rm     ; Два умножения знаковых полуслов
                                       ; (обычное или перекрестное)
                                       ; 64-разрядное накопление двух
                                       ; произведений
```

Где:

SMLAL{X}{Y}	Signed Multiply (halfwords, X and Y) Accumulate Long Знаковое умножение двух полуслов (16-разрядов), специфицированных суффиксами {X}{Y} с длинным аккумулированием произведения (64-разряда)
{X}{Y}	Специфицируют полуслова исходных регистров-операндов Rn и Rm, которые используются в качестве множителей. Поле {X} - определяет полуслово первого регистра-операнда Rn. Поле {Y} определяет полуслово второго регистра-операнда Rm. B — Bottom – Младшее полуслово регистра, биты[15:0]. T — Top – Старшее полуслово регистра, биты [31:16].
SMLALD	Signed Multiply Accumulate Long Dual Двойное знаковое умножение полуслов с длинным накоплением. Умножаются старшие полуслова исходных операндов и, соответственно, - младшие полуслова: **(bottom × bottom)** и **(top × top)**.

cond
код условного выполнения команды (опция)

SMLALDX	Signed Multiply Accumulate Long Dual Exchange Двойное знаковое умножение полуслов реверсивное (перекрещенными операндами) с длинным накоплением: **(bottom × top)** и **(top × bottom)**.	

RdHi, **RdLo** – Регистры назначения и одновременно – длинный 64-разрядный аккумулятор. RdLo – младшие 32 бита и RdHi – старшие 32 бита 64-разрядного числа со знаком.

Rn, **Rm** – Регистры, содержащие первый и второй операнды с полусловами-множителями.

Операция

Во всех командах
SMLALBB, **SMLALBT**, **SMLALTB**, **SMLALTT**, **SMLALD** и **SMLALDX**:

Первый символ **S** означают работу с числами со знаком, символы **ML** – умножение, символ **A** аккумулирование, **L** – длинное с расширением произведения до 64-х разрядов.

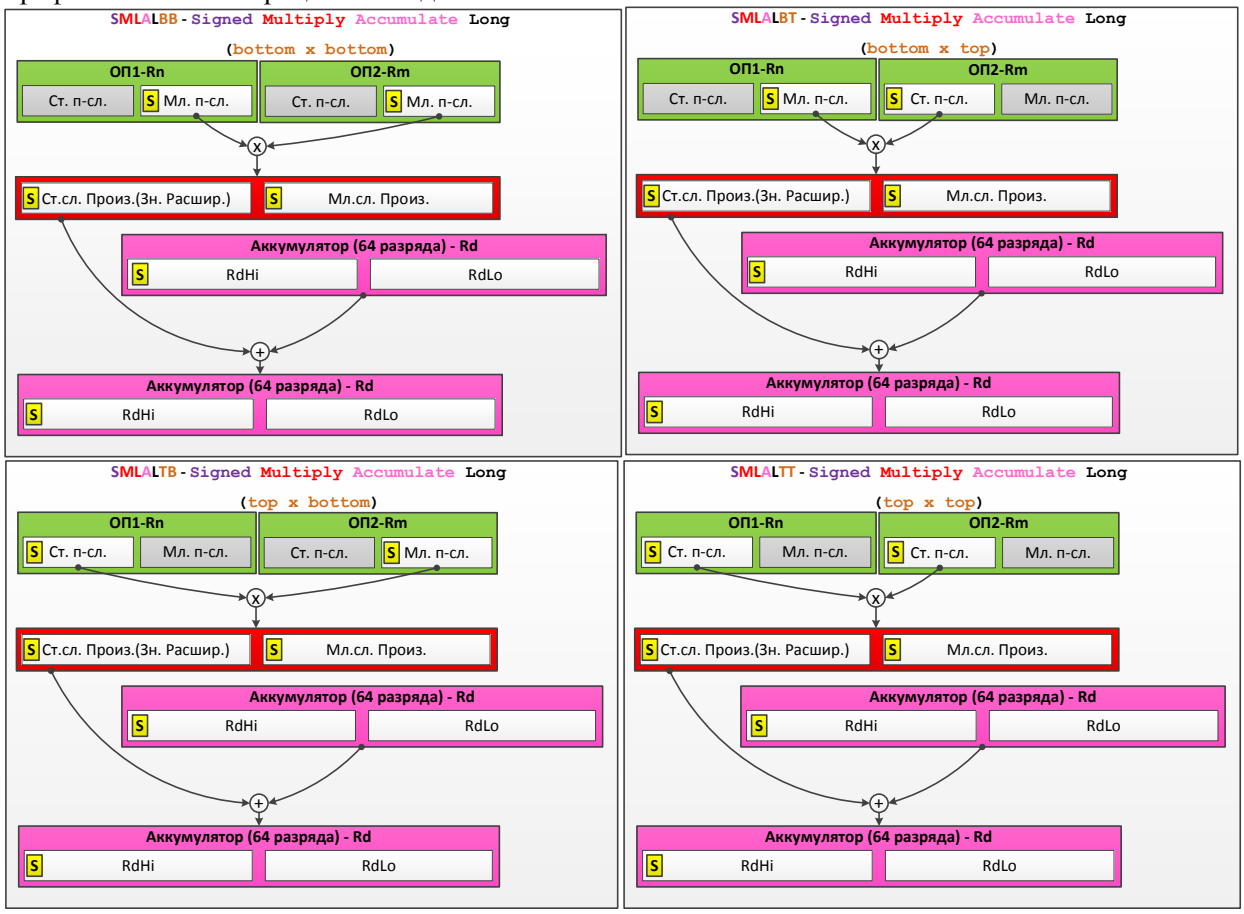
Суффиксы **BB**, **BT**, **TB**, **TT** означают, что в команде выполняется только одна специфицированная этими суффиксами операция умножения полуслов, расположенных в регистрах-источниках:
BB bottom * bottom (младшее * младшее);
BT bottom * top (младшее * старшее);
TB top * bottom (старшее * младшее);
TT top * top (старшее * старшее).

Суффикс **D** означает, что выполняются одновременно две операции умножения полуслов в регистрах-источниках с последующим накоплением обоих произведений в 64-разрядном длинном аккумуляторе. При этом выполняется умножение одноименных полуслов: (младшее * младшее) и (старшее * старшее).

Если к суффиксу **D** добавляется суффикс **X** (с обменом, перекрещиванием), то две операции умножения полуслов выполняются с перекрещиванием исходных операндов: (младшее * старшее) и (старшее * младшее). Далее оба произведения накапливаются в 64-разрядном аккумуляторе.

Команды SMLALBB, SMLALBT, SMLALTB и SMLALTT :
■ Умножают знаковые полуслова первого и второго операндов, специфицированные двумя последними суффиксами BB, BT, TB, TT.
■ Выполняют знаковое расширение 32-разрядного произведения до 64-х разрядов и добавляют «длинное произведение» к значению в 64-разрядном регистре-аккумуляторе (RdHi, RdLo).
■ Сохраняют полученный 64-разрядный результат в регистрах аккумулятора (RdHi, RdLo).
Не специфицированные в команде полуслова исходных регистров игнорируются (не участвуют в операции умножения).

Графическая иллюстрация команд:



Команды SMLALD и SMLALDX интерпретируют значения в регистрах исходных операндов Rn и Rm как 4-е 16-разрядных числа со знаком в дополнительном коде:

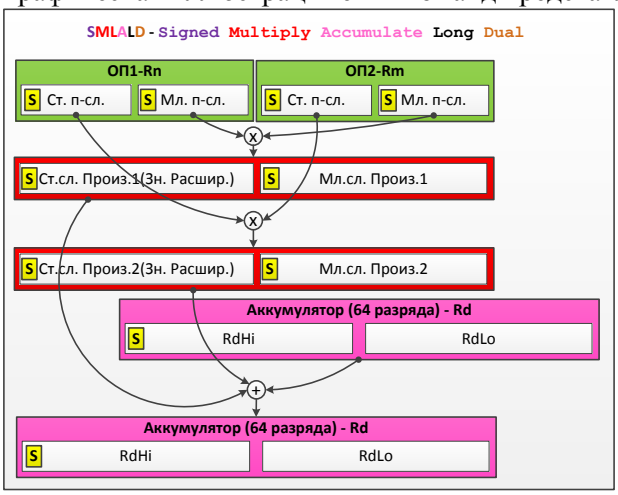
■ Если суффикс «X» в конце команды отсутствует (SMLALD), то полуслова со знаком из первого операнда Rn умножаются на соответствующие полуслова со знаком второго операнда Rm, т.е. старшее полуслово на старшее полуслово и младшее полуслово на младшее полуслово (top x top; bottom x bottom).

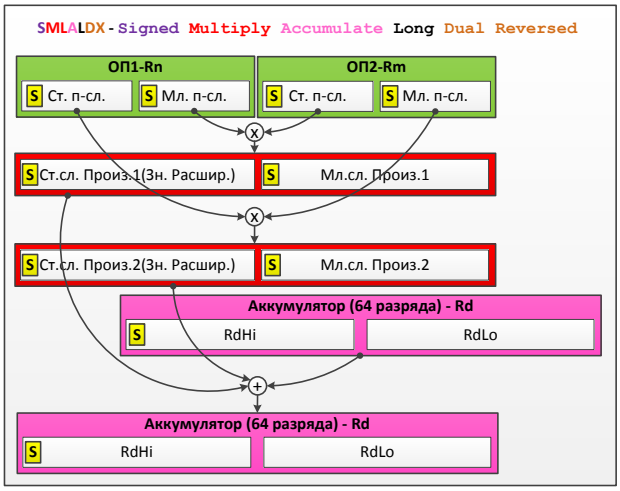
■ Если суффикс «X» в конце команды присутствует (SMLALDX), то выполняется «перекрестное» умножение полуслов исходных операндов со знаком Rn и Rm, т.е. старшее полуслово первого регистра-операнда на младшее полуслово второго регистра-операнда и младшее полуслово первого регистра-операнда на старшее полуслово второго регистра-операнда (top x bottom; bottom x top).

■ Полученные 32-разрядные произведения расширяются до 64-х разрядов с учетом знака и добавляются к текущему значению в 64-разрядном аккумуляторе (RdHi, RdLo), накапливаются.

■ Результат накопления сохраняется в том же 64-разрядном аккумуляторе (RdHi, RdLo).

Графическая иллюстрация этих команд представлена ниже:





Ограничения

- В команде нельзя использовать указатель стека SP и счетчик команд PC.
- Регистры аккумулятора RdHi и RdLo должны быть разными регистрами.

Выработка флагов

Команды не изменяют состояние флагов

Примеры

```
SMLALBT R2, R1, R6, R7      ; Знаковое умножение младшего
                            ; полуслова первого операнда R6 на
                            ; старшее полуслово второго операнда
                            ; R7. Знаковое расширение произведения
                            ; до 64-х бит.
                            ; Добавление 64-разрядного
                            ; произведения к значению в
                            ; аккумуляторе (R1:R2)

SMLALTB R2, R1, R6, R7      ; Знаковое умножение старшего
                            ; полуслова первого операнда R6
                            ; на младшее полуслово второго
                            ; операнда R7. Знаковое расширение
                            ; произведения до 64-х бит.
                            ; Добавление 64-разрядного
                            ; произведения к значению в
                            ; аккумуляторе (R1:R2)

SMLALD R6, R8, R5, R1       ; Знаковое умножение старших полуслов
                            ; первого R5 и второго R1 регистров
                            ; операндов. Знаковое расширение
                            ; произведения до 64-х разрядов.
                            ; Знаковое умножение младших полуслов
                            ; первого R5 и второго R1 регистров
                            ; операндов. Знаковое расширение
                            ; произведения до 64-х разрядов.
                            ; Накопление обоих произведений в
                            ; регистрах аккумулятора (R8:R6)
```

```
SMLALDX R6, R8, R5, R1      ; Знаковое умножение старшего
                            ; полуслова первого регистра-операнда
                            ; R5 на младшее полуслово второго
                            ; регистра-операнда R1
                            ; Знаковое расширение
                            ; произведения до 64-х разрядов.
                            ; Знаковое умножение младшего
                            ; полуслова первого регистра-операнда
                            ; R5 на старшее полуслово второго
                            ; регистра-операнда R1
                            ; Знаковое расширение
                            ; произведения до 64-х разрядов.
                            ; Накопление обоих произведений в
                            ; регистрах аккумулятора (R8:R6)
```

4.9 Двойное знаковое умножение полуслов, вычитание произведений и накопление разности в 32-разрядном или 64-разрядном аккумуляторе

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMLSD SMLSLD	Signed Multiply Subtract Dual Знаковое умножение двойное (полуслов), вычитание произведений, аккумулирование разности в 32-разрядном аккумуляторе Signed Multiply Subtract Long Dual Знаковое умножение двойное (полуслов) вычитание произведений, длинное аккумулирование разности в 64-разрядном аккумуляторе		V	V

Синтаксис

```
op{X}{cond} Rd, Rn, Rm, Ra      ; Команда SMLSD с 32-разрядным
                                ; накоплением
op{X}{cond} RdLo, RdHi, Rn, Rm  ; Команда SMLSLD с 64-разрядным
                                ; накоплением
```

Где:

op:

SMLSD	Signed Multiply Subtract Dual Знаковое умножение двойное (полуслов), вычитание произведений, аккумулирование разности
SMLSLD	Signed Multiply Subtract Long Dual Знаковое умножение двойное (полуслов) вычитание произведений длинное, аккумулирование разности

X - Специфицирует, какие полуслова из исходных регистров-операндов Rn и Rm используются в операциях умножения:

cond
код условного выполнения команды (опция)

X отсутствует	Умножаются младшие и старшие полуслова, соответственно, в регистрах множителей Rn, Rm: bottom x bottom; top x top
X присутствует	Перекрестное умножение: младшего полуслова первого регистра Rn на старшее полуслово второго регистра; старшего полуслова первого регистра Rn на младшее полуслово второго Rm: bottom x top; top x bottom

Rd – Регистр назначения.

Rn, Rm – Регистры, содержащие первый и второй операнды с полусловами - множителями

Ra – Регистр-аккумулятор (32 разряда): содержит аккумулируемое значение

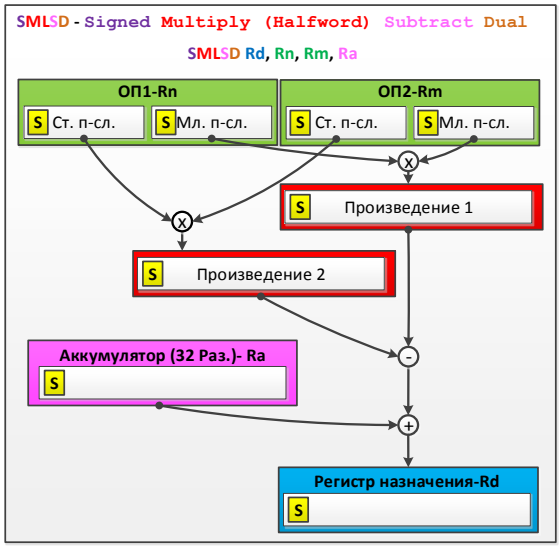
RdLo, RdHi – 64-разрядный регистр аккумулятор (для операций длинного накопления разности произведений **L**)

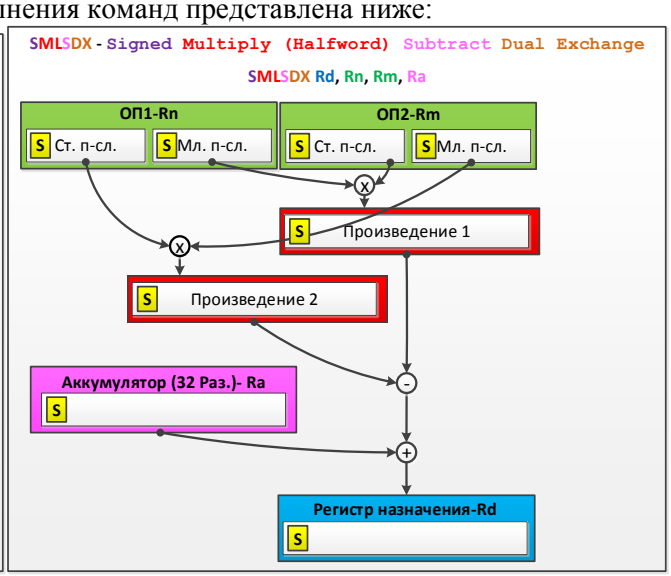
Операция

Команда SMLSD интерпретирует данные в первом и втором регистрах-операндах в качестве 4-х полуслов со знаком (16-разрядных чисел со знаком в дополнительном коде):

■ Если в команде присутствует суффикс «X», то меняет местами полуслова второго операнда.
■ Выполняет два знаковых умножения полуслов из первого и второго регистров-операндов (16 x 16).
■ Вычитает результат умножения старших полуслов из результата умножения младших полуслов – (обычное 32-разрядное вычитание).
■ Добавляет результат вычитания произведений к текущему значению в регистре-аккумуляторе (32-разрядном).
■ Сохраняет результат сложения в регистре назначения.

Графическая иллюстрация алгоритма выполнения команд представлена ниже:

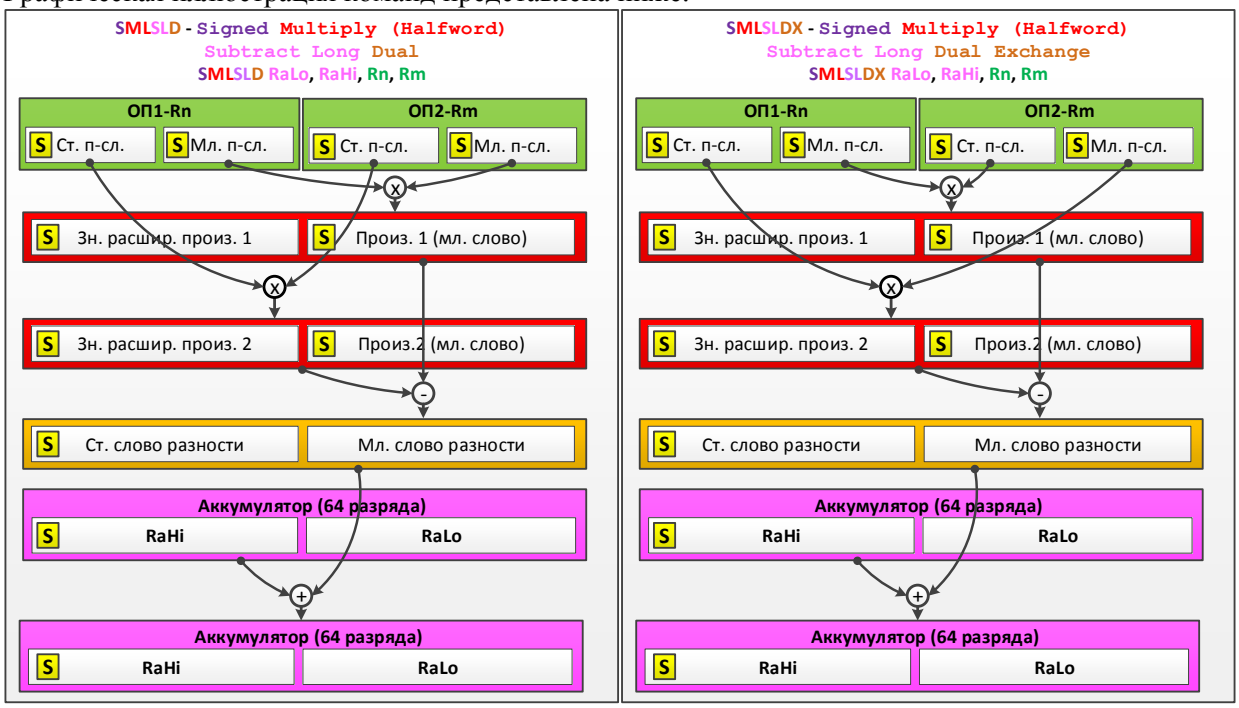




Команда SMLSLD также интерпретирует данные в первом и втором регистрах-операндах в качестве 4-х полуслов со знаком в дополнительном коде:
■ Если в команде присутствует суффикс «X», то меняет местами полуслова второго операнда.
■ Выполняет два знаковых умножения полуслов из первого и второго регистров-операндов (16 x 16).
■ Вычитает результат умножения старших полуслов из результата умножения младших полуслов.

■ Расширяет результат вычитания произведений до 64-х разрядов с учетом знака и добавляет к текущему значению в 64-разрядном аккумуляторе (RdHi, RdLo).
■ Сохраняет 64-разрядный результат в аккумуляторе (RdHi, RdLo).

Графическая иллюстрация команд представлена ниже:



Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды устанавливают флаг Q (насыщения) в том случае, если во время операции аккумулирования происходит переполнение.

Примеры

```
SMLSD R0, R4, R5, R6  ; Умножает старшие и младшие полуслова
                      ; первого R4 и второго R5 регистров-
                      ; операндов: top x top; bottom x bottom
                      ; Вычитает из произведения младших
                      ; произведение старших полуслов
                      ; Добавляет разность произведений к
                      ; аккумулятору (32 разряда) в регистре R6 и
                      ; сохраняет результат в регистре назначения
                      ; R0
SMLSDX R1, R3, R2, R0 ; Умножает младшее полуслово R3 на
                      ; старшее полуслово R2
                      ; Умножает старшее полуслово R3 на
                      ; младшее полуслово R2
                      ; Вычитает из первого произведения
                      ; второе
                      ; Добавляет разность к значению в
                      ; аккумуляторе R0
                      ; Сохраняет результат в регистре
                      ; назначения R1
```

```
SMLSLD R3, R6, R2, R7      ; Выполняет умножение младших и
                           ; старших полуслов первого R2 и
                           ; второго R7 регистров-операндов
                       ; Вычитает из первого произведения
                       ; второе и расширяет разность
                       ; до 64-х разрядов
                       ; Добавляет 64-разрядную разность к
                       ; текущему значению в 64-разрядном
                       ; аккумуляторе (R6:R3) и сохраняет
                       ; в аккумуляторе (R6:R3)

SMLSLDX R3, R6, R2, R7     ; Выполняет умножение младшего полу-
                           ; слова первого операнда R2 на старшее
                           ; полу-слово второго операнда R7
                           ; Выполняет умножение старшего полу-
                           ; слова первого операнда R2 на младшее
                           ; полуслово второго операнда R7
                           ; Вычитает из первого произведения
                           ; второе и расширяет разность
                           ; до 64-х разрядов
                           ; Добавляет 64-разрядную разность к
                           ; текущему значению в 64-разрядном
                           ; аккумуляторе (R6:R3) и сохраняет
                           ; в аккумуляторе (R6:R3)
```

4.10 Умножение слов со знаком с получением старшего слова произведения

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMMUL	Signed Most Significant Word Multiply Знаковое умножение слов с получением старшего значащего слова произведения		v	v

Синтаксис

op{R}{cond} Rd, Rn, Rm

Где:

op:

SMMUL	Signed Most Significant Word Multiply

R - rounding flag – Флаг округления

R отсутствует	Результат усекается, обрезается **(truncated)**. Младшее слово произведения отбрасывается
R присутсвует	Результат округляется **(rounded)** В этом случае к 64-разрядному произведению добавляется константа **0x80000000** перед тем, как старше слово выделяется **(extracted)** из произведения.

cond код условного выполнения команды (опция)

Rd – Регистр назначения.

Rn, Rm – Регистры, содержащие первый и второй операнды с 32-разрядными словами - множителями

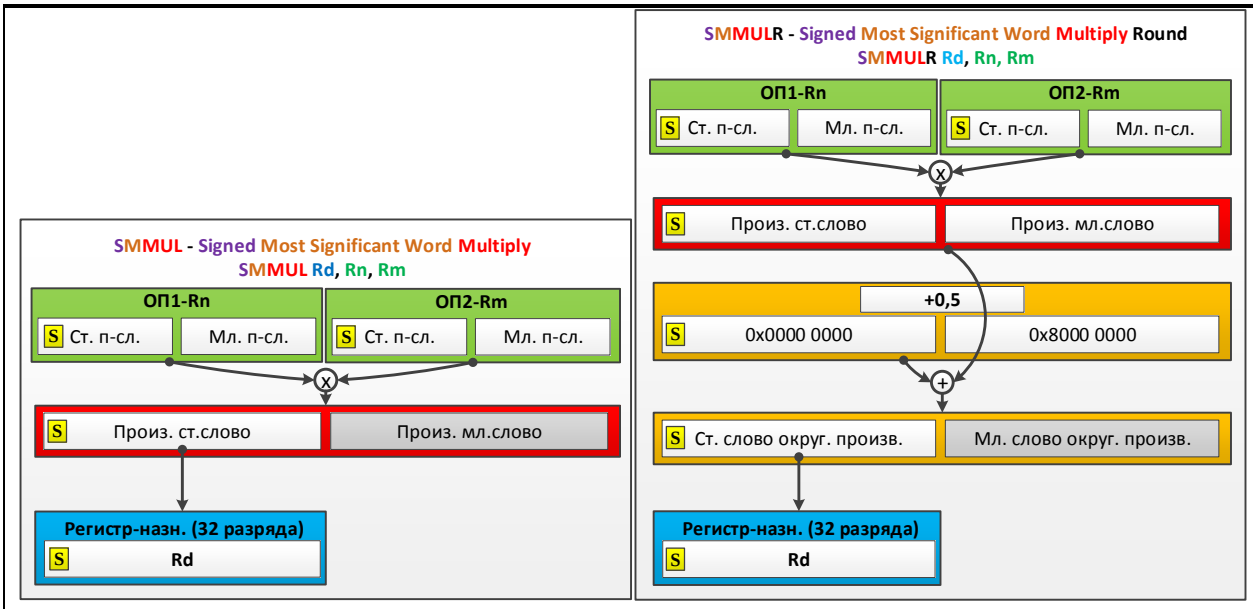
Операция

Команда SMMUL выполняет знаковое умножение слов в регистрах исходных операндов Rn и Rm, рассматривая исходные операнды как 32-разрядные числа со знаком в дополнительном коде:

■ Умножает 32-разрядные слова со знаком в регистрах операндов Rn и Rm, получая 64-разрядное произведение.
■ Если указана опция округления R, то добавляет к полученному произведению число 0x80000000, в противном случае – просто отбрасывает младшее 32-разрядное слово.
■ Сохраняет старшие значащие 32 бита результата в регистре-назначения Rd.

Пример: используется для вычисления произведения 32-разрядных дробей со знаком в дополнительном коде, в формате (S1.31), и получения произведения в формате (S2.30) при отбрасывании младшего слова или для получения произведения вещественных чисел в формате (S16.16) с сохранением только целой части произведения.

Графическая иллюстрация алгоритма выполнения команд представлена ниже (для примера умножения чисел (S16.16)*(S16.16):



Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
SMMUL R0, R4, R5      ; Умножает 32-разрядные слова в регистрах-
                      ; операндах R4 и R5 как числа со знаком
                      ; Отбрасывает младшие 32 разряда и
                      ; сохраняет старшие 32 разряда произведения
                      ; в регистре-назначения R0
SMMULR R6, R2    ; Умножает 32-разрядные слова в
                 ; регистрах-операндов R6 и R2 как знаковые
                 ; Выполняет округление произведения
                 ; Записывает старшее слово произведения в
                 ; в регистр-назначения R6
```

4.11 Знаковое умножение слов с положительным или отрицательным аккумулированием старшего значащего слова произведения

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SMMLA SMMLS	Signed Most Significant Word Multiply Accumulate Знаковое умножение слов и аккумулирование старшего значащего слова произведения Signed Most Significant Word Multiply Subtract Знаковое умножение слов и вычитание (отрицательное аккумулирование) старшего значащего слова произведения		V	V

Синтаксис

op{R}{cond} Rd, Rn, Rm, Ra

Где:

op:

SMMLA	Signed Most Significant Word Multiply Accumulate Знаковое умножение слов и аккумулирование старшего значащего слова произведения
SMMLS	Signed Most Significant Word Multiply Subtract Знаковое умножение слов и вычитание (отрицательное аккумулирование) старшего значащего слова произведения

R - rounding flag – Флаг округления

R отсутствует	Результат усекается, обрезается **(truncated)**, младшее слово произведения отбрасывается
R присутсвует	Результат округляется **(rounded)** В этом случае к 64-разрядному произведению добавляется константа **0x80000000** перед тем, как старшее слово выделяется **(extracted)** из произведения.

cond
код условного выполнения команды (опция)

Rd – Регистр назначения.

Rn, Rm – Регистры, содержащие первый и второй операнды со словами - множителями

Ra – Регистр- аккумулятор

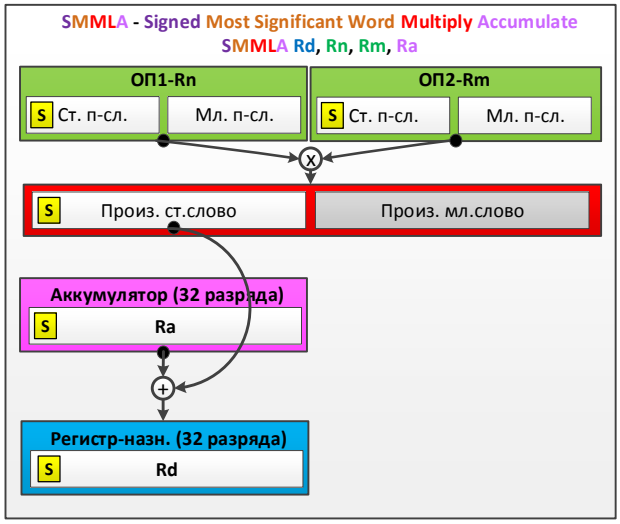
Операция

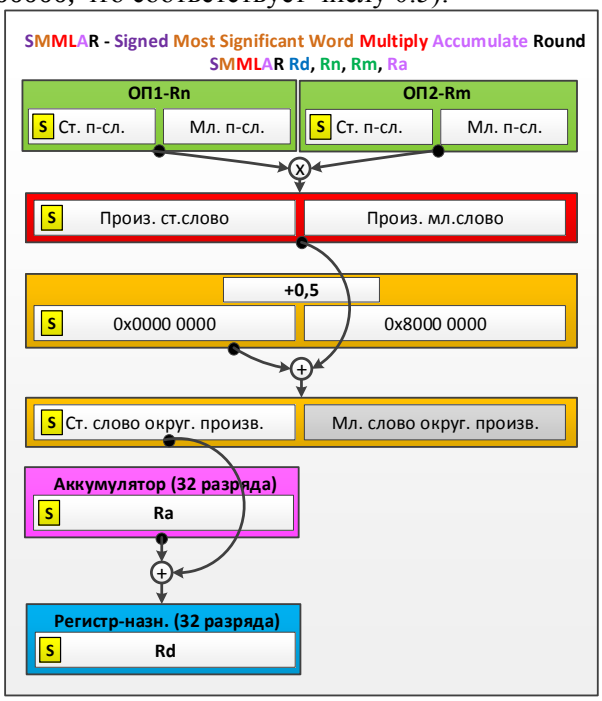
Команда **SMMLA** выполняет знаковое умножение слов в регистрах Rn и Rm, рассматривая исходные операнды как 32-разрядные числа со знаком в дополнительном коде:

■ Умножает 32-разрядные слова со знаком в регистрах операндов Rn и Rm, получая 64-разрядное произведение.

■ Если указана опция округления R, то добавляет к полученному произведению число 0x80000000
■ Выделяет старшие значащие 32 бита результата.
■ Добавляет полученное значение к значению в аккумуляторе Ra.
■ Сохраняет результат накопления в регистре назначения Rd.

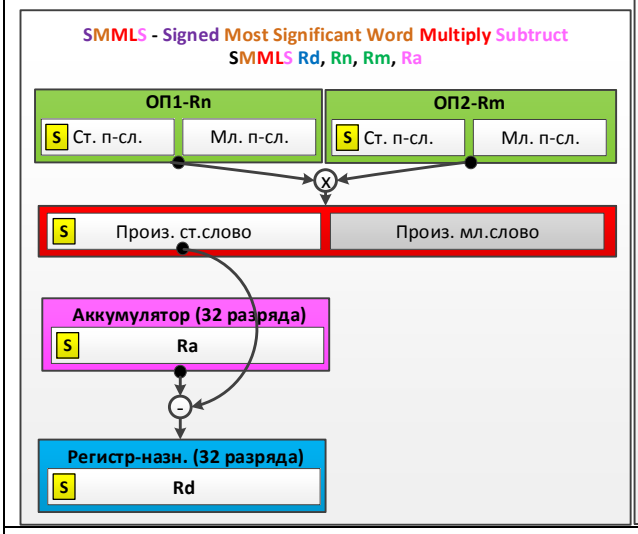
Графическая иллюстрация алгоритма выполнения команд представлена ниже (для примера умножаются числа в формате S16.16*S16.16 с получением результата S32.32. В режиме округления к младшему слову добавляется 0x80000000, что соответствует числу 0.5):

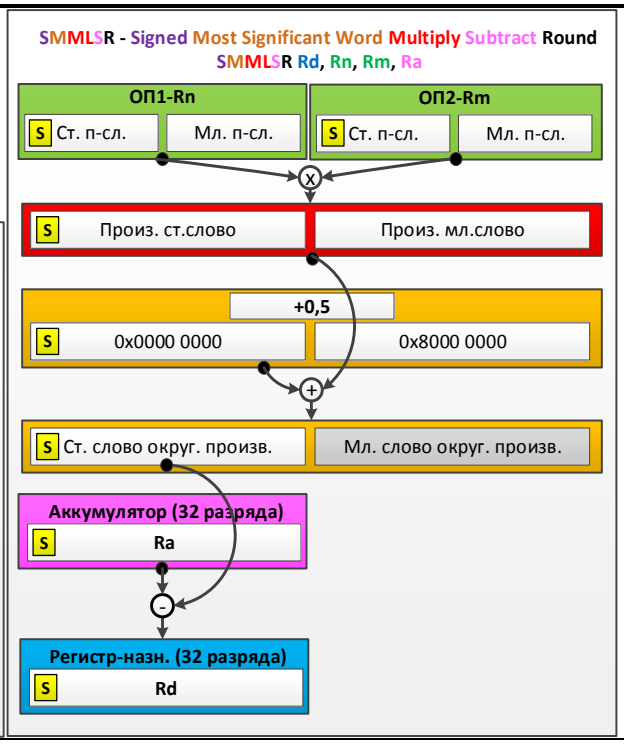




Аналогичные действия выполняет и команда **SMMLS**, за исключением того, что накопление – отрицательное (вычитание):
■ Умножает 32-разрядные слова со знаком в регистрах операндов Rn и Rm, получая 64-разрядное произведение.
■ Если указана опция округления R, то добавляет к полученному произведению число 0x80000000
■ Выделяет старшие значащие 32 бита результата.
■ Вычитает полученное значение из значения в аккумуляторе Ra.
■ Сохраняет результат вычитания в регистре назначения Rd.

Графическая иллюстрация алгоритма выполнения команд представлена ниже (для исходных форматов данных предыдущего примера):





Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Команда не оказывает влияния на флаги

Примеры

```
SMMLA R0, R4, R5, R6  ; Умножает слова со знаком(32-разряда) в
                      ; первом R4 и втором R5 операндах, получая
                      ; 64-разрядное произведение
                      ; Выделяет старшие 32 бита произведения
                      ; отбрасывая младшее 32-разрядное слово
                      ; произведения
                      ; Добавляет результат к аккумулятору R6
                      ; Сохраняет сумму в регистре назначения R0

SMMLAR R6, R2, R1, R4 ; Умножает слова со знаком(32-разряда) в
                      ; первом R2 и втором R1 операндах, получая
                      ; 64-разрядное произведение
                      ; Округляет результат, добавляя 0x80000000
                      ; Выделяет старшие 32 бита произведения
                      ; (округленные)
                      ; Добавляет к аккумулятору R4
                      ; Сохраняет сумму в регистре назначения R6

SMMLSR R3, R6, R2, R7; Умножает слова со знаком(32-разряда) в
                      ; первом R6 и втором R2 операндах, получая
                      ; 64-разрядное произведение
                      ; Округляет результат, добавляя 0x80000000
                      ; Выделяет старшие 32 бита произведения
                      ; (округленные)
                      ; Вычитает из значения в аккумуляторе R7
                      ; Сохраняет результат
                      ; в регистре назначения R3
```

П1.4 КОМАНДЫ УМНОЖЕНИЯ С НАКОПЛЕНИЕМ, ДЕЛЕНИЯ

```
SMMLS R4, R5, R3, R8 ; Умножает слова со знаком(32-разряда) в
                     ; первом R5 и втором R3 операндах, получая
                     ; 64-разрядное произведение
                     ; Выделяет старшие 32 бита произведения
                     ; отбрасывая младшее 32-разрядное слово
                     ; произведения
                     ; Вычитает результат из аккумулятора R8
                     ; Сохраняет сумму в регистре назначения R4
```

5 КОМАНДЫ НАСЫЩЕНИЯ

Оглавление

5.1. Насыщение слова со знаком/без знака 92

5.2. Параллельное насыщение 2-х полуслов со знаком/без знака 95

5.3. Сложение/вычитание слов со знаком с насыщением. Параллельное сложение/вычитание 2-х полуслов или 4-х байт со знаком с насыщением 96

5.4. Параллельное суммирование/вычитание 2-х полуслов или 4-х байт как чисел без знака с насыщением 99

5.5. Параллельное сложение и вычитание 2-х перекрещенных полуслов со знаком с насыщением 101

5.6. Параллельное сложение и вычитание 2-х перекрещенных полуслов без знака с насыщением 104

5.7. Знаковое удвоение второго операнда с насыщением и сложение/вычитание с первым операндом со знаковым насыщением 107

5.1 Насыщение слова со знаком/без знака

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SSAT USAT	Signed Saturate – Знаковое насыщение (ограничение) слов Unsigned Saturate – Без знаковое насыщение (ограничение) слов Насыщение с любой битовой позиции с опциональной возможностью сдвига слова перед насыщением	v	v	v

Синтаксис

```
op{cond} Rd, #n, Rm {, shift #s}
```

Где:

op оптокод один из:

SSAT - Saturates a signed value to a signed range. Насыщение знаковой переменной в диапазоне чисел со знаком.

USAT - Saturates a signed value to an unsigned range. Насыщение знаковой переменной в диапазоне чисел без знака.

cond код условного выполнения команды (опция)

Rd – Регистр назначения.

n – Specifies the bit position to saturate to. Определяет битовую позицию, до которой нужно выполнить ограничение (насыщение)

- n – от **1** до **32** для SSAT (знаковое насыщение)
- n – от **0** to **31** для USAT (без знаковое насыщение)

Rm - Регистр-источник данных для операции насыщения.

shift #s - опционально заданная величина сдвига, который применяется к данным в регистре-источнике Rm перед выполнением операции насыщения. Возможны следующие варианты сдвига:

ASR #s - Арифметический сдвиг вправо, где s в диапазоне 1-31

LSL #s - Логический сдвиг влево, где s в диапазоне 1-31

Операция

Эти команды ограничивают (насыщают) переменную знаковым или без знаковым n-разрядным значением.

Команда **SSAT** сначала выполняет сдвиг (если он специфицирован) значения в регистре-источнике, а затем, - ограничение максимально-допустимым диапазоном $(-2^{n-1}) \le x \le (2^{n-1}-1)$ чисел со знаком в дополнительном коде. При этом *n – требуемое число двоичных разрядов числа, которыми будет ограничиваться результат*.

Для 8-разрядных чисел со знаком (байт): $(-2^{8-1}) \le x \le (2^{8-1}-1)$, т.е. $-128 \le x \le +127$;

Для 16-разрядных чисел со знаком (полуслов): $(-2^{16-1}) \le x \le (2^{16-1}-1)$, т.е. $-32\,768 \le x \le +32\,767$;

Для 32-разрядных чисел со знаком (слов): $(-2^{32-1}) \le x \le (2^{32-1}-1)$, т.е.

$-2\,147\,483\,648 \le x \le +2\,147\,483\,647$.

Число разрядов n может задаваться в диапазоне от 1 до 32. Но, один разряд (n=1) для представления чисел со знаком в дополнительном коде позволяет представить только -1 и +0 (только знаковый разряд). Будьте осторожны с малыми значениями n!

Команда **USAT** сначала выполняет специфицированный сдвиг (если он задан) значения в регистре-источнике, а затем ограничивает результат диапазоном чисел без знака заданной разрядности n: $0 \le x \le 2^n-1$. Для этой команды число разрядов n может быть задано в интервале от 0 до 31. При минимальном n=0 ограничение числа выполняется на уровне нуля, при n=1 – в диапазоне 0÷1, при n=2 – в диапазоне 0÷3 и т.д.

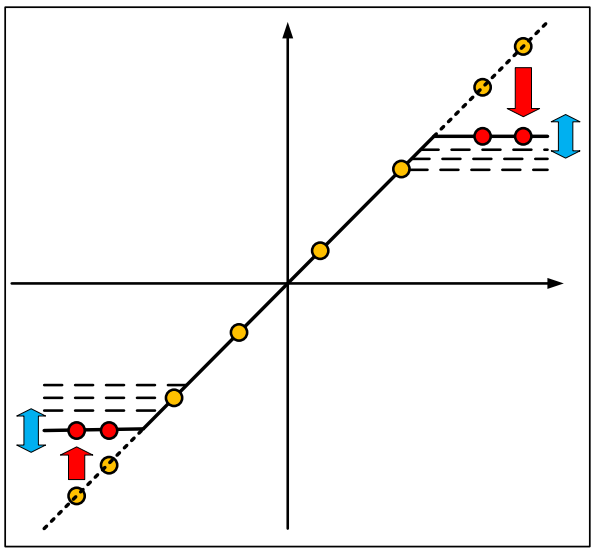
Чаще всего используется ограничение знакового операнда диапазоном n-разрядных байт, полуслов или слов без знака:
n=8: $(0 \le x \le 2^8-1)$ или $(0 \le x \le 255)$
n=16: $(0 \le x \le 2^{16}-1)$ или $(0 \le x \le 65535)$
n=31: $(0 \le x \le 2^{31}-1)$ или $(0 \le x \le 2\ 147\ 483\ 647)$

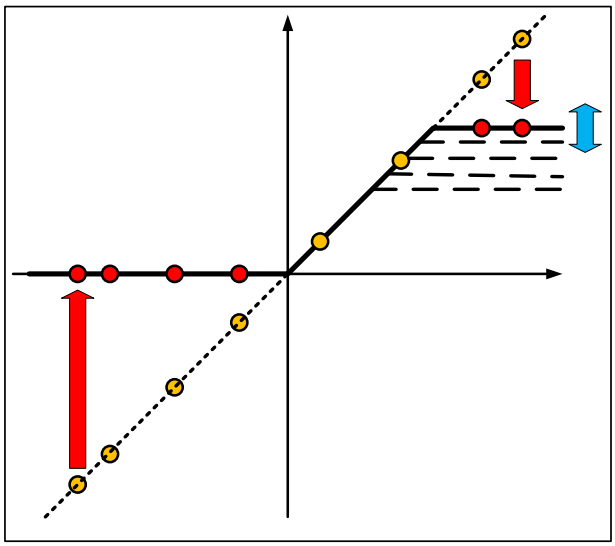
Знаковое n-разрядное ограничение с использованием команды SSAT означает следующее:
■ Если значение, которое ограничивается, меньше минимально возможного отрицательного числа -2^{n-1}, то возвращается минимальное возможное отрицательное число -2^{n-1}, т.е. происходит «насыщение (ограничение) снизу».
■ Если значение, которое ограничивается, больше максимально возможного положительного числа $2^{n-1}-1$, то возвращается максимально возможное положительное число $2^{n-1}-1$, т.е. происходит «насыщение (ограничение) сверху».
■ В противном случае, возвращается то же самое значение.

Таким образом, реализуется симметричное ограничение как сверху, так и снизу, как в аналоговых операционных усилителях. При этом граница насыщения почти симметрична относительно нуля и может произвольно устанавливаться программистом.

Без знаковое n-разрядное ограничение с использованием команды USAT означает следующее:
■ Если значение, которое ограничивается, меньше нуля (отрицательное), то возвращается минимально возможное число без знака, т.е. ноль 0.
■ Если значение, которое ограничивается, больше чем 2^n-1, то возвращается максимально возможное число без знака в пределах заданной разрядной сетки 2^n-1.
■ В противном случае, возвращается то же самое значение.

Графическая иллюстрация симметричного ограничения диапазоном чисел со знаком заданной разрядности и несимметричного ограничения диапазоном чисел без знака заданной разрядности представлена ниже:





Если возвращенный результат отличается от исходного значения, то произошло насыщение (saturation). В этом случае флаг насыщения Q в статусном регистре приложения APSR устанавливается в 1 и фиксируется («защелкивается»). В противном случае, его значение остается неизменным. Сбросить флаг насыщения можно только используя команду записи в регистр специального назначения MSR (по технологии «Чтение-Модификация-Запись»). Автоматическая очистка флага насыщения Q при чтении не поддерживается.
Замечание.
Команды условного перехода по состоянию флага насыщения отсутствуют. Для того, чтобы выполнить такой переход сначала считайте содержимое регистра APSR в один из регистров ЦПУ,

затем протестируйте этот флаг с использование команды TST, после чего выполните переход по одному из условий EQ/NE.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Если при выполнении команд возникает «насыщение», то флаг Q устанавливается в 1.

Примеры

```
SSAT R7, #16, R7, LSL #4   ; Логический сдвиг влево на 4 разряда
                           ; операнда в регистре-источнике R7
                           ; ограничение как сверху, так и снизу
                           ; максимальным 16-разрядным числом со
                           ; знаком и сохранение результата в том
                           ; же самом регистре R7

USATNE R0, #7, R5     ; Команда условного ограничения по NE
                      ; значения в регистре-источнике R5
                      ; без знаковым числом в диапазоне 0-127
                      ; с сохранением результата в R0.
```

5.2 Параллельное насыщение 2-х полуслов со знаком/без знака

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SSAT16 **USAT16**	Signed Saturate – Знаковое ограничение (насыщение) двух полуслов одновременно Unsigned Saturate – Без знаковое ограничение (насыщение) двух полуслов одновременно. Ограничение выполняется для полуслов до заданной битовой позиции (до заданной разрядности результата)		V	V

Синтаксис

```
op{cond} Rd, #n, Rm
```

Где:		
	op оптокод один из: **SSAT16** - Saturates a signed halfword value to a signed range. Ограничение двух знаковых полуслов в заданном диапазоне знаковых чисел. **USAT16** - Saturates a signed halfword value to an unsigned range. Ограничение двух знаковых полуслов в заданном диапазоне чисел без знака.	**cond** код условного выполнения команды (опция)
	Rd - Регистр назначения.	
	n - Specifies the bit position to saturate to. Специфицирует битовую позицию, до которой нужно выполнить ограничение (насыщение) ■ n - от **1** до **16** для SSAT (знаковое насыщение) ■ n - от **0** to **15** для USAT (без знаковое насыщение)	
	Rm - Регистр-источник данных для операции насыщения.	

Операция

Команда **SSAT16**:

1. Выполняет ограничение (насыщение) одновременно двух знаковых полуслов (16 разрядов) в регистре источнике диапазоном допустимых чисел со знаком заданной разрядности n (максимальное значение n = 16): $(-2^{n-1}) \le x \le (2^{n-1}-1)$
2. Сохраняет результат в виде двух полуслов со знаком (16 разрядов) в регистре назначения (32-х разрядном регистре).

Команда **USAT16**:

1. Выполняет ограничение (насыщение) двух знаковых 16-разрядных чисел (полуслов) допустимым диапазоном чисел без знака заданной разрядности. При этом максимальное значение n=15: $(0) \le x \le (2^n-1)$, т.е. ограничение выполняется максимум в диапазоне 0÷ 32765.
2. Сохраняет оба без знаковых полуслова в регистре назначения.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Если при выполнении команд возникает «насыщение» (для любого из двух исходных операндов), то флаг Q устанавливается в 1.

Примеры

```
SSAT16 R7, #9, R2      ; Выполняет ограничение сверху и снизу
                       ; старшего и младшего знакового полуслова
                       ; в регистре-источнике R2 диапазоном
                       ; 9-разрядных чисел со знаком в
                       ; дополнительном коде и сохраняет оба
                       ; полуслова в регистре R7
USAT16NE R0, #13, R5   ; Выполняет команду условного ограничения по
                       ; условию NE двух полуслов со знаком
                       ; диапазоном положительных чисел без знака
                       ; с 13-ю разрядами
                       ; Сохраняет соответствующие полуслова
                       ; в регистре R0
```

5.3 Сложение/вычитание слов со знаком с насыщением. Параллельное сложение/вычитание 2-х полуслов или 4-х байт со знаком с насыщением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
QADD and QSUB	Saturating Add and Saturating Subtract, signed. Знаковое сложение с насыщением (ограничением) и Знаковое вычитание с насыщением (ограничением).		v	v

Синтаксис

```
op{cond} {Rd}, Rn, Rm
```

Где:

op оптокод один из:

QADD - Saturating 32-bit add.
Сложение 32-разрядных слов со знаком с насыщением

QADD16 - Saturating two 16-bit integer additions.
Сложение двух 16-разрядных полуслов со знаком с насыщением.

QADD8 - Saturating four 8-bit integer additions.
Сложение четырех 8-разрядных байт со знаком с насыщением.

QSUB - Saturating 32-bit subtraction.
Вычитание двух 32-разрядных слов со знаком с насыщением.

QSUB16 - Saturating two 16-bit integer subtraction.
Вычитание двух 16-разрядных полуслов со знаком с насыщением.

QSUB8 - Saturating four 8-bit integer subtraction.
Вычитание четырех 8-разрядных байт

cond
код условного выполнения команды (опция)

	со знаком с насыщением.	
	Rd – Регистр назначения.	
	Rn, Rm – Регистры, содержащие первый и второй операнды	

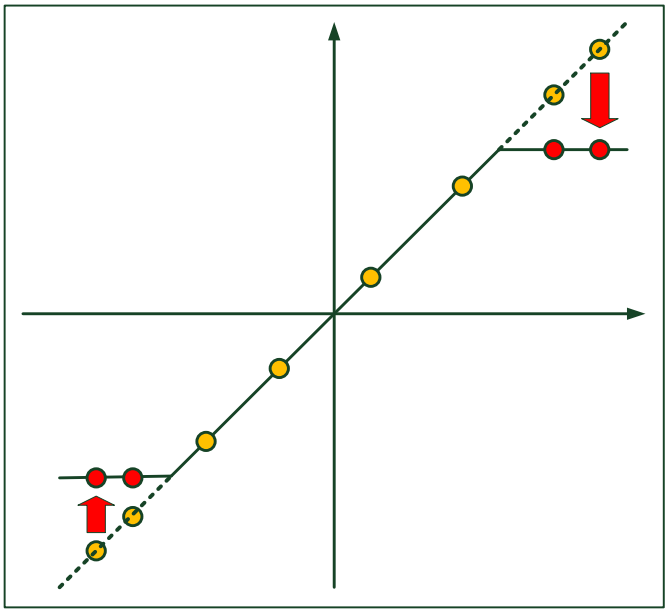
Операция

Эти команды выполняют **знаковое** сложение или вычитание слов, полуслов или байт из первого и второго операнда, насыщение (ограничение) результата и его запись в регистр назначения.

Команды QADD и QSUB сначала выполняют сложение или вычитание знаковых слов, а затем операцию насыщения результата максимально возможными знаковыми числами соответствующей разрядности:
$-2^{n-1} \le x \le 2^{n-1}-1$,
где n в зависимости от типа команды (работа со словом, полусловом или байтом) принимает значение 32, 16 или 8. При этом результат операции сложения/вычитания с насыщением будет в диапазоне:

Для 32-разрядных чисел со знаком (слов): $(-2^{32-1}) \le x \le (2^{32-1}-1)$, т.е.
$-2\,147\,483\,648 \le x \le +2\,147\,483\,647$
Для 16-разрядных чисел со знаком (полуслов): $(-2^{16-1}) \le x \le (2^{16-1}-1)$, т.е. $-32\,768 \le x \le +32\,767$
Для 8-разрядных чисел со знаком (байт): $(-2^{8-1}) \le x \le (2^{8-1}-1)$, т.е. $-128 \le x \le +127$

Графическая иллюстрация механизма «знакового ограничения» (насыщения) представлена ниже:



Если в результате операций сложения QADD или вычитания QSUB слов с насыщением, насыщение имеет место, то флаг Q в регистре статуса программы приложения APSR устанавливается в 1, в противном случае – не изменяется (операция сложения/вычитания слов с насыщением сопровождается фиксацией флага насыщения – «защелкиванием» флага).
При выполнении операций сложения/вычитания полуслов или байт с насыщением, флаг насыщения Q не модифицируется, даже если насыщение имеет место.
Замечание.
Для того, чтобы сбросить флаг насыщения Q, Вы должны применить команду MSR – пересылки данных из регистра общего назначения в регистр специального назначения – APSR. Сначала по команде MRS – считать содержимое статусного регистра APSR, протестировать флаг Q, сбросить его в копии, а затем сохранить копию обратно в регистре статуса (технология «Чтение-Модификация-Запись» регистров специального назначения).

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Если при выполнении команд сложения/вычитания слов с насыщением возникает «насыщение», то флаг Q устанавливается в 1.

Примеры

```
QADD16 R7, R4, R2  ; Складывает полуслова со знаком в регистрах
                   ; R4 и R2, выполняет ограничение диапазоном
                   ; максимально возможных 16-разрядных чисел и
                   ; сохраняет результаты-полуслова в регистре R7
QADD8 R3, R1, R6   ; Складывает байты со знаком в регистрах
                   ; R1 и R6, выполняет ограничение диапазоном
                   ; максимально возможных 8-разрядных чисел и
                   ; сохраняет результаты-байты в регистре R3
QSUB16 R4, R2, R3  ; Вычитает из полуслов со знаком в регистре
                   ; R2 соответствующие полуслова со знаком в
                   ; регистре R3 выполняет ограничение диапазоном
                   ; максимально возможных 16-разрядных чисел и
                   ; сохраняет результаты-полуслова в регистре R4
QSUB8 R4, R2, R5   ; Вычитает из байт со знаком в регистре R2
                   ; соответствующие байты со знаком в регистре R5
                   ; выполняет ограничение диапазоном
                   ; максимально возможных 8-разрядных чисел и
                   ; сохраняет результаты-байты в регистре R4
```

5.4 Параллельное суммирование/вычитание 2-х полуслов или 4-х байт как чисел без знака с насыщением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
UQADD and UQSUB	Unsigned Saturating Add and Unsigned Saturating Subtract. Без знаковое параллельное насыщение сумм и без знаковое насыщение разностей полуслов и байт без знака.		V	V

Синтаксис

```
op{cond} {Rd}, Rn, Rm
```

Где:		
	op оптокод один из: **UQADD16** - Unsigned Saturating two 16-bit integer additions. Без знаковое насыщение двух сумм 16-битных чисел без знака **UQADD8** - Unsigned Saturating four 8-bit integer additions. Без знаковое насыщение четырех сумм 8- битных чисел без знака **UQSUB16** - Unsigned Saturating two 16-bit integer subtractions. Без знаковое насыщение двух разностей 16- битных чисел без знака **UQSUB8** - Unsigned Saturating four 8-bit integer subtractions. Без знаковое насыщение четырех разностей 8-битных чисел без знака	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn, Rm – Регистры, содержащие первый и второй операнды	

Операция

Эти команды выполняют одновременно либо два сложения/вычитания 16-разрядных целых чисел без знака (полуслов) либо четыре сложения/вычитания целых 8-разрядных чисел без знака (байт), ограничение максимальным диапазоном представления целых чисел без знака - полуслов или байт и сохранение результатов в регистре назначения.

Команда **UQADD16**:
■ Складывает, соответственно, старшие и младшие полуслова первого и второго операндов, считая их целыми числами без знака.
■ Выполняет ограничение результатов сложения для каждого полуслова максимальным диапазоном представления чисел без знака $0 \le x \le (2^{16}-1) = 65535$ и сохранение соответствующих полуслов в регистре назначения.

Команда **UQADD8**:
■ Выполняет параллельно сложение четырех соответствующих байт первого и второго операндов, считая их целыми числами без знака.
■ Выполняет ограничение результатов сложения для каждого байта максимальным диапазоном представления чисел без знака $0 \le x \le (2^{8}-1) = 255$ и сохранение соответствующих байт в регистре назначения.

Команда **UQSUB16**:
■ Вычитает из старшего и младшего полуслов первого операнда соответствующие полуслова второго операнда, считая их целыми числами без знака.
■ Выполняет ограничение результатов вычитания для каждого полуслова максимальным диапазоном представления чисел без знака $0 \le x \le (2^{16}-1) = 65535$ и сохранение соответствующих полуслов в регистре назначения.

Команда **UQSUB8**:
■ Вычитает из каждого байта первого операнда соответствующий байт второго операнда, считая их целыми числами без знака.
■ Выполняет ограничение результатов вычитания для каждого байта максимальным диапазоном представления чисел без знака $0 \le x \le (2^{8}-1) = 255$ и сохранение соответствующих байт в регистре назначения.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Так как одновременно выполняется либо две, либо четыре операции сложения/вычитания, то может возникать ситуация нескольких насыщений, а флаг насыщения один. Поэтому, - он не меняется.

Примеры

```
UQADD16 R7, R4, R2    ; Складывает полуслова первого операнда R4
                      ; с соответствующими полусловами второго
                      ; операнда R2
                      ; Выполняет ограничение максимальным
                      ; диапазоном целых 16-разрядных чисел без
                      ; знака и сохраняет результаты в
                      ; соответствующих полусловах регистра
                      ; назначения R7
UQADD8 R4, R2, R5     ; Складывает байты первого операнда R2
                      ; с соответствующими байтами второго
                      ; операнда R5
                      ; Выполняет ограничение максимальным
                      ; диапазоном целых 8-разрядных чисел без
                      ; знака и сохраняет результаты в
                      ; соответствующих байтах регистра
                      ; назначения R4
UQSUB16 R6, R3, R0    ; Вычитает из полуслов первого операнда R3
                      ; соответствующие полуслова второго
                      ; операнда R0
                      ; Выполняет ограничение максимальным
                      ; диапазоном целых 16-разрядных чисел без
                      ; знака и сохраняет результаты в
                      ; соответствующих полусловах регистра
                      ; назначения R6
UQSUB8 R1, R5, R6     ; Вычитает из байтов первого операнда R5
                      ; соответствующие байты второго
                      ; операнда R6
                      ; Выполняет ограничение максимальным
                      ; диапазоном целых 8-разрядных чисел без
                      ; знака и сохраняет результаты в
                      ; соответствующих байтах регистра
                      ; назначения R1
```

5.5 Параллельное сложение и вычитание 2-х перекрещенных полуслов со знаком с насыщением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
QASX and QSAX	Saturating Add and Subtract with Exchange and Saturating Subtract and Add with Exchange, signed. Знаковое сложение и вычитание с насыщением перекрещенных операндов-полуслов. Знаковое вычитание и сложение с насыщением перекрещенных операндов-полуслов.		v	v

Синтаксис

```
op{cond} {Rd}, Rn, Rm
```

Где:	**op** оптокод один из: **QASX** - Add and Subtract with Exchange and Saturate. Сложение и вычитание с обменом, (перекрещенными полусловами), со знаковым насыщением. **QSAX** - Subtract and Add with Exchange and Saturate. Вычитание и сложение с обменом, (перекрещенными полусловами), со знаковым насыщением.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn, Rm – Регистры, содержащие первый и второй операнды	

Операция

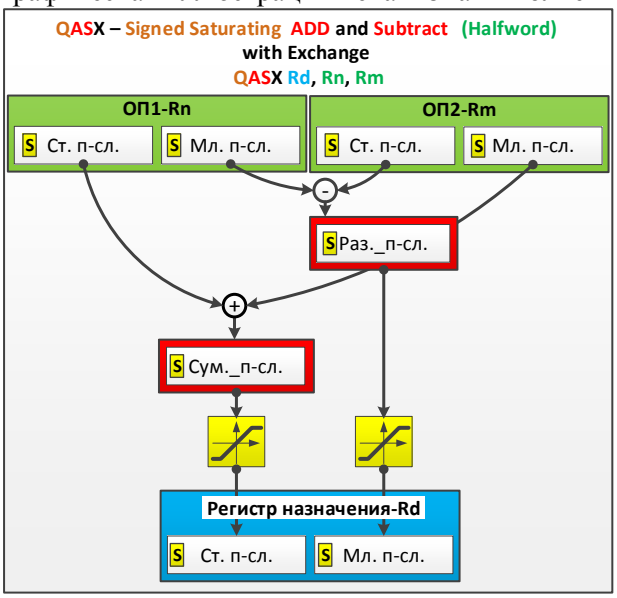
Команда **QASX** выполняет следующие действия:

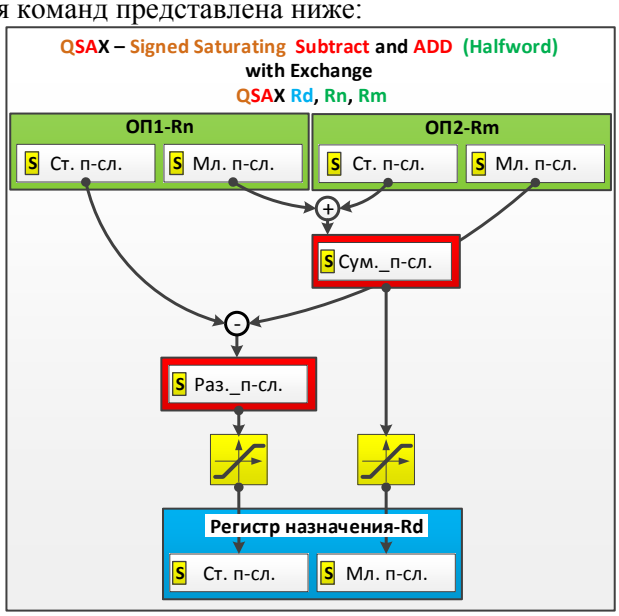
1. Складывает старшее полуслово первого операнда с младшим полусловом второго операнда, считая их целыми числами со знаком.
2. Вычитает из младшего полуслова первого операнда старшее полуслово второго операнда.
3. Выполняет ограничение (насыщение) результата сложения максимально возможным диапазоном 16-разрядных чисел со знаком ($-32768 \le x \le +32767$) и записывает результат насыщения в старшее полуслово регистра назначения.
4. Выполняет ограничение (насыщение) результата вычитания максимально возможным диапазоном 16-разрядных чисел со знаком ($-32768 \le x \le +32767$) и записывает результат насыщения в младшее полуслово регистра назначения.

Команда **QSAX** выполняет следующие действия:

1. Вычитает из старшего полуслова первого операнда младшее полуслово второго операнда, считая их целыми числами со знаком.
2. Складывает младшее полуслово первого операнда со старшим полусловом второго операнда.
3. Выполняет ограничение (насыщение) результата вычитания максимально возможным диапазоном 16-разрядных чисел со знаком ($-32768 \le x \le +32767$) и записывает результат насыщения в старшее полуслово регистра назначения.
4. Выполняет ограничение (насыщение) результата сложения максимально возможным диапазоном 16-разрядных чисел со знаком ($-32768 \le x \le +32767$) и записывает результат насыщения в младшее полуслово регистра назначения.

Результат каждой операции сложения/вычитания с насыщением будет в диапазоне:
$(-2^{16-1}) \le x \le (2^{16-1}-1)$, т.е. $-32\,768 \le x \le +32\,767$.
Графическая иллюстрация механизма выполнения команд представлена ниже:





Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Так как одновременно выполняются две команды сложения/вычитания слов с насыщением, то флаг «насыщения» Q не модифицируется (не определено в результате какой операции произошло насыщение).

Примеры

```
QASX R7, R4, R2 ; Складывает старшее полуслово первого регистра
                ; источника R4 с младшим полусловом второго
                ; регистра источника данных R2
                ; Выполняет насыщение (ограничение) результата
                ; максимально допустимым диапазоном 16-разрядных
                ; чисел со знаком
                ; Сохраняет результат в старшем полуслове
                ; регистра назначения R7
                ; Вычитает из младшего полуслова первого
                ; регистра источника R4 старшее полуслово второго
                ; регистра источника R2
                ; Выполняет насыщение (ограничение) результата
                ; максимально допустимым диапазоном 16-разрядных
                ; чисел со знаком
                ; Сохраняет результат в младшем полуслове
                ; регистра назначения R7
QSAX R0, R3, R5 ; Вычитает из старшего полуслова первого
                ; регистра-источника R3 младшее полуслово второго
                ; регистра-источника R5
                ; Выполняет насыщение (ограничение) результата
                ; максимально допустимым диапазоном 16-разрядных
                ; чисел со знаком
                ; Сохраняет результат в старшем полуслове
                ; регистра-назначения R0
                ; Складывает младшее полуслово первого регистра
                ; источника R3 со старшим полусловом второго
                ; регистра-источника R5
                ; Выполняет насыщение (ограничение) результата
                ; максимально-допустимым диапазоном 16-разрядных
                ; чисел со знаком
                ; Сохраняет результат в младшем полуслове
                ; регистра-назначения R0
```

5.6 Параллельное сложение и вычитание 2-х перекрещенных полуслов без знака с насыщением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
UQASX and UQSAX	Unsigned Saturating Add and Subtract with Exchange and Unsigned Saturating Subtract and Add with Exchange. Без знаковое насыщение суммы и разности перекрещенных операндов-полуслов. Без знаковое насыщение разности и суммы перекрещенных операндов-полуслов.		v	v

Синтаксис

```
op{cond} {Rd}, Rn, Rm
```

Где:	**op** оптокод один из: **UQASX** - Add and Subtract with Exchange and Saturate. Сложение и вычитание с обменом (перекрещенными полусловами) с ограничением диапазоном 16-разрядных чисел без знака **UQSAX** - Subtract and Add with Exchange and Saturate. Вычитание и сложение с обменом (перекрещенными полусловами) с ограничением диапазоном 16-разрядных чисел без знака.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn, Rm – Регистры, содержащие первый и второй операнды	

Операция

Команда **UQASX** выполняет следующие действия:

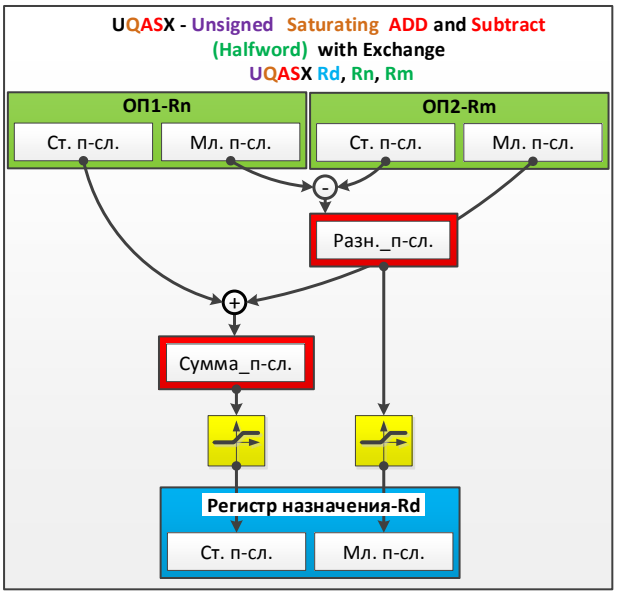
1. Складывает старшее полуслово первого операнда с младшим полусловом второго операнда.
2. Вычитает из младшего полуслова первого операнда старшее полуслово второго операнда.
3. Выполняет без знаковое ограничение (насыщение) результата сложения максимально возможным диапазоном 16-разрядных целых чисел без знака $0 \le x \le (2^{16}-1)=65535$, и записывает результат насыщения в старшее полуслово регистра назначения.
4. Выполняет без знаковое ограничение (насыщение) результата вычитания максимально возможным диапазоном 16-разрядных целых чисел без знака $0 \le x \le (2^{16}-1)=65535$, и записывает результат насыщения в младшее полуслово регистра назначения.

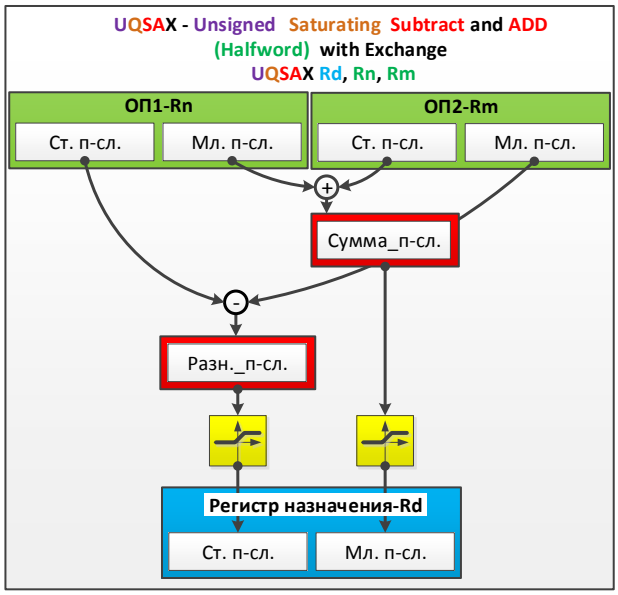
Команда **UQSAX** выполняет следующие действия:

1. Вычитает из старшего полуслова первого операнда младшее полуслово второго операнда.
2. Складывает младшее полуслово первого операнда со старшим полусловом второго операнда.
3. Выполняет без знаковое ограничение (насыщение) результата вычитания максимально возможным диапазоном 16-разрядных целых чисел без знака $0 \le x \le (2^{16}-1)=65535$, и записывает результат насыщения в старшее полуслово регистра назначения.

4. Выполняет без знаковое ограничение (насыщение) результата сложения максимально возможным диапазоном 16-разрядных чисел без знака $0 \le x \le (2^{16}-1)=65535$, и записывает результат насыщения в младшее полуслово регистра назначения.

При этом результат каждой операции сложения/вычитания с насыщением будет в диапазоне: $0 \le x \le 65535$.
Графическая иллюстрация механизма выполнения этих команд представлена ниже:





Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Эти команды не изменяют состояния флагов условного выполнения команд (condition code flags). Флаг насыщения также не вырабатывается, даже если насыщение имеет место при выполнении одной или даже двух операций сложения и вычитания полуслов.

Примеры

```
UQASX R7, R4, R2      ; Сложение старшего полуслова первого
                      ; операнда в регистре R4 с младшим
                      ; полусловом второго операнда в регистре R2
                      ; Ограничение диапазоном целых 16-разрядных
                      ; чисел без знака
                      ; Сохранение результата в старшем полуслове
                      ; регистра назначения R7
                      ; Вычитание из младшего полуслова первого
                      ; операнда в регистре R4 старшего
                      ; полуслова второго операнда в регистре R2
                      ; Ограничение диапазоном целых 16-разрядных
                      ; чисел без знака
                      ; Сохранение результата в младшем полуслове
                      ; регистра назначения R7

UQSAX R0, R3, R5      ; Вычитание из старшего полуслова первого
                      ; операнда в регистре R3 младшего
                      ; полуслова второго операнда в регистре R5
                      ; Ограничение диапазоном целых 16-разрядных
                      ; чисел без знака
                      ; Сохранение результата в старшем полуслове
                      ; регистра назначения R0
                      ; Сложение младшего полуслова первого
                      ; операнда в регистре R3 со старшим
                      ; полусловом второго операнда в регистре R5
                      ; Ограничение диапазоном целых 16-разрядных
                      ; чисел без знака
                      ; Сохранение результата в младшем полуслове
                      ; регистра назначения R0
```

5.7 Знаковое удвоение второго операнда с насыщением и сложение/вычитание с первым операндом со знаковым насыщением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
QDADD and QDSUB	Saturating Double and Add, signed Знаковое удвоение с насыщением (ограничением) и сложение с насыщением Saturating Double and Subtract, signed Знаковое удвоение с насыщением (ограничением) и вычитание с насыщением		V	V

Синтаксис

```
op{cond} {Rd}, Rn, Rm
```

Где:		
	op оптокод один из: **QDADD** - Saturating Double and Add. Двойное насыщение после удвоения второго операнда-слова и сложения с первым. **QDSUB** - Saturating Double and Subtract. Двойное насыщение после удвоения второго операнда-слова и вычитания из первого.	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения.	
	Rn, Rm – Регистры, содержащие первый и второй операнды-слова	

Операция

Команда удвоения с насыщением и сложения с насыщением **QDADD**:
■ Удваивает значение второго операнда-слова с автоматическим «знаковым ограничением».
■ Добавляет полученное значение к первому операнду и вновь выполняет «знаковое ограничение» результата.
■ Сохраняет результат в регистре назначения.
Команда удвоения с насыщением и вычитания с насыщением **QDSUB**:
■ Удваивает значение второго операнда-слова с автоматическим «знаковым ограничением».
■ Вычитает из значения первого операнда удвоенное и ограниченное значение второго операнда и вновь выполняет «знаковое ограничение» результата.
■ Сохраняет результат в регистре назначения.

Все операции знакового удвоения, сложения и вычитания сопровождаются операцией знакового ограничения результата максимально возможным диапазоном 32-разрядных чисел со знаком $-2^{31} \le x \le 2^{31} - 1$ т.е. (-2 147 483 648 ≤ x ≤ +2 147 483 647).

Если в результате какой-либо из этих операций происходит насыщение, то флаг насыщения Q в регистре слова состояния APSR устанавливается.

Ограничения

■ В команде нельзя использовать указатель стека SP и счетчик команд PC.

Выработка флагов

Если при выполнении любой из операций удвоения или сложения/вычитания происходит насыщение, то флаг насыщения Q устанавливается.

Примеры

```
QDADD R7, R4, R2      ; Удваивает значение второго операнда в
                      ; регистре R2, выполняет «знаковое
                      ; ограничение результата
                      ; Добавляет полученное значение к первому
                      ; операнду в R4, выполняет «знаковое
                      ; ограничение результата» и
                      ; сохраняет его в регистре назначения R7

QDSUB R0, R3, R5      ; Удваивает значение второго операнда в
                      ; регистре R5, выполняет «знаковое
                      ; ограничение результата
                      ; Вычитает полученное значение из первого
                      ; операнда в R3, выполняет «знаковое
                      ; ограничение результата» и
                      ; сохраняет его в регистре назначения R0
```

6 КОМАНДЫ УПАКОВКИ И РАСПАКОВКИ ДАННЫХ

Оглавление

6.1. Упаковка двух полуслов в слово 110

6.2. Распаковка байта/полуслова со знаком/без знака с расширением до формата слова.....112

6.3. Одновременная распаковка двух байт со знаком/без знака с расширением до формата полуслова 114

6.4. Расширение байта/полуслова со знаком/без знака с последующими сложением 116

6.1 Упаковка двух полуслов в слово

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
PKHBT и PKHTB	Pack Halfword. Упаковка двух полуслов в слово	v	v	v

Синтаксис

```
op{cond} {Rd}, Rn, Rm {, LSL #imm}
op{cond} {Rd}, Rn, Rm {, ASR #imm}
```

Значение символов в мнемокоде операции упаковки двух полуслов в слово		
PK	**H**	**BT/TB**
		BT – Bottom-Top – Сначала младшее полуслово из регистра Rn, затем старше полуслово из регистра Rm после опционального, логического сдвига влево на заданное число разрядов. **TB** – Top-Bottom – Сначала старшее полу-слово из регистра Rn, затем младшее полуслово из регистре Rm после опционального арифметического сдвига вправо на заданное число разрядов.
	H – Halfword - Полуслово	
PK – Pack – Упаковать в слово		

Где:

cond
код условного выполнения команды (опция)

Rd – Регистр назначения

Rn – Регистр-источник первого операнда - полуслова

Rm – Регистр-источник второго операнда - полуслова, значение которого может быть опционально сдвинуто на заданное число разрядов

imm – число разрядов сдвига второго операнда. Тип операции сдвига зависит от типа операции:

Для команды **PKHBT**	**LSL** – логический сдвиг влево на число разрядов от 1 до 31. Если указано 0, то – нет сдвига.
Для команды **PKHTB**	**ASR** – арифметический сдвиг вправо на число разрядов от 1 до 32. Если указано 32, то – результат сдвига будет нулевым.

Операция

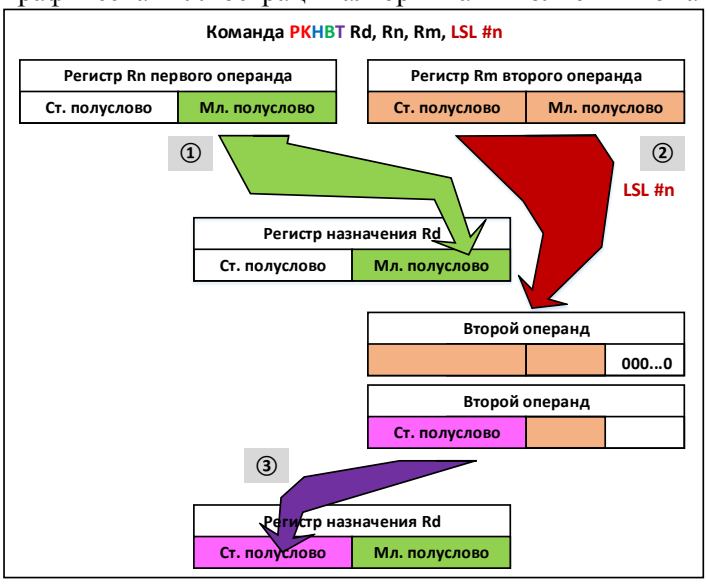
Команда **PKHBT**:

1. Извлекает младшее полуслово (bottom halfword) из первого операнда Rn и сохраняет его в младшем полуслове регистра назначения Rd.
2. Извлекает второй операнд Rm, опционально сдвигает его влево на заданное число разрядов и сохраняет старшее полуслово (top halfword) в старшем полуслове регистра назначения Rd.

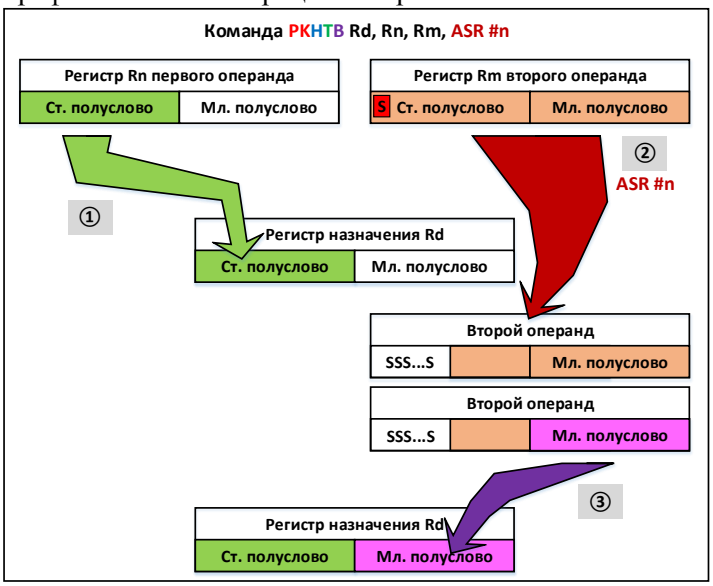
Команда **PKHTB**:

1. Извлекает старшее полуслово (top halfword) первого операнда Rn и сохраняет его в старшем полуслове регистра назначения Rd.
2. Извлекает второй операнд Rm, опционально сдвигает его вправо на заданное число разрядов и сохраняет младшее полуслово (bottom halfword) в младшем полуслове регистра назначения Rd.

Графическая иллюстрация алгоритма выполнения команды PKHBT:



Графическая иллюстрация алгоритма выполнения команды PKHTB:



Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
PKHBT R3, R4, R5 LSL #0     ; Записывает младшее полуслово R4
                            ; в младшее полуслово R3
                            ; Записывает старше полуслово R5 без
                            ; сдвига влево в старшее полуслово R3

PKHTB R4, R0, R2 ASR #1     ; Записывает старшее полуслово R0 в
                            ; старшее полуслово R4
                            ; Сдвигает арифметически вправо на
                            ; один разряд значение в регистре R2 и
                            ; сохраняет младшее полуслово в
                            ; младшем полуслове регистра R4
```

6.2 Распаковка байта/полуслова со знаком/без знака с расширением до формата слова

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SXT and UXT	Sign extend and Zero extend. Знаковое и без знаковое расширение.	v	v	v

Синтаксис

```
op{cond} {Rd,} Rm {, ROR #n}
```

Значение символов в мнемонике команды распаковки - расширения формата			
S/U	XT	B/H	
		B -byte - Расширить байт **H** - halfword - Расширить полуслово	
	XT - e**XT**end - Расширить формат операнда до полного слова		
S –Signed Extend – Расширение формата числа со знаком в дополнительном коде **U** – Unsigned Extend – Расширение формата числа без знака			

```
SXTB        Расширить байт со знаком до 32-разрядного слова
SXTH        Расширить полуслово со знаком до 32-разрядного слова
UXTB        Расширить байт без знака до 32-разрядного слова
UXTH        Расширить полуслово без знака до 32-разрядного слова
```

Где:

cond
код условного выполнения команды (опция)

Rd – Регистр назначения

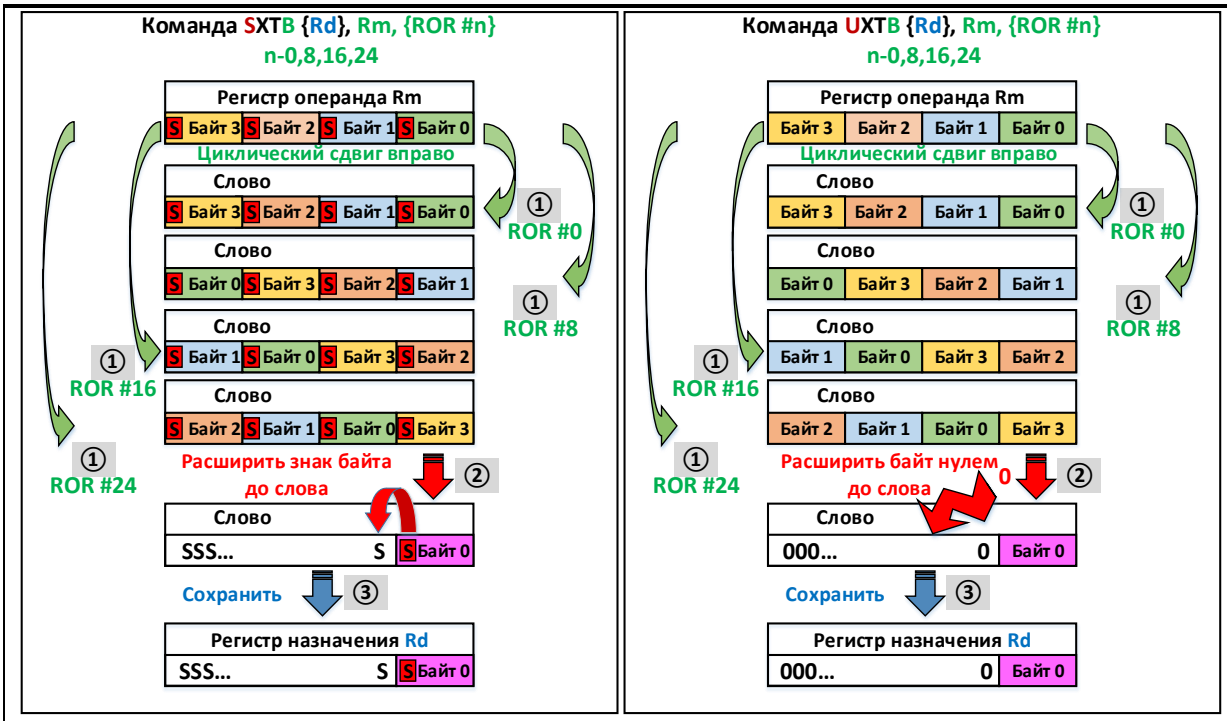
Rm – Регистр-источник, содержащий операнд, подлежащий расширению формата.

ROR #n – Попутная операция циклического сдвига вправо. Возможны только три варианта:
ROR #8 – на 8 бит (вправо на один байт);
ROR #16 – на 16 бит (вправо на два байта);
ROR #24 – на 24 бита (вправо на три байта);
Если опция сдвига отсутствует, то сдвиг не производится (эквивалентно **ROR #0**).

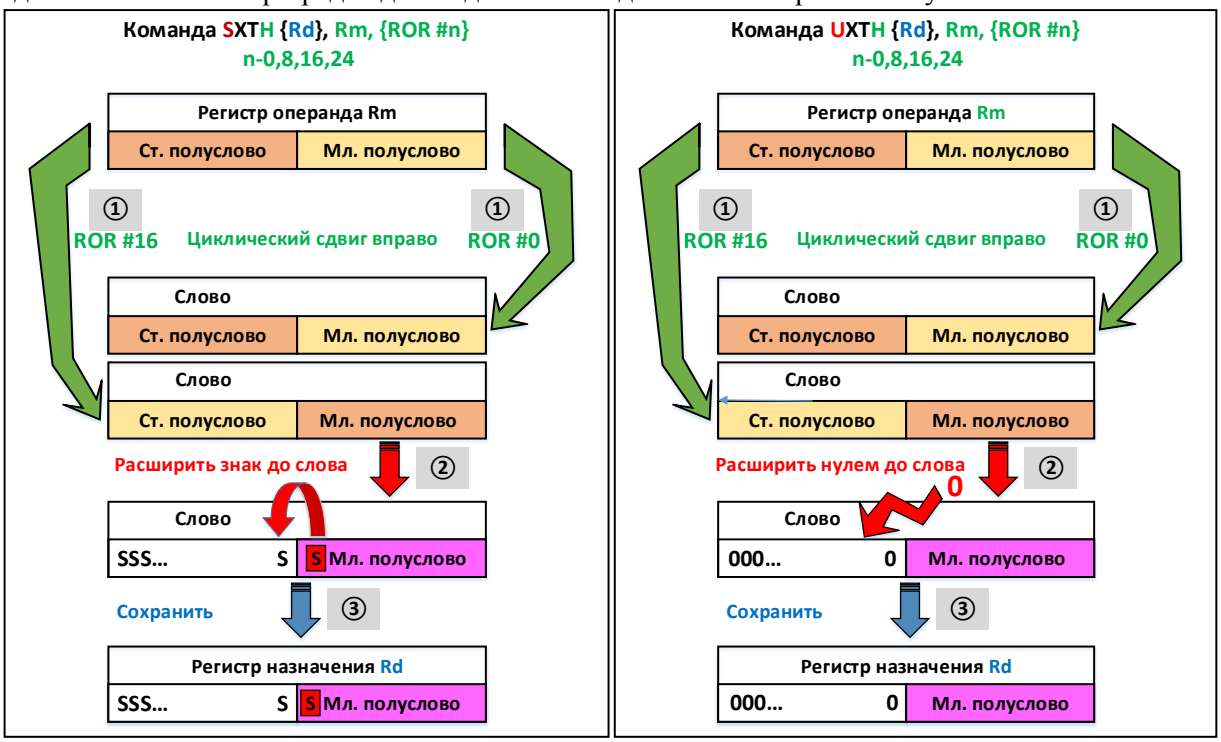
Операция

Выполняются следующие действия:
1. Исходное содержимое регистра-источника Rm циклически сдвигается вправо на 0, 8, 16 или 24 бита (на 0, 1, 2 или 3 байта).
2. Из слова выделяется младший байт (bits[7:0]) или младшее полуслово (bits[15:0]) и расширяется с учетом знака «S» (старшим знаковым разрядом) или для команд с префиксом «U» - как без знаковое (нулем):
- SXTB знаковое расширение младшего байта (bits[7:0]) до 32-х бит.
- UXTB без знаковое расширение младшего байта (bits[7:0]) до 32-х бит.
- SXTH знаковое расширение младшего полуслова (bits[15:0]) до 32-х бит.
- UXTH без знаковое расширение младшего полуслова (bits[15:0]) до 32-х бит.

Графическая иллюстрация алгоритма выполнения команд SXTB и UXTB показана ниже. Команды позволяют за счет циклического сдвига на число разрядов 0, 8, 16, 24 выделить любой из 4-х байтов исходного слова:



Графическая иллюстрация команд SXTH и UXTH показана ниже. Наиболее часто используется сдвиг на 0 или на 16 разрядов для выделения младшего или старшего полуслова:



Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов
Примеры

```
SXTH R4, R6, ROR #16  ; Циклически сдвинуть вправо на 16 бит
                      ; содержимое регистра R6, т.е. загрузить
                      ; старшее полуслово во младшее
                      ; Расширить до 32-х разрядов, как число со
                      ; знаком в дополнительном коде и сохранить
                      ; в регистре R4
UXTB R3, R10          ; Распаковать младший байт в регистре R10
                      ; расширить нулем до 32-х разрядов и
                      ; сохранить в регистре R3
```

6.3 Одновременная распаковка двух байт со знаком/без знака с расширением до формата полуслова

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SXTB16 and UXTB16	Sign extend and Zero extend byte 16 Знаковое и без знаковое расширение двух байт до 16-разрядных слов		v	v

Синтаксис

```
op{cond} {Rd,} Rm {, ROR #n}
```

Значение символов в мнемонике команды распаковки двух байт и расширения формата до полуслова			
S/U	XTB	16	
		Расширить оба младших байта полуслов регистра-источника до полуслов	
	XTB – e**XT**end **B**yte - Расширить формат байта		
S –Signed Extend – Расширение формата байта со знаком в дополнительном коде **U** – Unsigned Extend – Расширение формата байта без знака			

Оптокод op:

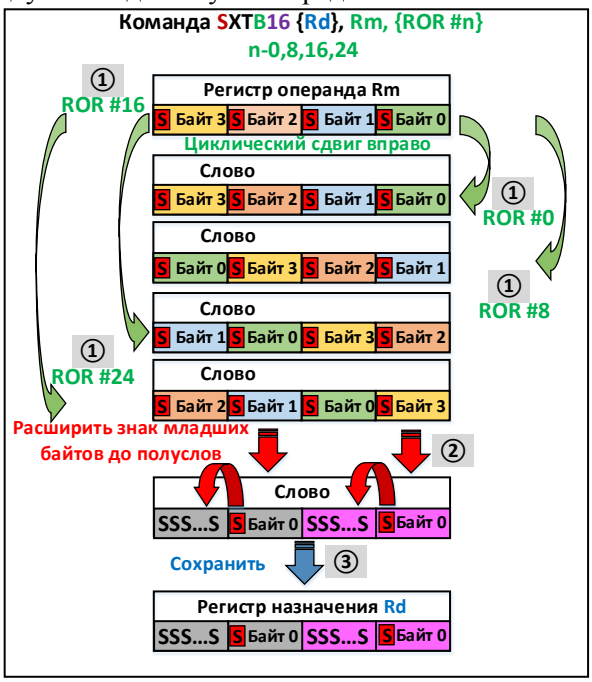
SXTB16	Расширить оба младших знаковых байта двух полуслов из регистра-источника Rm до полуслов
UXTB16	Расширить оба младших байта без знака из двух полуслов регистра-источника Rm до полуслов

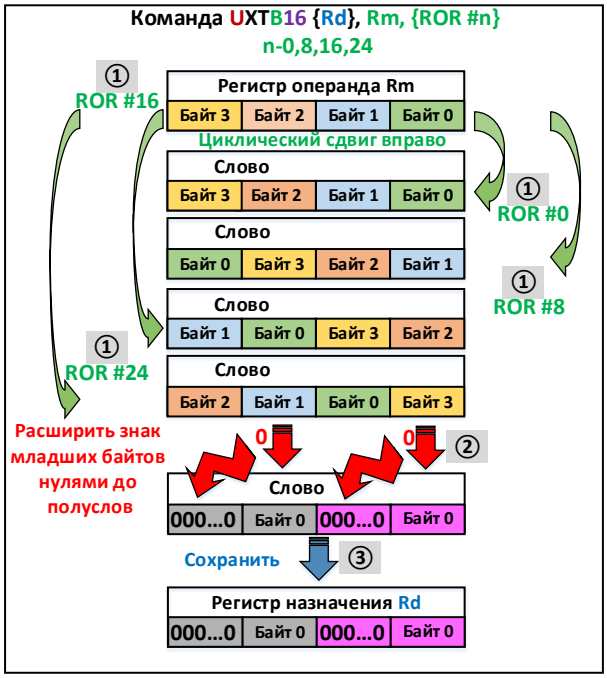
Где:	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения
	Rm – Регистр-источник, содержащий операнды, подлежащие расширению формата.
	ROR #n – Попутная операция циклического сдвига вправо. Возможны варианты: **ROR #8** – на 8 бит (вправо на один байт); **ROR #16** – на 16 бит (вправо на два байта); **ROR #24** – на 24 бита (вправо на три байта); Если опция сдвига отсутствует, то сдвиг не производится (эквивалентно **ROR #0**)

Операция

Выполняются следующие действия:
1. Исходное содержимое регистра-источника Rm циклически сдвигается вправо на 0, 8, 16 или 24 бита (на 0, 1, 2 или 3 байта).
2. Выделяются оба младших байта в каждом из двух полуслов и выполняется:
■ Для SXTB16 - знаковое расширение младшего байта младшего полуслова (bits[7:0]) и младшего байта старшего полуслова (bits [23:16]) до полуслов.
■ Для UXTB16 – без знаковое расширение младшего байта младшего полуслова (bits[7:0]) и младшего байта старшего полуслова (bits [23:16]) до полуслов.
3. Полученное слово сохраняется в регистре приемнике Rd.

Графическая иллюстрация команд SXTB16 и UXTB16, выполняющих одновременное расширение двух байт до полуслов представлена ниже:





Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
        SXTB16 R4, R6, ROR #16      ; Циклически сдвинуть вправо на 16 бит
                                    ; содержимое регистра R6, т.е.
                                    ; выполнить обмен полуслов
                                    ; Расширить младший байт из каждого
                                    ; полуслова до слова, считая байт
                                    ; числом со знаком.
```

6.4 Расширение байта/полуслова со знаком/без знака с последующими сложением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SXTA and UXTA	Signed and Unsigned Extend and Add. Знаковое и без знаковое расширение и дополнительное сложение		v	v

Синтаксис

```
op{cond} {Rd,} Rn, Rm {, ROR #n}
```

Значение символов в мнемонике команды распаковки байт/полуслов, расширения формата и дополнительной операции сложения

S/U	XT	A	B/H	16
				Расширить оба младших байта регистра-источника до полуслова
			B -byte - Расширить байт **H** - halfword - Расширить полуслово	
		После расширения выполнить дополнительную операцию сложения (Add – A)		
	XT - e**XT**end - Расширить формат			
S –Signed Extend – Расширение формата числа со знаком в дополнительном коде **U** – Unsigned Extend – Расширение формата числа без знака				

Оптокод op:

SXTAB	Расширить младший байт со знаком из Rm до 32-разрядного слова и сложить со словом в регистре Rn
SXTAH	Расширить младшее полуслово со знаком из Rm до 32-разрядного слова и сложить со словом в регистре Rn
SXTAB16	Расширить оба младших знаковых байта полуслов из регистра Rm до знаковых полуслов и сложить каждое из них с соответствующим полусловом в регистре Rn (двойное 16-битное сложение)
UXTAB	Расширить младший байт без знака из Rm до 32-разрядного слова и сложить со словом в регистре Rn
UXTAH	Расширить младшее полуслово без знака из Rm до 32-разрядного слова и сложить со словом в регистре Rn
UXTAB16	Расширить оба младших байта без знака из полуслов регистра Rm до без знаковых полуслов и сложить каждое из них с соответствующим полусловом в регистре Rn (двойное 16-битное сложение)

Где:

cond
код условного выполнения команды (опция)

Rd – Регистр назначения

Rn – Регистр-источник операнда/ов для операции сложения

Rm – Регистр-источник, содержащий операнд/ы, подлежащие расширению формата.

ROR #n – Попутная операция циклического сдвига вправо. Возможны варианты:
ROR #8 – на 8 бит (вправо на один байт);
ROR #16 – на 16 бит (вправо на два байта);
ROR #24 – на 24 бита (вправо на три байта);
Если опция сдвига отсутствует, то сдвиг не производится (эквивалентно **ROR #0**)

Операция

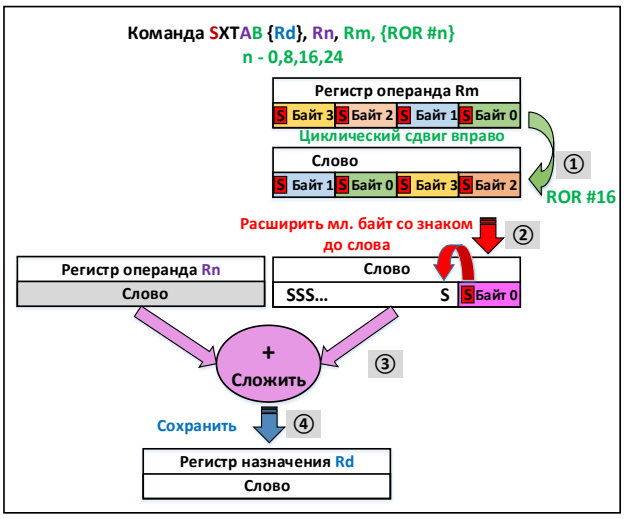
Выполняются следующие действия:

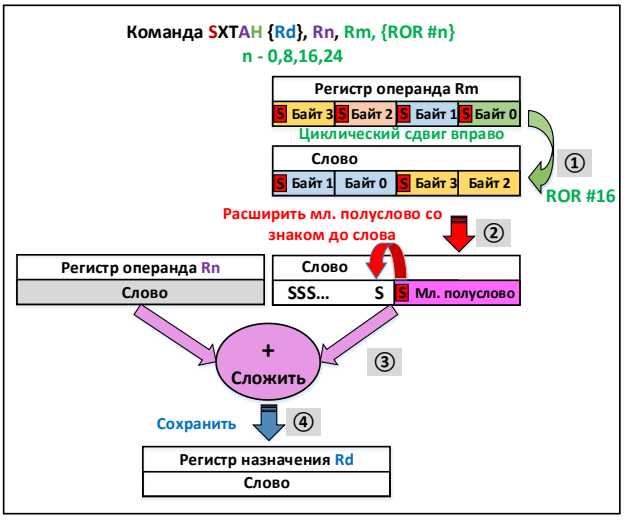
1. Исходное содержимое второго регистра-источника Rm циклически сдвигается вправо на 0, 8, 16 или 24 бита (на 0, 1, 2 или 3 байта). Это - операция распаковки нужного байта или полуслова.
2. Из слова выделяется младший байт (bits[7:0]) или младшее полуслово (bits[15:0]) и расширяется до слова с учетом знака (префикс «S») или как без знаковое нулем (префикс «U»):

■ SXTAB знаковое расширение младшего байта (bits[7:0]) до 32-х бит.
■ UXTAB без знаковое расширение младшего байта (bits[7:0]) до 32-х бит.
■ SXTAH знаковое расширение младшего полуслова (bits[15:0]) до 32-х бит.
■ UXTAH без знаковое расширение младшего полуслова (bits[15:0]) до 32-х бит.
■ SXTAB16 знаковое расширение младшего байта младшего полуслова (bits[7:0]) и младшего байта старшего полуслова (bits [23:16]) до 16-разрядных полуслов.
■ UXTAB16 без знаковое расширение младшего байта младшего полуслова (bits[7:0]) и младшего байта старшего полуслова (bits [23:16]) до 16-разрядных полуслов.

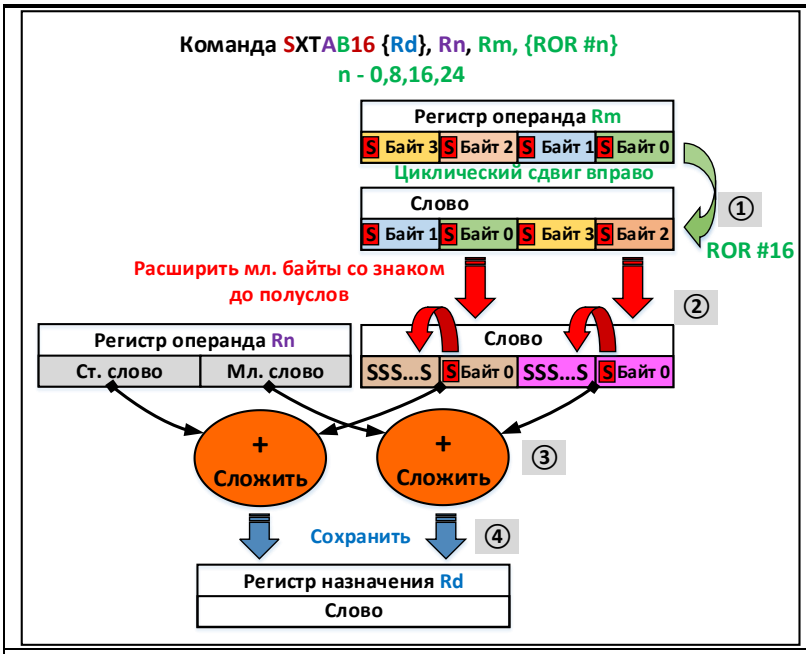
3. Для первых 4-х команд расширенные до полного 32-разрядного слова значения складываются со словом в первом регистре-источнике Rn, результат сохраняется в регистре приемнике Rd.
4. Для двух последних команд с суффиксом «16» выполняется 16-битное сложение двух пар полуслов: младшего полученного, с младшим из регистра Rn; старшего полученного, со старшим из регистра Rn. Это как бы команды двойного сложения полуслов. Так как состояние флагов не меняется, то сложения полуслов выполняются без контроля переполнений.

Графическая иллюстрация алгоритма выполнения команд SXTAB и SXTAH представлена ниже для одного из 4-х возможных вариантов сдвига исходного операнда (на 16 бит):





Графическая иллюстрация команды одновременного расширения двух байт со знаком до полуслов и двойного сложения полуслов SXTAB16 дана ниже на примере сдвига вправо на 16 разрядов:



Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
SXTAH R4, R8, R6, ROR #16  ; Циклически сдвинуть содержимое  R6
                           ; вправо на 16 бит (выделить старшее
                           ; полуслово в младшей части слова)
                           ; расширить младшее полуслово
                           ; со знаковом до 32-разрядного слова и
                           ; сложить с содержимым регистра R8
                           ; сохранить в регистре R4
UXTAB R3, R4, R10          ; Расширить младший байт в регистре
                           ; R10 как число без знака до слова,
                           ; сложить с содержимым регистра R4 и
                           ; сохранить результат в регистре R3
```

7 КОМАНДЫ РАБОТЫ С БИТОВЫМИ ПОЛЯМИ

Оглавление

7.1. Очистка и вставка битового поля 119

7.2. Расширение битового поля до слова 120

7.1 Очистка и вставка битового поля

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
BFC и BFI	Bit Field Clear and Bit Field Insert. Очистить битовое поле. Вставить битовое поле.	V	V	V

Синтаксис

```
BFC{cond} Rd, #lsb, #width
BFI{cond} Rd, Rn, #lsb, #width
```

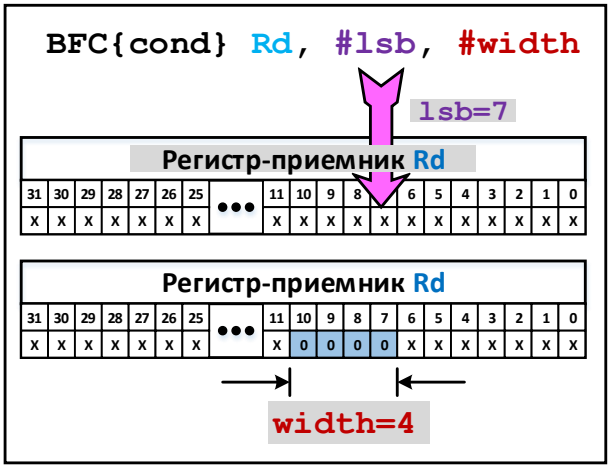
Где:	
	cond код условного выполнения команды (опция)
	Rd – Регистр назначения
	Rn – Регистр-источник
	#lsb – Позиция младшего значащего бита (least-significant bit)битового поля (bitfield). Целое число в диапазоне от 0 до 31.
	#width – ширина битового поля в числе двоичных разрядов. Целое число в диапазоне от 1 до 32.

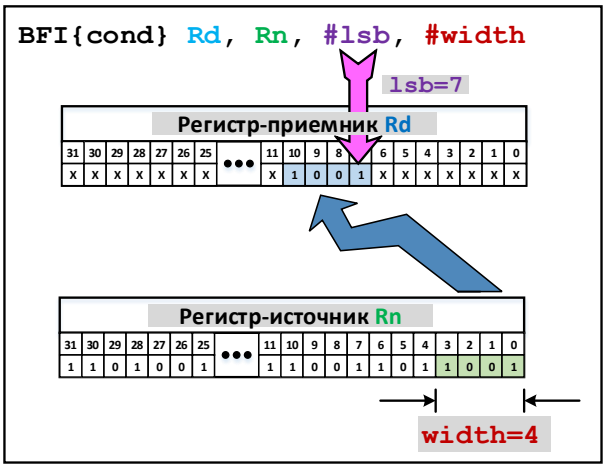
Операция

Команда **BFC**:
Очищает битовое поле в регистре назначения Rd, начиная с указанного в команде младшего значащего бита lsb шириной width бит. Все остальные биты регистра назначения остаются неизменными.

Команда **BFI**:
Копирует битовое поле из регистра источника Rn, начиная с нулевого разряда (bit[0]) заданной длины width в регистр приемник Rd, начиная с указанного в команде младшего значащего бита lsb. Все остальные биты регистра назначения остаются неизменными.

Графическая иллюстрация алгоритма выполнения этих команд представлена ниже:





Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
BFC R4, #8, #12        ; Очистить 12 бит в регистре R4, начиная
                       ; с 8-го бита (биты 8 – 19)
BFI R9, R2, #8, #12    ; Вставить 12 начальных бит из регистра
                       ; источника R2 (биты 0-11)в регистр R9,
                       ; начиная с 8-го бита (заменить биты 8 – 19)
```

7.2 Расширение битового поля до слова

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SBFX и UBFX	Signed Bit Field Extract and Unsigned Bit Field Extrac Выделить и расширить битовое поле как число со знаком Выделить и расширить битовое поле как число без знака	V	V	V

Синтаксис

```
SBFX{cond} Rd, Rn, #lsb, #width
UBFX{cond} Rd, Rn, #lsb, #width
```

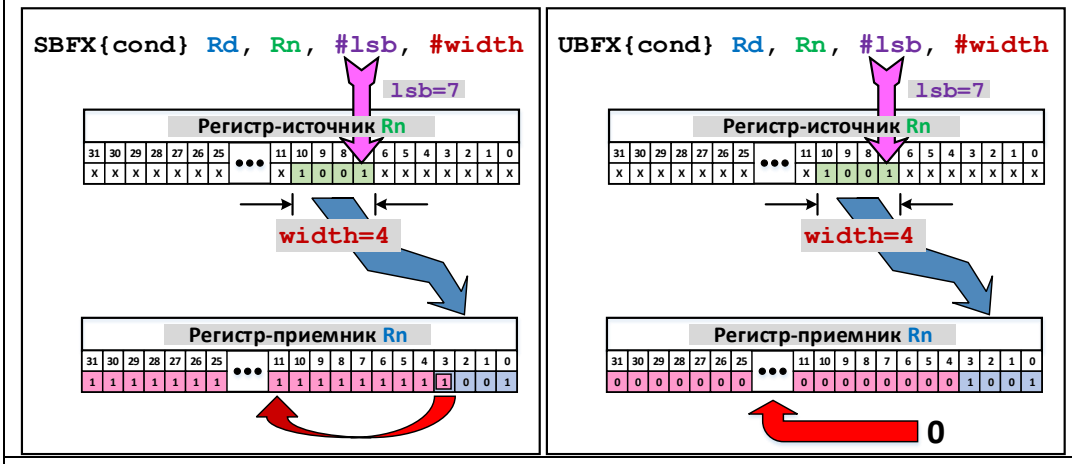
Где:	**cond** код условного выполнения команды (опция)
	Rd – Регистр назначения
	Rn – Регистр-источник
	#lsb – Позиция младшего значащего бита (least-significant bit) выделяемого битового поля (bitfield). Целое число в диапазоне от 0 до 31.
	#width – ширина битового поля в числе двоичных разрядов. Целое число в диапазоне от 1 до 32.

Операция

Команда **SBFX**:
Извлекает битовое поле шириной width бит из регистра источника Rn, начиная с указанного в команде младшего значащего бита lsb, расширяет его с учетом знака до 32-х разрядов и сохраняет результат в регистре назначения Rd.

Команда **UBFX**:
Извлекает битовое поле шириной width бит из регистра источника Rn, начиная с указанного в команде младшего значащего бита lsb, расширяет его как число без знака до 32-х разрядов и сохраняет результат в регистре назначения Rd.

Графическая иллюстрация алгоритма выполнения этих команд представлена ниже:



Ограничения

Регистр назначения Rd не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
        SBFX R0, R1, #20, #4        ; Извлечь 4-е бита с 20-го по 23-й из
                                    ; регистра источника R1, расширить до
                                    ; 32-х бит с учетом знака и сохранить
                                    ; результат в регистре приемнике R0
        UBFX R8, R11, #9, #10       ; Извлечь 10-ь бит с 9-го по 18-ый из
                                    ; регистра источника R11, расширить до
                                    ; 32-х разрядов нулями и сохранить
                                    ; в регистре приемнике R8.
```

8 КОМАНДЫ СОПРОЦЕССОРА

Оглавление

8.1. Загрузка регистров сопроцессора непосредственными данными 123
8.2. Загрузка регистров сопроцессора данными из памяти 124
8.3. Сохранение содержимого регистров сопроцессора в памяти 125
8.4. Множественная загрузка/сохранение регистров 126
8.5. Пересылка данных между регистрами сопроцессора 130
8.6. Пересылка данных между регистрами ЦПУ и сопроцессора 131
8.7. Двойная пересылка данных между регистрами ЦПУ и сопроцессора 132
8.8. Копирование слова из регистра двойной точности в регистр ЦПУ или из регистра ЦПУ в одно из слов регистра двойной точности 133
8.9. Обмен данными между регистром специального назначения сопроцессора и регистрами ЦПУ 135
8.10. Множественное сохранение/восстановление содержимого регистров сопроцессора в стеке/из стека 137
8.11. Обработка одного операнда 139
8.12. Арифметические операции с двумя операндами 142
8.13. Стандартное умножение с положительным и отрицательным накоплением 146
8.14. Стандартное умножение, умножение с положительным или отрицательным накоплением и инверсией результата 148
8.15. Совмещенное умножение с положительным и отрицательным накоплением 150
8.16. Совмещенное умножение с положительным и отрицательным накоплением и инверсией результата 152
8.17. Сравнение операндов с плавающей точкой 154
8.18. Конвертация между целыми числами и числами в формате с плавающей точкой 155
8.19. Конвертация между числами в форматах с фиксированной и с плавающей точкой 157
8.20. Преобразование чисел с плавающей точкой половинной точности в формат однократной точности и обратно 160

8.1 Загрузка регистров сопроцессора непосредственными данными

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	Move floating-point Immediate Загрузка непосредственного числа с плавающей точкой			V

Синтаксис

```
VMOV{cond}.F32 Sd, #imm     ; Загрузка непосредственного операнда с
                            ; плавающей точкой в регистр модуля
                            ; плавающей точки
```

Где:

cond
код условного выполнения команды (опция)

Sd - Регистр назначения, один из S0-S31

imm - Непосредственный операнд – константа в формате с плавающей точкой.
Константа может быть любым числом, которое может быть вычислено как +/– ($n * 2^{-r}$), где n и r целые числа в диапазонах
(16 <= n <= 31), (0 <= r <= 7).
Это означает, что загрузить можно только ограниченный ряд чисел, например, при n=16: ±16.0; ±8.0; ±4.0; … ±0.5; ±0.25; ±0.125
При n=31: ±15.5; ±7.75; ±3.875; ±1.9375; ±0.96875; …

Операция

Команда выполняет загрузку константы в регистр с плавающей точкой.

Ограничения

Нет ограничений

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMOV.F32 S0,#-4.0    ; Загрузка в регистр s0 числа -4.0
```

8.2 Загрузка регистров сопроцессора данными из памяти

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VLDR	Loads a single extension register from memory Загрузка одного регистра расширения - модуля плавающей точки из памяти			V

Синтаксис

```
VLDR{cond}{.32} Sd, [Rn {,#imm}]
VLDR{cond}{.32} Sd, [PC, #imm]
VLDR{cond}.F32 Sd, label          ; Псевдо-команда
VLDR{cond}{.64} Dd, [Rn {,#imm}]
VLDR{cond}{.64} Dd, [PC, #imm}]
VLDR{cond}.F64 Dd, label          ; Псевдо-команда
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **size** – Спецификатор длины данных в битах, загружаемых в регистры расширения – или 32 или 64 (опция)

32	Загружается слово в регистр однократной точности **S**
64	Загружается двойное слово в регистр двукратной точности **D**

- **Sd** – Регистр назначения при загрузке слова
- **Dd** – Регистр назначения при загрузке двойного слова
- **Rn** – Базовый регистр. Содержит базовый адрес области памяти, из которой будет выполняться загрузка данных. Может использоваться указатель стека SP, если данные для загрузки извлекаются из стека.
- **imm** – положительное (**+**) или отрицательное (**-**) смещение адреса, расположенного в регистре базового адреса.
  Разрешенные значения смещений определяются как:
  любое число в диапазоне от 0 до ±1020, кратное 4 (размеру слова)
- **PC** – текущее содержимое счетчика команд
- **label** – метка ячейки памяти (или адрес литеральных данных), из которой берутся данные для загрузки

Операция

1. Команда первого типа выполняет загрузку одного регистра модуля плавающей точки (однократной или двукратной точности) из ячейки памяти, адрес которой специфицируется содержимым базового регистра Rn и опционально заданным смещением.
2. Команда второго типа выполняет загрузку регистра модуля плавающей точки из ячейки памяти по адресу, равному текущему содержимому счетчика команд PC плюс смещение (относительная адресация по счетчику команд PC).
3. Команда третьего типа выполняет загрузку регистра сопроцессора из ячейки памяти, помеченной меткой. Фактически это псевдокоманда, которая автоматически заменяется транслятором на команду относительной адресации по счетчику команд PC плюс смещение.

Замечание.
В Ассемблере имеются еще две псевдо-команды **VLDR.F32 Sd,=Constant** и **VLDR.F64 Dd,=Constant**, которые позволяют загрузить регистры модуля FPU любыми начальными данными. В этом случае автоматически генерируется так называемый «литеральный» пул, который содержит нужную константу в формате с плавающей точкой и команда загрузки с относительной адресацией по текущему состоянию счетчика команд (п. 2). Константы по умолчанию располагаются в конце текущей кодовой секции.

Ограничения

Нет ограничений

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
      VLDR S0, [R0,#8]       ; Загрузить регистр однократной точности S0
                             ; из памяти по базовому адресу в регистре
                             ; ЦПУ R0 и смещению +8.
      VLDR D2, F_33          ; Загрузить регистр сопроцессора D2
                             ; константой с плавающей точкой двойной
                             ; точности, расположенной по адресу F_33
      VLDR.F32 s0,=-4.3333   ; Псевдо-команда загрузки числа с плавающей
                             ; точкой однократной точности
; Эта псевдо-команда автоматически заменяется транслятором командой
; VLDR.F32, s0, [PC, #imm]
; с относительной адресацией по текущему значению счетчика команд PC
; до места расположения константы в кодовой памяти (литерального пула)
; Константа также генерируется автоматически
```

8.3 Сохранение содержимого регистров сопроцессора в памяти

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VSTR	Floating-point Store Сохранение содержимого одного регистра сопроцессора в памяти			V

Синтаксис

```
VSTR{cond}{.32} Sd, [Rn {,#imm}]      ; Сохранить содержимое регистра
                                      ; Sd в памяти по базовому адресу
                                      ; в Rn и смещению imm
VSTR{cond}{.64} Dd, [Rn {,#imm}]      ; Сохранить содержимое регистра
                                      ; Dd в памяти по базовому адресу
                                      ; в Rn и смещению imm
```

Где:

cond – Код условного выполнения команды (опция)

size – Спецификатор длины данных в битах, сохраняемых из регистра сопроцессора в памяти(опция)

32	Сохраняется слово из регистра однократной точности **S**
64	Сохраняется двойное слово из регистра двукратной точности **D** (два слова последовательно)

Sd – Регистр однократной точности – источник данных

Dd – Регистр двукратной точности – источник данных

Rn – Базовый регистр. Может использоваться указатель стека SP. Содержит базовый адрес области памяти, в которой будут сохраняться данные.

imm – положительное (**+**) или отрицательное (**-**) смещение базового адреса, расположенного в базовом регистре Rn.
Разрешенные значения смещений: любые числа в диапазоне от 0 до ±1020, кратные 4. Если смещение опущено, - оно считается равным нулю offset=0.

Операция

Команда выполняет сохранение содержимого одного регистра модуля плавающей точки в ячейке памяти, адрес которой специфицируется содержимым регистра базового адреса Rn и опционально заданным смещением.

Замечание.

Команды сохранения содержимого регистров в памяти – это единственные команды, в которых на месте первого операнда указывается имя не приемника данных, а источника. Это связано с тем, что сохранение выполняется в режиме косвенной адресации по содержимому базового регистра и смещению. По умолчанию операнды с косвенной адресацией указываются в конце команды.

Ограничения

Использование счетчика команд PC в качестве базового регистра Rn запрещается – не поддерживается сохранение содержимого регистров сопроцессора с использованием относительной адресации по счетчику команд.

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VSTR S0, [R0,#+4]      ; Сохранить содержимое регистра однократной
                       ; точности S0 в памяти по базовому адресу в
                       ; регистре ЦПУ R0 и смещению +4.
VSTR D2, [R1]          ; Сохранить содержимое регистра двукратной
                       ; точности D2 в памяти по адресу,
                       ; расположенному в регистре R1
                       ; (Классическая косвенная адресация)
                       ; Сохраняются два слова -64 бита
```

8.4 Множественная загрузка/сохранение регистров

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VLDM	Floating-point Load Multiple Множественная загрузка регистров сопроцессора из памяти			V

Синтаксис

```
VLDM{mode}{cond}{.size} Rn{!}, list   ; Загрузка регистров расширения,
                                      ; указанных в листинге
```

Где:

cond – Код условного выполнения команды (опция)

mode – Режим адресации:

IA	**Increment After.** «Инкрементирование адреса после доступа». Последовательные адреса начинаются с адреса, специфицированного в регистре базы Rn. После очередной загрузки данных из памяти адрес ячейки памяти автоматически инкрементируется. Это режим адресации «по умолчанию». Суффикс «IA» в мнемокоде команды для него может быть опущен.
DB	**Decrement Before.** «Декрементирование адреса перед доступом». Последовательные адреса заканчиваются как раз перед адресом, специфицированным в регистре базы Rn. Перед каждым доступом к памяти адрес автоматически декрементируется.

Size – Спецификатор длины данных (опция) в битах, загружаемых в регистры расширения – или 32 или 64. Если указан, то должен

	соответствовать типу регистров в списке: **32** – для регистров однократной точности **S, 64** – для регистров двойной точности **D.**
	Rn – Базовый регистр. Может использоваться указатель стека SP. Содержит базовый адрес области памяти, из которой будет осуществляться последовательная загрузка данных.
	! – Суффикс обратной перезаписи (авто-модификации базового регистра после множественной пересылки данных), который предписывает записать последний модифицированный адрес памяти назад в базовый регистр Rn – сделать обновление базового регистра для возможного последующего доступа к памяти (обращения к очередному блоку памяти). Опция авто-обновления может указываться для режима адресации «Декрементирование перед» **== DB,** для режима адресации «Инкрементирование после» **== IA –** авто-обновление делается автоматически и «!» можно опустить.
	list – листинг регистров расширения (**extension registers)** для загрузки данных, а именно, листинг регистров двойных слов Dn или одиночных слов Sn **(doubleword or singleword registers),** отделенных друг от друга запятой и окруженных фигурными скобками. Загрузка данных начинается в регистр с максимальным номером и выполняется в порядке убывания номеров регистров. Последним загружается регистр с наименьшим номером.

Операция

Команда выполняет загрузку нескольких регистров расширения из последовательной области памяти, адрес которой специфицируется содержимым регистра базового адреса Rn – одним из регистров ARM-ядра.

Замечание.
Ассемблер поддерживает две **псевдо-команды** загрузки регистров сопроцессора с пост-инкрементной и пре-декрементной адресацией:

```
VLDR{cond}{.size} Fd, [Rn], #offset       ; post-increment
VLDR{cond}{.size} Fd, [Rn, #-offset]!     ; pre-decrement
```

В качестве регистра назначения Fd можно указывать любой регистр однократной Sd или двукратной точности Dd. В качестве смещения offset – число, равное 4 для пересылки данных в регистр Sd и 8 – для пересылки данных в регистр Dd. В режиме пост-инкрементирования содержимое базового регистра автоматически обновляется, в режиме пре-декрементирования, - только при указании суффикса обновления «!». Транслятор заменяет эти псевдокоманды командами VLDM.

Ограничения

■ Если опция длины данных указана, то это должна быть длина данных в битах (32 или 64) для регистров расширения, заданных в листинге.
■ Указатель стека SP может использоваться в качестве регистра базового адреса, следовательно, данные могут загружаться из стека. Для всего набора команд ARM принято, что если суффикс {!} обратной перезаписи базового регистра не специфицирован, то в качестве базового адреса можно использовать и счетчик команд PC. В этом случае загрузка данных выполняется из кодовой памяти.
■ Листинг регистров расширения для множественной загрузки данных должен содержать по крайней мере один регистр. Если он содержит имена регистров двойной точности (64-разрядных), то их не должно быть больше чем 16.
■ Если используется режим адресации «Декрементирование перед доступом к данным» (Decrement Before), то к имени регистра базового адреса должна быть добавлена опция авто-перезаписи {!}, если далее предполагается работа с очередным блоком данных.

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VLDM R1,{S0,S1,S3}    ; Загрузка трех регистров сопроцессора S0-S3
                      ; из памяти с начальным адресом в R1
                      ; с авто-модификацией адреса в R1 после
                      ; выполнения всех пересылок
VLDR S1, [R7],#+4     ; Псевдо-команда загрузки регистра S1 из
                      ; памяти по косвенному адресу в R7 с
                      ; пост-инкрементной адресацией (+4)
                      ; Заменяется командой VLDM R1,{S1}
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VSTM	Floating-point Store Multiple Множественное сохранение содержимого регистров с плавающей точкой в памяти			V

Синтаксис

```
VSTM{mode}{cond}{.size} Rn{!}, list   ; Сохранение содержимого
                                      ; регистров расширения,
                                      ; указанных в листинге
```

Где:

cond – Код условного выполнения команды (опция)

mode – Режим адресации:

IA	**Increment After.** «Инкрементирование адреса после доступа». Последовательные адреса начинаются с адреса, специфицированного в регистре базы Rn. После очередного сохранения данных в памяти адрес ячейки памяти автоматически инкрементируется. Это режим адресации «по умолчанию» – суффикс **IA** в мнемокоде команды может быть опущен.
DB	**Decrement Before.** «Декрементирование адреса перед доступом». Последовательные адреса заканчиваются как раз перед адресом, специфицированным в регистре базы Rn. Перед каждым доступом к памяти адрес автоматически декрементируется.

Size – Спецификатор длины данных (опция) в битах, сохраняемых из регистров сопроцессора в памяти – или 32 или 64. Если указан, то должен соответствовать типу регистров в листинге – **32** для регистров однократной точности **S** и **64** – для регистров друкратной точности **D**.

Rn – Базовый регистр. Может использоваться указатель стека SP. Содержит базовый адрес области памяти, в которую будет осуществляться последовательная загрузка данных.

! – Суффикс обратной перезаписи (авто-модификации базового регистра после множественной пересылки данных), который предписывает записать последний адрес модифицированной памяти назад в базовый регистр Rn – сделать обновление базового регистра для возможного последующего доступа к памяти (обращения к следующему блоку памяти).
Команда авто-обновления может указываться для режима адресации «Декрементирование перед» **== DB**, для режима адресации «Инкрементирование после» **== IA –** авто-обновление делается автоматически и суффикс «!» можно опустить.

list – листинг регистров расширения (**extension registers)** для сохранения данных, а именно, регистров двойных слов **Dn** или

	одиночных слов **Sn (doubleword or singleword registers)**, отделенных друг от друга запятой и окруженных фигурными скобками. Сохранение данных выполняется, начиная с регистра сопроцессора с наименьшим номером в порядке возрастания номеров, и заканчивается сохранением содержимого регистра с максимальным номером.

Операция

Команда выполняет сохранение содержимого нескольких регистров расширения в последовательной области памяти, начальный адрес которой специфицируется содержимым регистра базового адреса Rn – одним из регистров ARM-ядра.
Команды **VPUSH List** и **VSTMDB SP!, List** эквивалентны, так как в процессоре используется пре-декрементный стек при записи данных с авто-обновлением указателя стека при каждой записи. В дизассемблере обе команды показываются как VPUSH.

Замечание.
В Ассемблере имеются псевдо-команды сохранения содержимого одного регистра сопроцессора в памяти с использованием пост-инкрементной и пре-декрементной адресации:

```
VSTR{cond}{.size} Fd, [Rn], #offset    ; Пост-инкрементная
VSTR{cond}{.size} Fd, [Rn, #-offset]! ; Пре-декрементная
```

В качестве регистра-источника данных Fd можно использовать регистр однократной Sd или двукратной точности Dd. В первом случае смещение должно быть равно 4, а во втором – 8. В режиме пост-инкрементной адресации содержимое базового регистра обновляется автоматически, в режиме пре-декрементной адресации – только при наличии суффикса «!».
Эти команды транслируются в команды VSTM.

Ограничения

■ Использование счетчика команд PC в качестве регистра базового адреса Rn запрещается.
■ Листинг регистров расширения для множественного сохранения данных должен содержать по крайней мере один регистр. Если он содержит имена регистров двойной точности (64-разрядных), то их не должно быть больше чем 16.

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
        VSTMDB R1!,{S0,S1,S3}      ; Сохранение содержимого трех
                                   ; регистров сопроцессора S0-S3
                                   ; в памяти с начальным адресом в R1
                                   ; с авто-декрементированием указателя
                                   ; перед каждой записью в память и
                                   ; авто-модификацией базового регистра
                                   ; R1 после всех записей
```

8.5 Пересылка данных между регистрами сопроцессора

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	VMOV Register Copies the contents of one register to another Копирование содержимого одного регистра сопроцессора в другой			V

Синтаксис

```
VMOV{cond}.F32 Sd, Sm        ; Копирование слова из регистра
                             ; источника Sm в регистр приемник Sd
VMOV{cond}.F64 Dd, Dm        ; Копирование двойного слова из регистра
                             ; источника Dm в регистр приемник Dd
```

Где:	
	cond - Код условного выполнения команды (опция)
	Sm - Регистр-источник для операций пересылки слова
	Sd - Регистр назначения для операций пересылки слова
	Dm - Регистр-источник для операций пересылки двойного слова
	Dd - Регистр назначения для операций пересылки двойного слова

Операция

Копирование содержимое одного регистра модуля плавающей точки в другой. Возможно копирование данных, как из регистров однократной точности, так и из регистров двойной точности.

Ограничения

Нет ограничений

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMOV S0, S5      ; Пересылка содержимого регистра однократной
                 ; точности S5 в регистр S0
VMOV D3, D1      ; Пересылка содержимого регистра двукратной
                 ; точности D1 в регистр D3
```

8.6 Пересылка данных между регистрами ЦПУ и сопроцессора

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	Transfers a single-precision register to and from an ARM core register Пересылка данных из регистра ARM-ядра в регистр однократной точности модуля плавающей точки и обратно			V

Синтаксис

```
VMOV{cond} Sd, Rm      ; Копирование данных из регистра ARM-ядра Rm в
                       ; регистр модуля плавающей точки Sd
VMOV{cond} Rd, Sm      ; Копирование данных из регистра модуля плавающей
                       ; точки Sm в регистр ARM-ядра Rd
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **R** – Регистр-ARM ядра. В первом случае - источник данных Rm, во втором случае – приемник данных Rd.
- **S** – Регистр модуля плавающей точки однократной точности. В первом случае приемник данных Sd, во втором – источник Sm.

Операция

Копирование одного слова из регистра ARM-ядра в регистр модуля плавающей точки или наоборот, из регистра плавающей точки однократной точности в регистр ARM-ядра.

Ограничения

Регистр ARM-ядра не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
        VMOV S0, R3     ; Копирование слова из регистра ЦПУ R3 в регистр
                        ; сопроцессора однократной точности S0
        VMOV R6, S2     ; Копирование слова из регистра сопроцессора
                        ; однократной точности S2 в регистр ЦПУ R6
```

8.7 Двойная пересылка данных между регистрами ЦПУ и сопроцессора

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	Transfers two consecutively numbered single-precision registers to and from two ARM core registers. Пересылка содержимого двух регистров модуля плавающей точки с последовательными номерами в два любых регистра ARM-ядра или в обратном направлении.			V

Синтаксис

```
VMOV{cond} Sd, Sd1, Rn, Rm        ; Копирование содержимого двух любых
                                  ; регистров ARM-ядра в два регистра
                                  ; модуля плавающей точки с соседними
                                  ; номерами
VMOV{cond} Rd1, Rd2, Sm, Sm1      ; Копирование содержимого двух
                                  ; регистров модуля плавающей точки с
                                  ; соседними номерами в два
                                  ; регистра ARM-ядра
```

Где:

cond - Код условного выполнения команды (опция)

R - Регистр-ARM ядра.
В первом случае - источник двух слов Rn и Rm, первое из которых копируется в регистр плавающей точки с меньшим номером Sd, а второе – в регистр плавающей точки со следующим по порядку номером Sd1.
Во втором случае – регистры-приемники данных Rd1 и Rd2, в которые последовательно пересылаются данные из регистра плавающей точки с меньшим номером Sm, а затем – со следующим номером по порядку Sm1.

S - Регистры модуля плавающей точки однократной точности с последовательными номерами. В первом случае - приемники данных Sd и Sd1, во втором – источники данных Sm и Sm1.

Операция

Пересылка содержимого любых двух регистров ARM-ядра в два регистра модуля плавающей точки с последовательными номерами.
Пересылка содержимого двух регистров модуля плавающей точки с последовательными номерами в два любых регистра ARM-ядра.

Ограничения

Регистры ARM-ядра могут быть регистрами с любыми номерами.
Они не могут быть указателем стека SP или счетчиком команд PC.
Регистры модуля плавающей точки должны быть двумя регистрами с последовательными номерами.

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMOV S0,S1,R7,R10      ; Копирование содержимого регистров ЦПУ R7
                       ; и R10 в регистры сопроцессора S0, S1
VMOV R3,R10,S4,S5      ; Копирование содержимого регистров S4,S5
                       ; сопроцессора в регистры ЦПУ R3, R10
```

8.8 Копирование слова из регистра двойной точности в регистр ЦПУ или из регистра ЦПУ в одно из слов регистра двойной точности

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	Transfers one word of a doubleword floating-point register to an ARM core register Пересылка одного слова (младшего или старшего) из регистра плавающей точки двойной точности в регистр ARM-ядра (пересылка скаляра, т.е. слова «как есть»)			V

Синтаксис

```
VMOV{cond} Rd, Dm[x]        ; Пересылка одного слова из регистра
                            ; плавающей точки двойной точности в регистр
                            ; ARM-ядра
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Dm** Регистр-источник данных для операции пересылки слова. Регистр двойной точности в модуле плавающей точки (D0-D15).
- **Rd** – Регистр-приемник данных, один из регистров ARM-ядра.
- **x** – Специфицирует, какая половина регистра двойного слова сопроцессора используется для копирования данных:

0	Младшая половина двойного слова (младшее слово)
1	Старшая половина двойного слова (старшее слово)

Операция

Копирование одного слова из младшей или старшей половины регистра плавающей точки двойной точности в регистр ARM-ядра.

Ограничения

Регистр назначения в ARM-ядре не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMOV R0, D3[0]  ; Копирование младшего слова из регистра двойной
                ; точности D3 в регистр ЦПУ R0
VMOV R1, D3[1]  ; Копирование старшего слова из регистра двойной
                ; точности D3 в регистр ЦПУ R1
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMOV	Transfers one word to a doubleword floating-point register from an ARM core register Пересылка слова из регистра ARM-ядра в регистр плавающей точки двойной точности (в младшую или старшую часть) - пересылка скаляра, т.е. слова «как есть»			V

Синтаксис

```
VMOV{cond}{.32} Dd[x], Rm  ; Пересылка слова из регистра ARM-ядра в
                           ; младшую или старшую часть регистра
                           ; сопроцессора двойной точности
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Rm** Регистр-источник данных для операции пересылки слова. Любой из регистров ARM-ядра.
- **Dd** – Регистр двукратной точности сопроцессора (один из D0-D15)
- **x** – Специфицирует, какая половина регистра сопроцессора двукратной точности используется в качестве приемника данных:

0	Младшая половина двойного слова (младшее слово)
1	Старшая половина двойного слова (старшее слово)

Операция

Пересылает содержимое регистра ARM-ядра в младшую или старшую часть регистра сопроцессора двукратной точности (рассматривая это содержимое просто как скаляр – «как есть»).

Ограничения

Регистр-источник в ARM-ядре не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMOV D3[0], R0  ; Копирование содержимого регистра ЦПУ R0 в
                ; младшее слово регистра двойной точности D3
VMOV D3[1], R1  ; Копирование содержимого регистра ЦПУ R1 в
                ; старшее слово регистра двойной точности D3
```

8.9 Обмен данными между регистром специального назначения сопроцессора и регистрами ЦПУ

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMRS	Move to ARM Core register from floating-point System Register. Пересылка в регистр ARM-ядра содержимого системного регистра модуля плавающей точки (регистра статуса и управления FPU – FPSCR)			V

Синтаксис

```
VMRS{cond} Rd, FPSCR          ; Пересылка в любой из регистров ARM-
                              ; ядра
VMRS{cond} APSR_nzcv, FPSCR   ; Пересылка в регистр статуса
                              ; программы приложения APSR
                              ;(копируются только флаги: n,z,c,v)
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Rd** Регистр-приемник данных. Любой из регистров ARM-ядра R0-R14
- **APSR_nzcv** – Биты флагов регистра статуса программы приложения
- **FPSCR** – Регистр статуса и управления модулем плавающей точки

Операция

1. Текущее состояние регистра статуса и управления модулем плавающей точки FPSCR копируется в регистр общего назначения ARM-ядра.
2. Биты флагов из регистра статуса и управления модулем плавающей точки FPSCR копируются в соответствующие биты флагов N, Z, C и V регистра статуса программы приложения APSR. После этого можно пользоваться обычными командами условного выполнения команд или условного ветвления программы.

Ограничения

Регистр-приемник в ARM-ядре не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Вторая версия команды предназначена именно для установки флагов N, Z, C, V.

Примеры

```
VMRS R0, FPSCR          ; Копирование содержимого регистра статуса и
                        ; управления модуля плавающей точки FPSCR в
                        ; в регистр ЦПУ R0
VMOV APSR_zncv, FPSCR ; Копирование флагов из регистра статуса и
                        ; управления сопроцессором FPSCR в регистр
                        ; статуса программы приложения APSR
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMSR	Move to floating-point System Register from ARM Core register Загрузка данных из регистра ARM-ядра в системный регистр модуля плавающей точки (регистр статуса и управления FPU – FPSCR)			V

Синтаксис

```
VMSR{cond} FPSCR, Rm        ; Пересылка содержимого регистра общего
                            ; назначения ЦПУ в регистр статуса
                            ; и управления сопроцессором
                            ; (Инициализация или пере-инициализация FPU)
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Rm** Регистр общего назначения ЦПУ – источник данных
- **FPSCR** – Регистр статуса и управления модулем плавающей точки

Операция

Выполняется загрузка регистра статуса и управления модуля плавающей точки FPU данными из регистра общего назначения ARM-ядра.
Используется для инициализации или пере-инициализации сопроцессора.

Ограничения

Регистр-источник в ARM-ядре не может быть указателем стека SP или счетчиком команд PC

Выработка флагов

Выполняется полное обновление содержимого регистра статуса и управления модуля плавающей точки FPSCR, в том числе флагов, расположенных в этом регистре - N, Z, C, V.

Примеры

```
      VMSR FPSCR, R0        ; Копирование содержимого регистра
                            ; ЦПУ R0 в регистр статуса и
                            ; управления модуля плавающей точки FPSCR
                            ; R0 содержит код инициализации FPU
```

8.10 Множественное сохранение/восстановление содержимого регистров сопроцессора в стеке/из стека

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VPUSH	Floating-point extension register Push. Сохранение содержимого регистров модуля плавающей точки в стеке.			V

Синтаксис

```
VPUSH{cond}{.size} list    ; Сохранение содержимого регистров
                           ; сопроцессора в стеке
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Size** – Спецификатор длины передаваемых данных (опция). Если он присутствует, то для регистров однократной точности **S**, указанных в листинге, должен быть равен **32**, а для регистров двукратной точности **D**, – **64** (длина данных в битах)
- **List** – Листинг регистров (список), значения которых будут сохраняться в стеке. Представляет собой список регистров сопроцессора однократной **S** или двойной **D** точности, разделенных запятыми, заключенный в фигурные скобки. Допускается задание списка регистров в виде диапазона. В списке должны быть регистры только одного типа – S или D. Смешивание типов не допускается.

Операция

Эта команда сохраняет в стеке значения нескольких, указанных в списке, регистров сопроцессора в последовательности возрастания их номеров – начиная с регистра с самым младшим номером и кончая регистром с самым старшим номером.

Она эквивалентна команде **VSTMDB sp!,Registers** множественного сохранения в стеке содержимого регистров сопроцессора с декрементированием адреса перед каждым доступом к памяти и авто-обновлением содержимого указателя (SP) в конце пересылки. Обе команды дизассемблируются в команды VPUSH.

Ограничения

Листинг должен содержать по крайней мере имя одного регистра сопроцессора, но не более чем 16 имен регистров.

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
VPUSH {S0,S1,S2}      ; Сохранить в стеке содержимое регистров
                      ; однократной точности S0, S1, S2
VPUSH {D0,D1,D2}      ; Сохранить в стеке содержимое регистров
                      ; двукратной точности D0, D1, D2
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VPOP	Floating-point extension register Pop. Восстановление содержимого регистров модуля плавающей точки из стека.			V

Синтаксис

```
VPOP{cond}{.size} list      ; Восстановление содержимого регистров
                            ; сопроцессора из стека
```

Где:

- **cond** – Код условного выполнения команды (опция)
- **Size** – Спецификатор длины передаваемых данных (опция). Если он присутствует, то для регистров однократной точности **S**, указанных в листинге должен быть равен **32**, а для регистров двукратной точности **D**, – **64** (длина данных в битах)
- **List** – Листинг регистров (список), значения которых будут восстанавливаться из стека. Представляет собой перечень регистров сопроцессора однократной **S** или двойной **D** точности, разделенных запятыми, заключенный в фигурные скобки. Допускается задание списка регистров в виде диапазона. В списке должны быть регистры только одного типа – S или D. Смешивание типов не допускается.

Операция

Эта команда восстанавливает значения нескольких регистров сопроцессора в обратной последовательности, то есть по убыванию их номеров: сначала восстанавливается значение регистра с самым старшим номером, а в конце – с самым младшим.
Эквивалентна команде **VLDM SP!, Registers** с авто-инкрементированием указателя SP после каждой записи данных в очередной регистр сопроцессора и с авто-модификацией указателя (SP) после завершения всех операций считывания данных из стека.

Ограничения

Листинг должен содержать по крайней мере имя одного регистра сопроцессора, но не более чем 16 имен регистров.

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
      VPOP {S0,S1,S2}       ; Восстановить из стека содержимое регистров
                            ; однократной точности S0, S1, S2
      VPOP {D0,D1,D2}       ; Восстановить из стека содержимое регистров
                            ; двукратной точности D0, D1, D2
```

8.11 Обработка одного операнда

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VABS	Floating-point Absolute Абсолютное значение числа с плавающей точкой			V

Синтаксис

```
VABS{cond}.F32 Sd, Sm
```

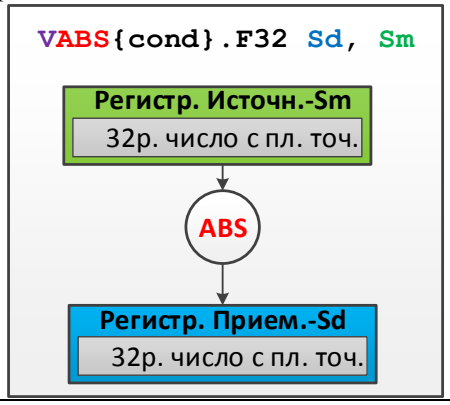
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции
	Sm – Регистр исходного операнда модуля FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Вычисляет абсолютное значение числа с плавающей точкой в регистре исходного операнда.
2. Сохраняет результат в регистре назначения.

Графическая иллюстрация алгоритма:



Ограничения

Нет ограничений

Выработка флагов

Команда очищает флаг знака S, т.к. результат операции всегда положительный

Примеры

```
VABS.F32 S4, S6 ; Вычисление абсолютного значения числа в формате
                ; с плавающей точкой в регистре S6
                ; и сохранение результата в регистре S4
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VNEG	Floating-point Negate Отрицание числа в формате с плавающей точкой			V

Синтаксис

```
VNEG {cond}.F32 Sd, Sm
```

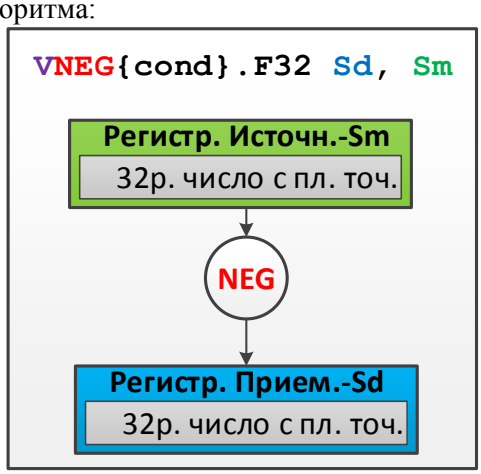
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции
	Sm – Регистр исходного операнда модуля FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Выполняет отрицание числа в формате с плавающей точкой в регистре исходного операнда.
2. Сохраняет результат в регистре-приемнике модуля обработки чисел с плавающей точкой

Эта команда всего лишь инвертирует знаковый бит числа в формате с плавающей точкой sign bit – выполняет операцию «унарного минуса».
Графическая иллюстрация алгоритма:



Ограничения

Нет ограничений

Выработка флагов

Не влияет на состояние флагов

Примеры

```
        VNEG.F32   S3, S4      ; Изменить знак числа в регистре-источнике
                               ; S4 и сохранить результат в регистре S3
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VSQRT	Floating-point Square Root Корень квадратный из числа в формате с плавающей точкой			V

Синтаксис

```
VSQRT {cond}.F32 Sd, Sm
```

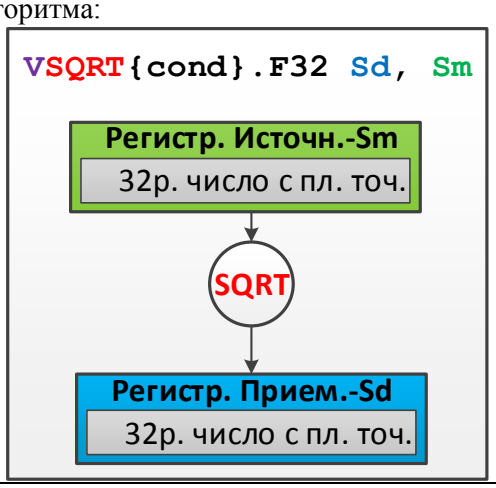
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции
	Sm – Регистр исходного операнда модуля FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Вычисляет корень квадратный из числа в формате с плавающей точкой в регистре исходного операнда.
2. Сохраняет результат в регистре-приемнике модуля обработки чисел с плавающей точкой

Графическая иллюстрация алгоритма:



Ограничения

Нет ограничений

Выработка флагов

Не влияет на состояние флагов

Примеры

```
VSQRT.F32  S5, S6      ; Вычисляет корень квадратный из числа
                       ; в регистре-источнике S6 и сохраняет
                       ; результат в регистре-приемнике S5
```

8.12 Арифметические операции с двумя операндами

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VADD	Floating-point Add Сложение чисел с плавающей точкой			V

Синтаксис

```
VADD{cond}.F32 {Sd,} Sn, Sm
```

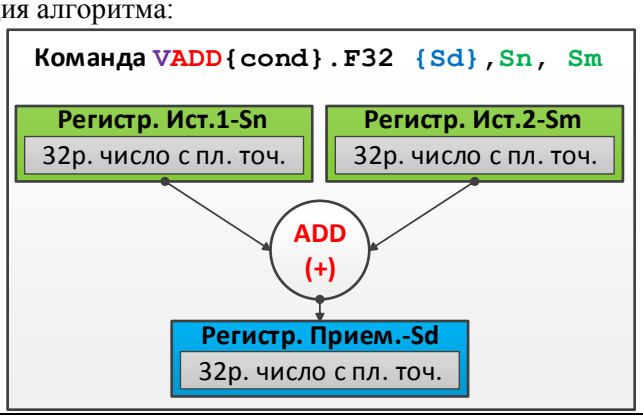
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции сложения. Если он опущен, то функцию приемника выполняет регистр первого операнда Sn
	Sn – Регистр первого операнда (слагаемого) в модуле FPU, содержащий число в формате с плавающей точкой
	Sm – Регистр второго операнда (слагаемого) в модуле FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Складывает два числа в формате с плавающей точкой из регистров первого и второго операндов.
2. Сохраняет результат в регистре назначения.

Графическая иллюстрация алгоритма:



Ограничения

Нет ограничений

Выработка флагов

Команда не изменяет состояние флагов

Примеры

```
VADD.F32 S4, S6, S7   ; Сложение двух чисел с плавающей точкой в
                      ; регистрах исходных операндов S6 и S7
                      ; Сохранение суммы в регистре S4
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VSUB	Floating-point Subtract. Вычитание чисел с плавающей точкой			V

Синтаксис

```
VSUB{cond}.F32 {Sd,} Sn, Sm
```

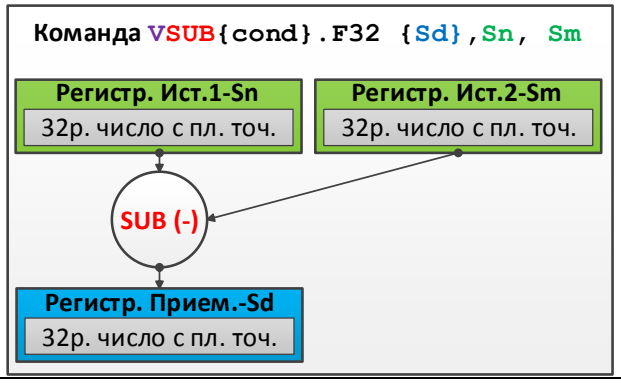
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции вычитания. Если он опущен, то функцию приемника выполняет регистр первого операнда Sn
	Sn – Регистр первого операнда в модуле FPU, содержащий число в формате с плавающей точкой
	Sm – Регистр второго операнда в модуле FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Вычитает из содержимого первого регистра с плавающей точкой содержимое второго регистра с плавающей точкой.
2. Сохраняет результат в регистре назначения.

Графическая иллюстрация алгоритма выполнения команды:



Ограничения

Нет ограничений

Выработка флагов

Команда не изменяет состояние флагов

Примеры

```
VSUB.F32 S4, S6, S7   ; Вычитание двух чисел с плавающей точкой в
                      ; регистрах исходных операндов S6 и S7
                      ; Сохранение разности в регистре S4
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMUL	Floating-point Multiply Умножение чисел с плавающей точкой			V

Синтаксис

```
VMUL{cond}.F32 {Sd,} Sn, Sm
```

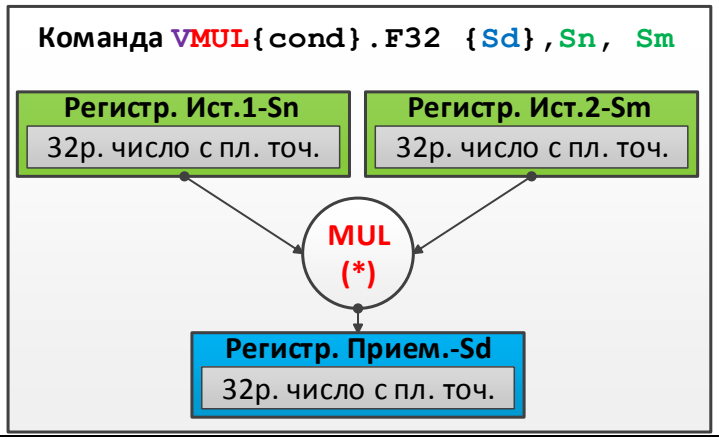
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции умножения. Если он опущен, то функцию приемника выполняет регистр первого операнда Sn
	Sn – Регистр первого операнда в модуле FPU, содержащий число в формате с плавающей точкой
	Sm – Регистр второго операнда в модуле FPU, содержащий число в формате с плавающей точкой

Операция

Эта команда:

1. Умножает содержимое первого регистра с плавающей точкой на содержимое второго регистра с плавающей точкой.
2. Сохраняет произведение в регистре назначения.

Графическая иллюстрация алгоритма выполнения команды:



Ограничения

Нет ограничений

Выработка флагов

Команда не изменяет состояние флагов

Примеры

```
VMUL.F32 S3, S6, S7   ; Умножение двух чисел с плавающей точкой в
                      ; регистрах исходных операндов S6 и S7
                      ; Сохранение произведения в регистре S3
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VDIV	Divides floating-point values Деление двух чисел с плавающей точкой			V

Синтаксис

```
VDIV{cond}.F32 {Sd,} Sn, Sm
```

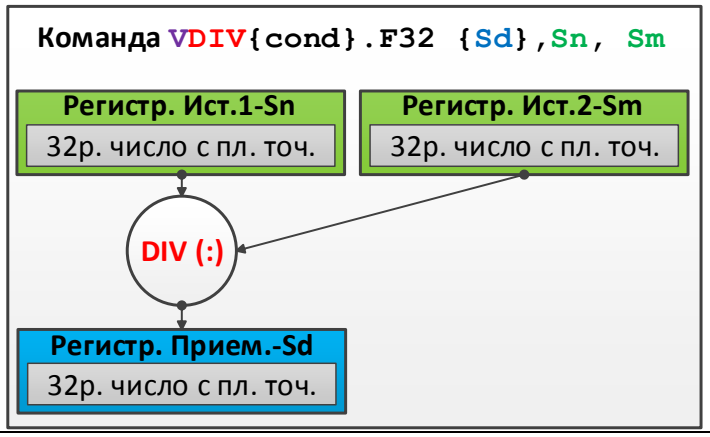
Где:	**cond** код условного выполнения команды (опция)
	Sd – Регистр назначения модуля FPU, принимающий результат операции деления. Если он опущен, то функцию приемника выполняет регистр первого операнда Sn
	Sn – Регистр первого операнда в модуле FPU, содержащий число в формате с плавающей точкой - делимое
	Sm – Регистр второго операнда в модуле FPU, содержащий число в формате с плавающей точкой - делитель

Операция

Эта команда:

1. Делит содержимое первого регистра с плавающей точкой на содержимое второго регистра с плавающей точкой.
2. Сохраняет результат деления (частное) в регистре назначения.

Графическая иллюстрация алгоритма выполнения команды:



Ограничения

Нет ограничений

Выработка флагов

Команда не изменяет состояние флагов

Исключения

Команда может вызывать исключения: Деление на ноль (Division by Zero), Некорректная операция (Invalid Operation), Переполнение вверх (Overflow), Переполнение вниз (Underflow), Неточность (Inexact).

Примеры

```
VDIV.F32 S3, S10, S11 ; Деление двух чисел с плавающей точкой в
                      ; регистрах исходных операндов S10 и S11
                      ; Сохранение частного в регистре назначения
                      ; (S3) = (S10):(S11)
```

8.13 Стандартное умножение с положительным и отрицательным накоплением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VMLA	Multiplies two floating-point values and accumulates Умножение двух чисел с плавающей точкой и аккумулирование			V
VMLS	Multiplies two floating-point values and subtracts the results Умножение двух чисел с плавающей точкой и вычитание результата			

Синтаксис

```
VMLA{cond}.F32 Sd, Sn, Sm        ; Умножение числе с плавающей точкой и
                                 ; аккумулирование
VMLS{cond}.F32 Sd, Sn, Sm        ; Умножение числе с плавающей точкой и
                                 ; вычитание (отрицательное
                                 ; аккумулирование)
```

Где:	
	cond код условного выполнения команды (опция)
	Sd – Регистр назначения в модуле плавающей точки - аккумулятор
	Sn – Регистр первого множителя с плавающей точкой
	Sm – Регистр второго множителя с плавающей точкой

Операция

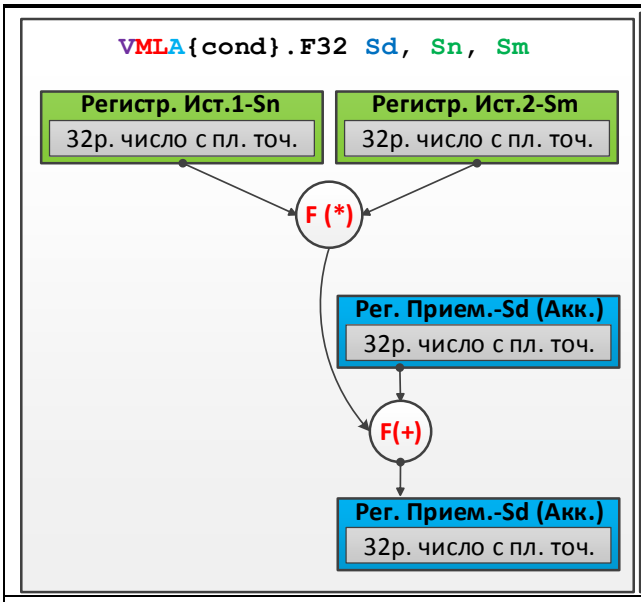
Команда **VMLA**:

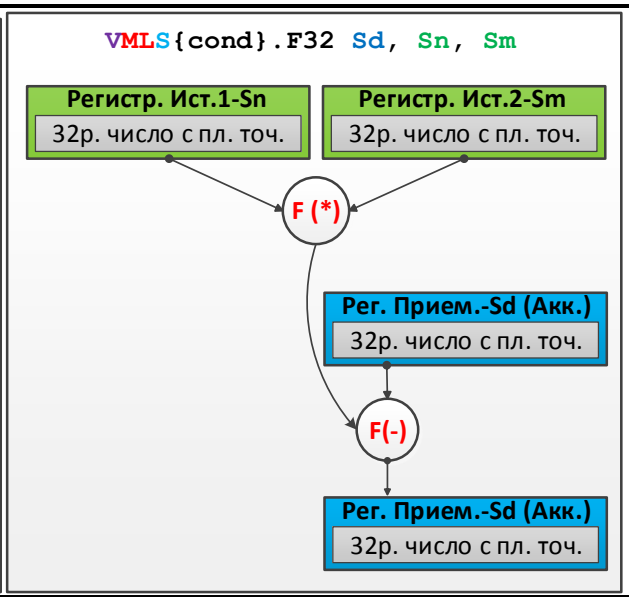
1. Умножает два числа в формате с плавающей точкой однократной точности в регистрах исходных операндов Sn и Sm.
2. Добавляет результат к содержимому регистра-назначения Sd и сохраняет сумму там же (аккумулирование).

Команда **VMLS**:

1. Умножает два числа в формате с плавающей точкой однократной точности в регистрах исходных операндов Sn и Sm.
2. Вычитает произведение из содержимого регистра-назначения Sd и сохраняет разность там же (отрицательное аккумулирование).

Графическая иллюстрация алгоритма выполнения этих команд:





Ограничения

Нет ограничений

Исключения

Команда может генерировать исключения: Некорректная операция (Invalid Operation), Переполнение вверх (Overflow), Переполнение вниз (Underflow), Неточность (Inexact) или Входное денормализованное число (Input Denormal).

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VMLA.F32 S3, S5, S7   ; Умножение чисел с плавающей точкой в
                      ; регистрах S5 и S7
                      ; Аккумулирование произведения в регистре
                      ; S3: (S3) = (S3)+(S5)*(S7)
VMLS.F32 S2, S10, S11 ; Умножение чисел с плавающей точкой в
                      ; регистрах S10 и S11
                      ; Вычитание произведения из содержимого
                      ; регистра S2.
                      ; Сохранение разности в регистре S2
                      ; (S2) = (S2)-(S10)*(S11)
```

8.14 Стандартное умножение, умножение с положительным или отрицательным накоплением и инверсией результата

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VNMUL VNMLA VNMLS	Floating-point multiply with negation followed by add or subtract Умножение двух чисел с плавающей точкой, умножение с положительным или отрицательным накоплением, инверсия результата			V

Синтаксис

```
VNMUL{cond}.F32 {Sd,} Sn, Sm      ; Умножение чисел с плавающей точкой и
                                  ; инверсия знака произведения
VNMLA{cond}.F32 Sd, Sn, Sm        ; Умножение чисел с плавающей точкой,
                                  ; аккумулирование и инверсия
VNMLS{cond}.F32 Sd, Sn, Sm        ; Умножение числе с плавающей точкой,
                                  ; вычитание (отрицательное
                                  ; аккумулирование)и инверсия
```

Где:

- **cond** — код условного выполнения команды (опция)
- **Sd** – Регистр назначения в модуле плавающей точки - аккумулятор
- **Sn** – Регистр первого множителя с плавающей точкой
- **Sm** – Регистр второго множителя с плавающей точкой

Операция

Команда **VNMUL**:

1. Умножает два числа в формате с плавающей точкой однократной точности в регистрах исходных операндов Sn и Sm.
2. Инвертирует результат и сохраняет его в регистре назначения Sd.

Команда **VNMLA**:

1. Умножает два числа в формате с плавающей точкой однократной точности в регистрах исходных операндов Sn и Sm.
2. Добавляет произведение к содержимому регистра-назначения Sd (аккумулятора).
3. Инвертирует результат и сохраняет его в регистре назначения Sd.

Команда **VNMLS**:

1. Умножает два числа в формате с плавающей точкой однократной точности в регистрах исходных операндов Sn и Sm.
2. Вычитает произведение из содержимого регистра-назначения Sd (аккумулятора).
3. Инвертирует результат и сохраняет его в регистре назначения Sd.

Если обозначить текущее содержимое аккумулятора - (A), текущее содержимое множителей (B) и (C), то команды выполняют следующие действия:

Команда **VNMUL**: `(A)= -(B*C);`

Команда **VNMLA**: `(A)= -[(A)+(B*C)];`

Команда **VNMLS**: `(A)= -[(A)-(B*C)];`

Ограничения

Нет ограничений

Исключения

Команда может генерировать исключения: Некорректная операция (Invalid Operation), Переполнение вверх (Overflow), Переполнение вниз (Underflow), Неточность (Inexact) или Входное денормализованное число (Input Denormal).

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VNMUL.F32 S2, S3, S4  ; Умножение чисел с плавающей точкой в
                      ; регистрах S3 и S4, инверсия знака
                      ; произведение и сохранение результата в S2
VNMLA.F32 S3, S5, S7  ; Умножение чисел с плавающей точкой в
                      ; регистрах S5 и S7. Сложение произведения
                      ; с содержимым аккумулятора S3. Инверсия
                      ; результата и сохранение в регистре S3.
                      ; (S3) = -[(S3)+(S5)*(S7)]
VNMLS.F32 S2,S10,S11  ; Умножение чисел с плавающей точкой в
                      ; регистрах S10 и S11.
                      ; Вычитание произведения из содержимого
                      ; регистра S2. Инверсия результата
                      ; Сохранение результата в регистре S2
                      ; (S2) = -[(S2)-(S10)*(S11)]
```

8.15 Совмещенное умножение с положительным и отрицательным накоплением

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VFMA VFMS	Floating-point Fused Multiply Accumulate Умножение чисел с плавающей точкой совмещенное с накоплением Floating-point Fused Multiply and Subtract Умножение чисел с плавающей точкой совмещенное с вычитанием (отрицательным накоплением)			v

Синтаксис

```
VFMA{cond}.F32 {Sd,} Sn, Sm      ; Умножение, совмещенное с накоплением
                                 ; Одно округление результата
VFMS{cond}.F32 {Sd,} Sn, Sm      ; Умножение, совмещенное с
                                 ; отрицательным накоплением
                                 ; Одно округление результата
```

Где:	
	cond код условного выполнения команды (опция)
	Sd – Регистр назначения в модуле плавающей точки, аккумулятор
	Sn – Регистр первого множителя с плавающей точкой
	Sm – Регистр второго множителя с плавающей точкой

Операция

Команда **VFMA**:

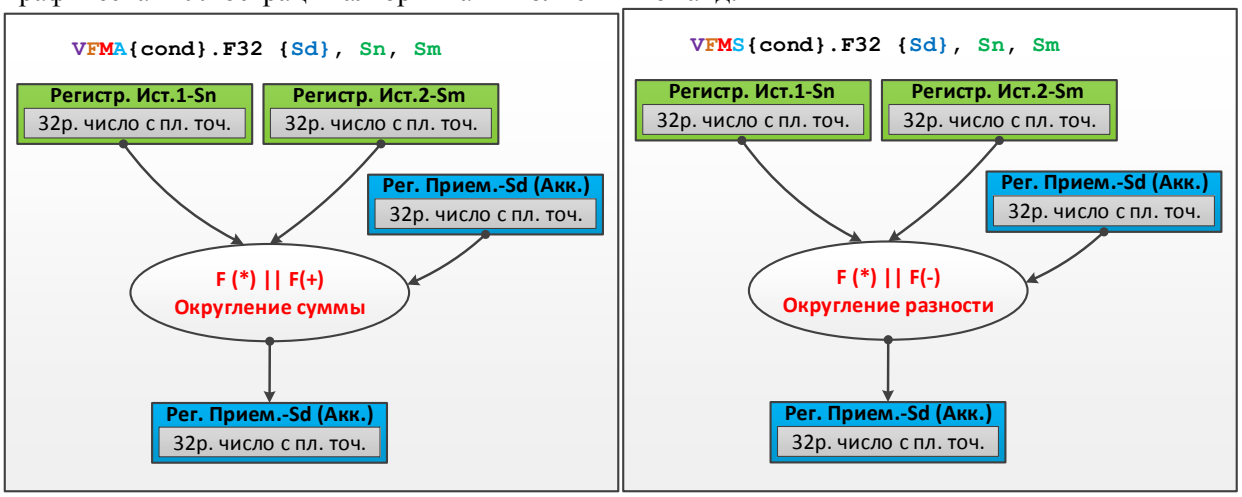
1. Умножает два числа в формате с плавающей точкой в регистрах исходных операндов Sn, Sm.
2. Аккумулирует результат в регистре назначения Sd (аккумуляторе). Результат умножения перед сложением не округляется (not rounded). К аккумулятору добавляется каждое частное произведение и округляется только конечный результат накопления.

Команда **VFMS**:

1. Умножает два числа в формате с плавающей точкой в регистрах исходных операндов Sn, Sm.
2. Вычитает результат из содержимого регистра назначения Sd (аккумулятора). Результат умножения перед вычитанием не округляется (not rounded). Из аккумулятора вычитается каждое частное произведение и округляется только конечный результат отрицательного накопления.

Примечание.
Fused floating-point multiply – умножение чисел с плавающей точкой совмещенное с операцией накопления. Вместо двух операций округления (произведения и суммы/разности) выполняется только одно округление результата накопления, за счет чего достигается повышение точности. Это более совершенная технология аппаратного умножения с накоплением.

Графическая иллюстрация алгоритма выполнения команд:



Ограничения

Нет ограничений

Исключения

Команда может генерировать исключения: Некорректная операция (Invalid Operation), Переполнение вверх (Overflow), Переполнение вниз (Underflow), Неточность (Inexact) или Входное денормализованное число (Input Denormal).

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VFMA.F32 S3, S5, S7   ; Умножение чисел с плавающей точкой в
                      ; регистрах S5 и S7 с параллельным
                      ; накоплением в регистре S3
                      ; Округление накопленной суммы
                      ; (S3)=(S3)+(S5)*(S7)
VFMS.F32 S2, S10, S11 ; Умножение чисел с плавающей точкой в
                      ; регистрах S10 и S11 с параллельным
                      ; отрицательным накоплением (вычитанием)
                      ; из регистра S2.
                      ; Округление разности
                      ; (S2)= (S2)-(S10)*(S11)
```

8.16 Совмещенное умножение с положительным и отрицательным накоплением и инверсией результата

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VFNMA VFNMS	Floating-point Fused Negate Multiply Accumulate Умножение чисел с плавающей точкой с параллельным аккумулированием, инвертирование результата Floating-point Fused Multiply Negate and Subtract Умножение чисел с плавающей точкой с параллельным отрицательным аккумулированием (вычитанием), инвертирование результата			v

Синтаксис

```
VFNMA{cond}.F32 {Sd,} Sn, Sm          ; Умножение с параллельным
                                      ; аккумулированием,
инвертирование                        ; результата
VFNMS{cond}.F32 {Sd,} Sn, Sm          ; Умножение с параллельным
                                      ; вычитанием произведения,
                                      ; инвертирование результата
```

Где:	
	cond код условного выполнения команды (опция)
	Sd – Регистр назначения в модуле плавающей точки, аккумулятор
	Sn – Регистр первого множителя с плавающей точкой
	Sm – Регистр второго множителя с плавающей точкой

Операция

Команда **VFNMA**:

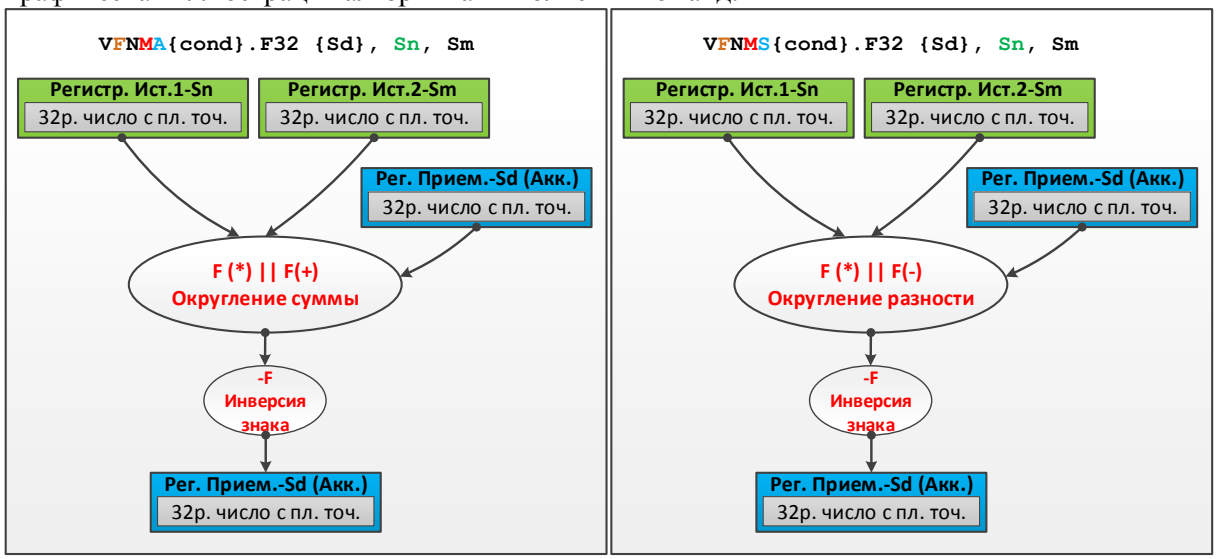
1. Умножает два числа в формате с плавающей точкой в регистрах исходных операндов Sn, Sm.
2. Аккумулирует результат в регистре назначения Sd (аккумуляторе). Результат умножения перед сложением не округляется (not rounded). К аккумулятору добавляется каждое частное произведение. Округляется результат накопления.
3. Инвертирует полученное значение и сохраняет в аккумуляторе – регистре назначения Sd.

Команда **VFNMS**:

1. Умножает два числа в формате с плавающей точкой в регистрах исходных операндов Sn, Sm.
2. Вычитает результат из содержимого регистра назначения Sd (аккумулятора). Результат умножения перед вычитанием не округляется (not rounded). Из аккумулятора вычитается каждое частное произведение. Округляется результат отрицательного накопления.
3. Инвертирует полученное значение и сохраняет в аккумуляторе – регистре назначения Sd.

Примечание.
Fused floating-point multiply – умножение чисел с плавающей точкой, которое выполняется параллельно с операцией положительного/отрицательного накопления. Округляется только накопленное значение. Это более совершенная технология аппаратного умножения с накоплением.

Графическая иллюстрация алгоритма выполнения команд:



Ограничения

Нет ограничений

Исключения

Команда может генерировать исключения: Некорректная операция (Invalid Operation), Переполнение вверх (Overflow), Переполнение вниз (Underflow), Неточность (Inexact) или Входное денормализованное число (Input Denormal).

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
VFNMA.F32 S3, S4, S8  ; Умножение чисел с плавающей точкой в
                      ; регистрах S4 и S8 с параллельным
                      ; аккумулированием произведения
                      ; (S3)=(S3)+(S4)*(S8)
                      ; Инвертирование результата (S3)=-(S3)
VFNMS.F32 S2, S10, S11     ; Умножение чисел с плавающей точкой в
                           ; регистрах S10 и S11 с параллельным
                           ; вычитанием произведения из
                           ; содержимого регистра S2
                           ; (S2)= (S2)- (S10)*(S11);
                           ; Инвертирование результата
                           ; отрицательного накопления и
                           ; сохранение в аккумуляторе
                           ; (S2)=-(S2)
```

8.17 Сравнение операндов с плавающей точкой

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VCMP VCMPE	Compares two floating-point registers, or one floating-point register and zero. Сравнение содержимого двух регистров с плавающей точкой или одного регистра с плавающей точкой с нулем			V

Синтаксис

```
VCMP{E}{cond}.F32 Sn, Sm
VCMP{E}{cond}.F32 Sn, #0.0
```

Где:

cond
код условного выполнения команды (опция)

E	Error - Включение контроля некорректных операндов. Если суффикс «E» указан, то любой операнд, который представляет собой «Не число» (NaN) вызывает исключение по неправильной операции Invalid Operation. В противном случае исключение возникает лишь в том случае, когда когда один из операндов является сигнальным «Не числом» (sNaN).

Sn – Регистр первого операнда в модуле FPU, содержащий число в формате с плавающей точкой, которое сравнивается

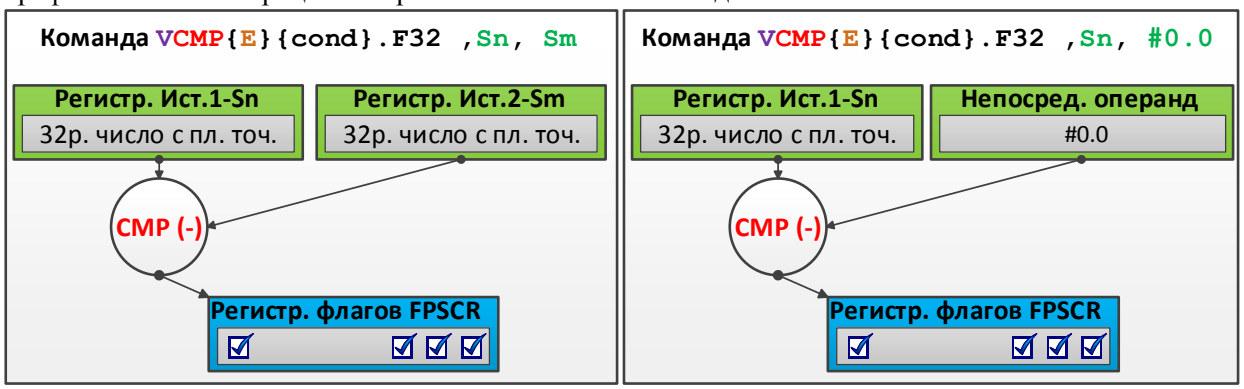
Sm – Регистр второго операнда в модуле FPU, содержащий число в формате с плавающей точкой, с которым выполняется сравнение

Операция

Эта команда:

1. Вычитает из содержимое первого регистра с плавающей точкой содержимое второго регистра с плавающей точкой или ноль.
2. Сохраняет флаги результата сравнения в регистре статуса и управления модулем обработки чисел с плавающей точкой FPSCR .

Графическая иллюстрация алгоритма выполнения команды:



Ограничения

Если указан суффикс проверки корректности исходных операндов, то в случае, если один из них оказывается «не числом» (NaN), возникает исключение по недопустимой операции (Invalid Operation exception).

Выработка флагов

Команда сохраняет результат сравнения двух чисел в формате с плавающей точкой (флаги) в регистре статуса и управления модулем FPU – в регистре FPSCR. Для того, чтобы переслать эти флаги в регистр флагов ARM-процессора, необходимо выполнить дополнительную команду VMRS (считывания флагов модуля FPU). После этого можно использовать любые команды ветвления и команды условного выполнения.

Примеры

```
VCMP.F32 S4, S2        ; Сравнение двух чисел с плавающей точкой
                       ; в регистрах S4 и S2
VCMP.F32 S4, #0.0      ; Сравнение одного числа с плавающей точкой
                       ; в регистре S4 с нулем
```

8.18 Конвертация между целыми числами и числами в формате с плавающей точкой

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VCVT **VCVTR**	**Converts between floating-point and integer – Конвертация между числами с плавающей точкой и целыми числами** Converts from integer to floating-point Конвертация целых чисел в формат чисел с плавающей точкой Converts a value in a register floating-point to a 32-bit integer Конвертация значения в регистре с плавающей точкой в 32-разрядное целое			V

Синтаксис

```
VCVT{cond}.F32.Tm Sd, Sm        ; Конвертация целого числа в формат с
                                ; плавающей точкой
VCVT{R}{cond}.Tm.F32 Sd, Sm     ; Из формата с плавающей точкой в
                                ; формат 32- разрядного целого числа
```

Где:

cond – Код условного выполнения команды (опция)

R	**Rounding.** Если опция «**R**» специфицирована, то в операции используется режим округления (rounding mode), заданный в регистре статуса и управления модуля плавающей точки FPSCR. Если опция «R» опущена, то в операции используется режим округления к нулю Zero rounding mode (усечение).

Tm – Тип данных целочисленного операнда (источника или приемника)

S32	32-разрядное целое число со знаком в дополнительном коде
U32	32-разрядое целое число без знака

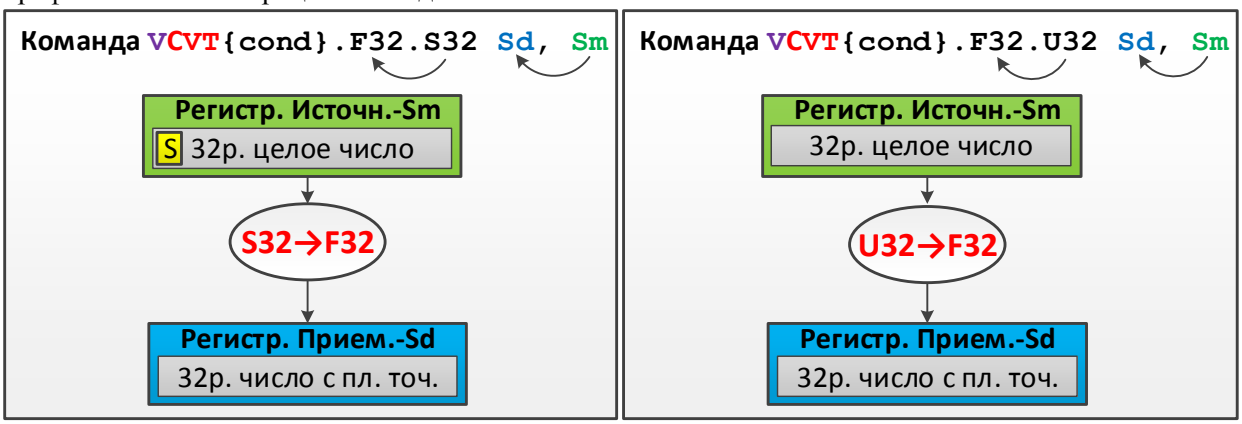
Sd – Регистр назначения в модуле FPU

Sm – Регистр-источник операнда в модуле FPU

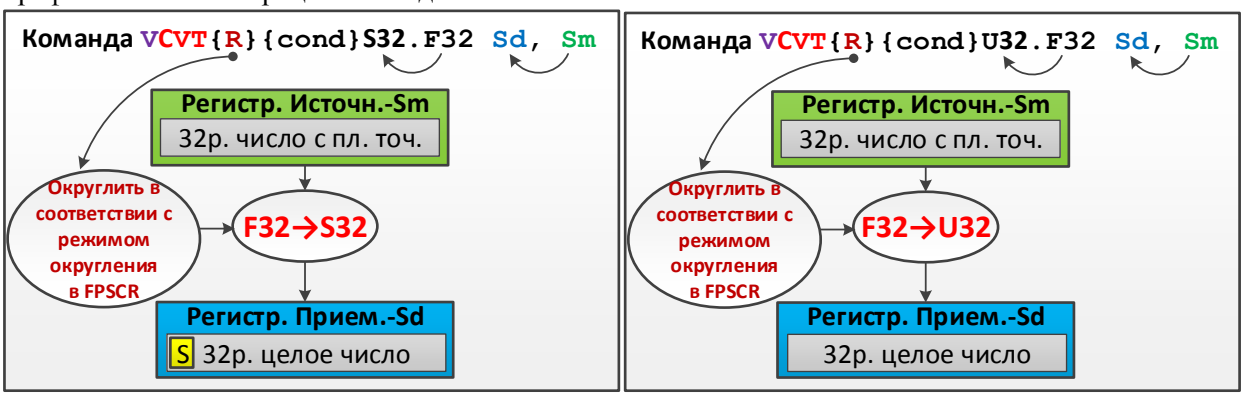
Операция

Команда **VCVT{cond}.F32.Tm Sd, Sm**:
Конвертирует 32-разрядное целое число со знаком (S32) или без знака (U32) из регистра источника Sm в формат числа с плавающей точкой однократной точности (32 разряда). Сохраняет результат операции в регистре назначения Sd.

При этом используется режим округления, специфицированный в регистре статуса и управления модуля плавающей точки FPSCR (обычно, до ближайшего возможного числа в формате с плавающей точкой).
Графическая иллюстрация команд:



Команда **VCVT{R}{cond}.Tm.F32 Sd, Sm:**
Конвертирует число в формате с плавающей точкой из регистра источника Sm в 32-разрядное целое со знаком (S32) или без знака (U32). Сохраняет результат операции в регистре назначения Sd.
Обычно используется режим округления до нуля Zero rounding mode. Однако, при указанной опции «R» будет использоваться тот режим округления, который специфицирован в регистре статуса и управления модулем плавающей точки FPSCR.
Если результат преобразования выходит за возможный диапазон целых числе, то в качестве результата возвращается максимально- или минимально возможное целое и одновременно формируется флаг исключения OFC («Переполнение вверх») или UFC («Переполнение вниз»). Кроме того, в этом случае формируется флаг IXC («Неточность»).
Графическая иллюстрация команд:



Ограничения

Нет ограничений

Выработка флагов

Возможно формирование флагов исключений для операции преобразования из формата числа с плавающей точкой в формат целого числа

Примеры

```
VCVT.F32.S32 S5, S7   ; Преобразование 32-разрядного целого числа
                      ; со знаком из регистра-источника S7
                      ; в число с плавающей точкой однократной
                      ; точности и сохранение в регистре S5
VCVT.F32.U32 S5, S7   ; Преобразование 32-разрядного целого числа
                      ; без знака из регистра-источника S7
                      ; в число с плавающей точкой однократной
                      ; точности и сохранение в регистре S5
VCVT. S32.F32 S3, S8  ; Преобразование 32-разрядного числа
                      ; в формате с плавающей точкой из регистра
                      ; S8 в 32-разрядное целое со знаком
                      ; и сохранение в регистре S3
```

8.19 Конвертация между числами в форматах с фиксированной и с плавающей точкой

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VCVT	**Converts between floating-point and fixed-point** **Конвертация между числами с плавающей точкой и фиксированной точкой** Конвертация числа с плавающей точкой в число с фиксированной точкой и обратно			V

Синтаксис

```
VCVT{cond}.Td.F32 Sd, Sm, #fbits; Из формата с плавающей точкой
                                ; в формат с фиксированной точкой
VCVT{cond}.F32.Td Sd, Sm, #fbits; Из формата с фиксированной точкой в
                                ; формат с плавающей точкой
```

Где:

cond – Код условного выполнения команды (опция)

Td – Тип данных операнда с фиксированной точкой

S32	32-разрядное целое со знаком в дополнительном коде
U32	32-разрядое целое без знака
S16	16-разрядное целое со знаком в дополнительном коде
U16	16-разрядное целое без знака

Sd – Регистр назначения в модуле FPU

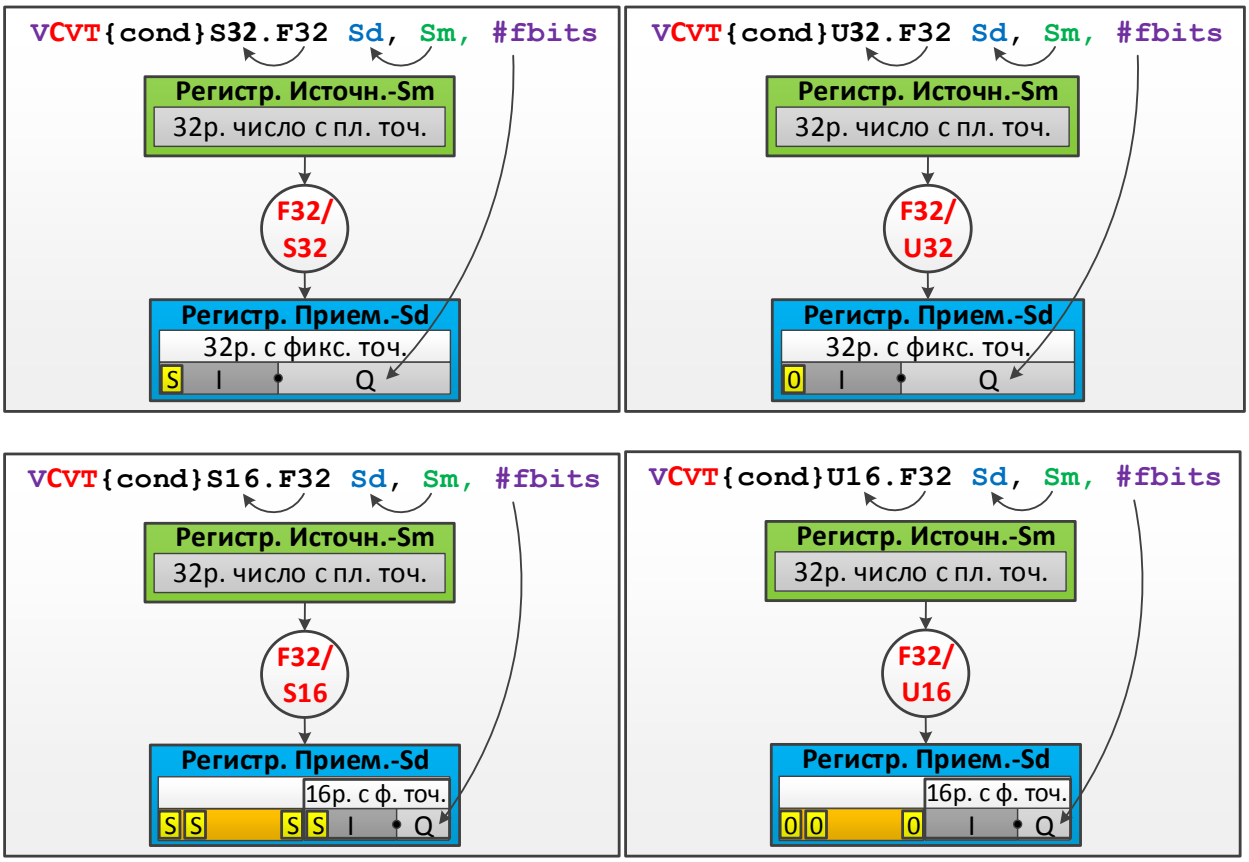
Sm – Регистр-источник операнда в модуле FPU

#fbits – число бит дробной части (fraction bits) числа с фиксированной точкой – Q для формата (I.Q)

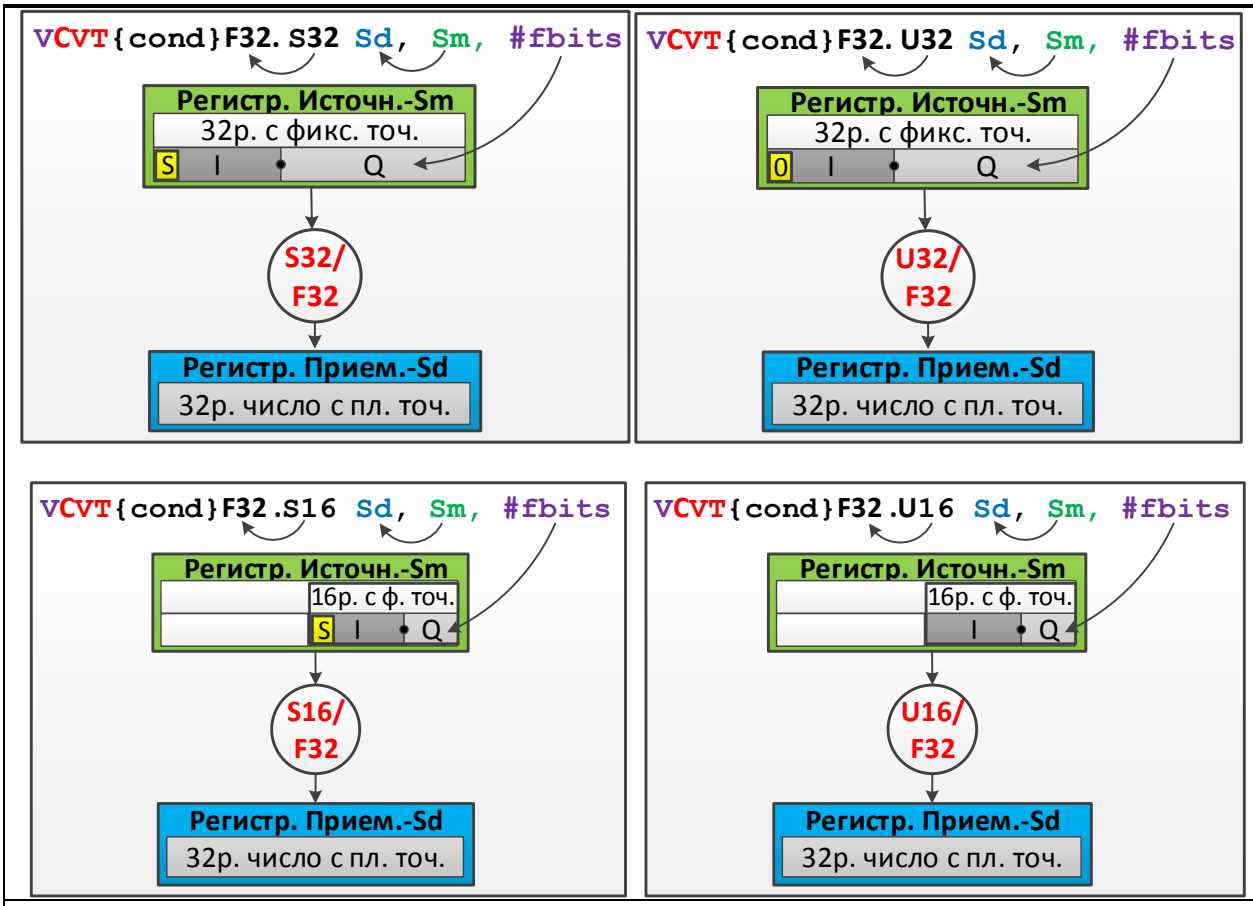
S32 или **U32**	Для 32-разрядных чисел с фиксированной точкой число разрядов дробной части может быть в диапазоне 1-32 (32 – для чистых дробей без знака). Ноль исключается, так как для этого есть команда преобразования в целые числа.
S16 или **U16**	Для 16-разрядных чисел с фиксированной точкой число разрядов дробной части может быть в диапазоне 0-16 (16 – для чистых дробей без знака)

Операция

Команда **VCVT{cond}.Td.F32 Sd, Sm, #fbits**
конвертирует число в формате с плавающей точкой из регистра Sm в число с фиксированной точкой с заданным числом разрядов дробной части **fbits**.
Сохраняет результат в регистре назначения Sd.
Результат в фиксированном формате (I.Q) может быть либо 16-разрядным, либо 32-х разрядным.
Если результат должен быть 16-разрядном числом со знаком типа S16, то он расширяется знаком до 32-х разрядов и записывается в регистр назначения.
Если результат должен быть 16-разрядным числом без знака типа U16, то он расширяется до 32-х разрядов нулями и записывается в регистр назначения.
В операции преобразования числа с плавающей точкой в число с фиксированной точкой используется режим округления до нуля Zero rounding mode.
Графическая иллюстрация алгоритма выполнения команды ниже:



Команда **VCVT{cond}.F32.Td Sd, Sm, #fbits**
конвертирует число в формате с фиксированной точкой с заданным числом разрядов дробной части **fbits** из регистра источника Sm в число с плавающей точкой.
Сохраняет результат в регистре назначения Sd.
Число в фиксированном формате может быть либо 16-разрядным, либо 32-х разрядным.
Операнд с фиксированной точкой (I.Q) типа S16 или U16 извлекается из младших значащих разрядов регистра-источника, старшие разряды (старшее полуслово) игнорируются.
В операции преобразования числа с фиксированной точкой в число с плавающей точкой используется режим округления Nearest rounding mode (до ближайшего возможного).
Графическая иллюстрация алгоритма выполнения команд:



Ограничения

Нет ограничений

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
        VCVT.F32.S32 S5, S7,#24     ; Преобразование 32-разрядного числа
                                    ; со знаком в формате с фиксированной
                                    ; точкой (8.24) из регистра-источника
                                    ; S7 в формат с плавающей точкой и
                                    ; сохранение в регистре S5
        VCVT.F32.S16 S5, S7,#12     ; Преобразование 16-разрядного числа
                                    ; со знаком в формате с фиксированной
                                    ; точкой (4.12) из регистра-источника
                                    ; S7 в формат с плавающей точкой и
                                    ; сохранение в регистре S5
        VCVT.S32.F32 S3, S5,#16     ; Преобразование 32-разрядного числа
                                    ; с плавающей точкой из регистра S5
                                    ; в 32-разрядное число со знаком в
                                    ; формате с фиксированной точкой
                                    ; (16.16) и сохранение в S3
```

8.20 Преобразование чисел с плавающей точкой половинной точности в формат однократной точности и обратно

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
VCVTB VCVTT	Converts between a half-precision value and a single-precision value Конвертирование между числами с плавающей точкой половинной точности (half-precision) и однократной точности single-precision			V

Синтаксис

```
VCVT{y}{cond}.F32.F16 Sd, Sm      ; Конвертация числа с плавающей точкой
                                  ; половинной точности F16 в число с
                                  ; плавающей точкой однократной
                                  ; точности F32
VCVT{y}{cond}.F16.F32 Sd, Sm      ; Конвертация числа с плавающей точкой
                                  ; однократной точности F32 в число с
                                  ; плавающей точкой половинной точности
                                  ; F16
```

Где:

cond – Код условного выполнения команды (опция)

y	Специфицирует полуслово регистра модуля с плавающей точкой, в котором размещается исходный операнд в формате половинной точности F16 или полуслово регистра назначения, в которое будет записан результат преобразования в формате половинной точности
B	Bottom – младшее полуслово [15:0] регистра источника Sm или регистра приемника Sd
T	Top – старшее полуслово [15:0] регистра источника Sm или регистра приемника Sd

Sd – Регистр назначения в модуле FPU (32 разряда)

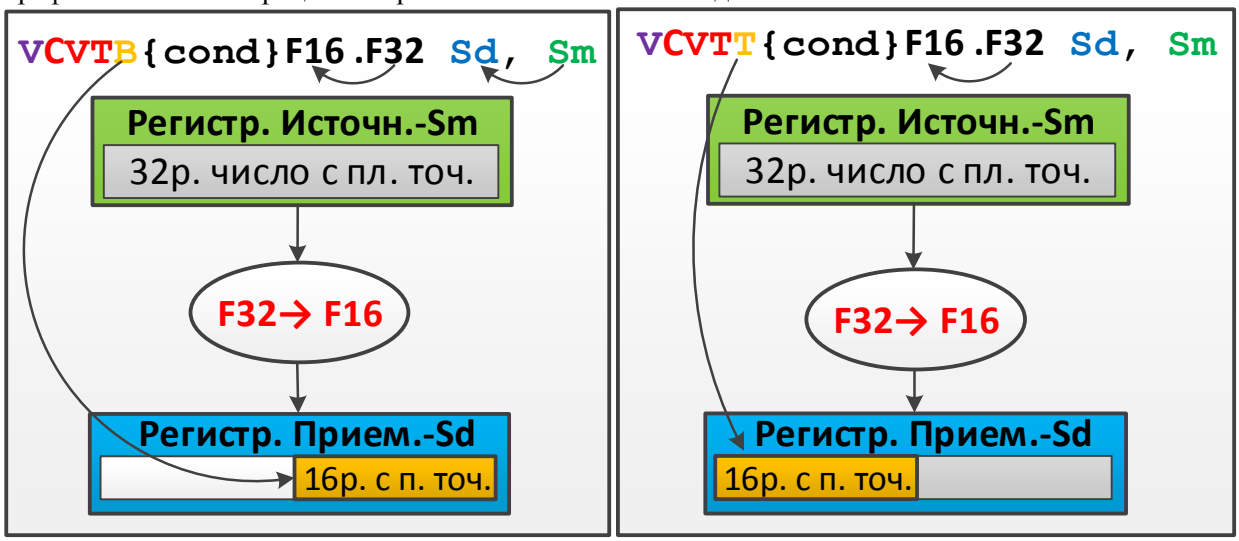
Sm – Регистр исходного операнда в модуле FPU (32 разряда)

Операция

Команда с суффиксом **.F16.F32:**

1. Конвертирует число формате с плавающей точкой однократной точности F32, расположенное в регистре-источнике Sm, в число с плавающей точкой половинной точности F16.
2. Сохраняет результат в младшем полу-слове (для суффикса «B») или в старшем полу-слове (для суффикса «T») регистра приемника Sd.

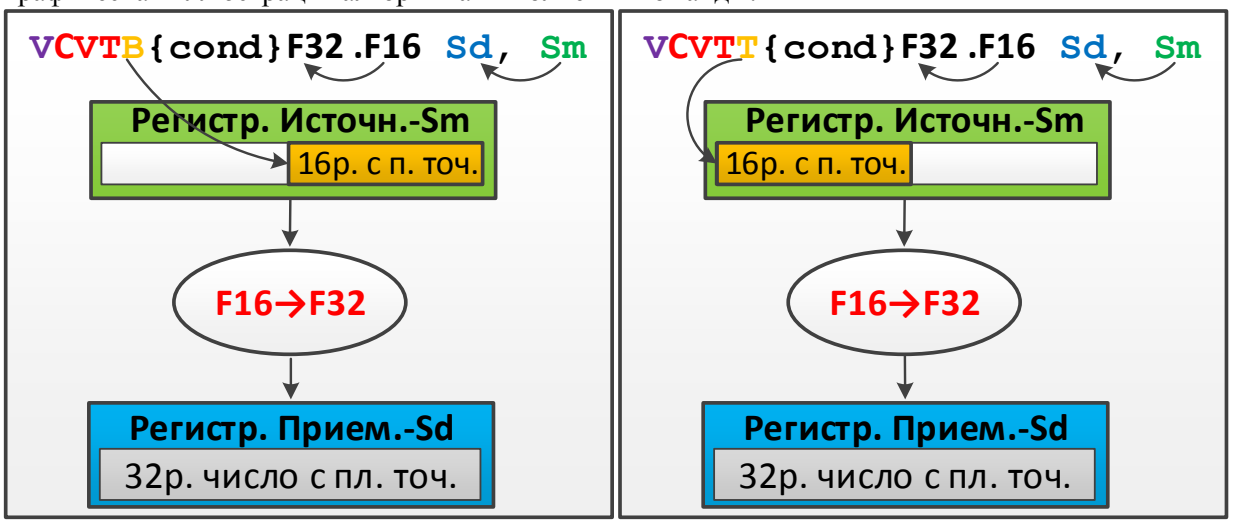
Графическая иллюстрация алгоритма выполнения команды:



Команда с суффиксом **.F32.F16:**

1. Конвертирует число с плавающей точкой половинной точности F16, расположенное в младшем полуслове (для суффикса «B») или в старшем полуслове (для суффикса «T») регистра-источника Sm в число с плавающей точкой однократной точности F32.
2. Сохраняет результат в регистре назначения Sd.

Графическая иллюстрация алгоритма выполнения команды:



Ограничения

Нет ограничений

Выработка флагов

Не оказывает влияния на флаги

Примеры

```
        VCVTB.F16.F32 S4, S5  ; Число с плавающей точкой однократной
                              ; точности из регистра S5 преобразовать
                              ; преобразовать в формат половинной точности
                              ; и сохранить в младшем слове регистра S4
        VCVTB.F32.F16 S9, S4  ; Обратное преобразование числа с плавающей
                              ; точкой половинной точности из младшего
                              ; слова регистра S4 в формат однократной
                              ; точности с сохранением в регистре S9
```

9 КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Оглавление

9.1. Прямая и косвенная передача управления 162
9.2. Сравнение и переход по нулю или не нулю 164
9.3. Команда начала блока условно выполняемых команд 165
9.4. Команды табличной передачи управления 168

9.1 Прямая и косвенная передача управления

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
B, BL, BX, and BLX	Branch instructions Команды передачи управления	v	v	v

Синтаксис

```
B{cond} label     Branch (immediate).
                  Передача управления по метке (непосредственному
                  адресу)
BL{cond} label    Branch with link (immediate).
                  Передача управления по метке (непосредственному
                  адресу)с обратной связью (сохранением адреса
                  возврата в регистре LR).
BX{cond} Rm       Branch indirect (register).
                  Косвенная передача управления по содержимому регистра.
BLX{cond} Rm      Branch indirect with link (register).
                  Косвенная передача управления по содержимому регистра
                  с обратной связью (сохранением адреса возврата в
                  регистре связи LR).
```

Где:

cond
код условного выполнения команды (опция)

label – Метка. Относительное выражение по отношению к текущему значению счетчика команд PC.

Rm – Регистр-источник, содержащий значение для загрузки в счетчик команд PC в операциях косвенной передачи управления. Бит 0 (Bit[0]) этого регистра должен содержать 1 для подтверждения режима работы с набором команд Thump при выполнении перехода. Этот бит будет автоматически сброшен при загрузке значения в счетчик команд PC.

Операция

Команды выполняют передачу управления по метке или по содержимому регистра Rm при косвенной передаче управления.
Если в команде присутствует суффикс «L» (Link), то адрес возврата, являющийся адресом следующей команды, сохраняется в регистре связи Link Register (LR) – R14. Команды BL и BLX по существу являются командами вызова подпрограмм с сохранением адреса возврата в регистре LR.

Если в команде присутствует суффикс «X», то это означает косвенный переход, то есть передачу управления по адресу, расположенному в регистре, имя которого указано в качестве операнда команды.

Внимание!

Проявляйте осторожность при работе с командами косвенной передачи управления BX и BLX. Адрес назначения в регистре Rm должен иметь нулевой бит, равный 1, - для подтверждения процессору режима работы с набором команд Thumb-2. Если он окажется равным 0, то возникнет исключение «Ошибка использования» UsageFault. Это связано с тем, что в предыдущих архитектурах ARM нулем передается указание процессору на переход в режим выполнения ARM-команд из режима Thumb. Для ядер Cortex-M3/M4/M4F возможен только один набор команд Thumb-2 (объединяющий наборы ARM и Thumb), хотя нулевой бит счетчика команд PC для совместимости по-прежнему управляет переключением из одного набора в другой.

Только одна команда условной передачи управления по метке `B{cond}label` может при наличии условия <cond> располагаться в любом месте программы (вне блока условного исполнения IT) или внутри него. Она становится в этом случае командой условной передачи управления и не требует дополнительной команды начала блока условного выполнения IT.
Все остальные команды могут быть условными только в том случае, когда они располагаются внутри блока IT. Вне блока IT команды не должны содержать условия выполнения <cond> .
Любые команды с суффиксами условного выполнения будут автоматически помещаться транслятором внутрь блоков условного выполнения с числом команд в блоке до 4-х. (См. более подробно в описании команды IT).

Диапазоны возможных относительных смещений для разных команд передачи управления

Команда	Диапазон возможных смещений
B label	от −16 MB до +16 MB
Bcond label (outside IT block)	Вне IT-блока: от −1 MB до +1 MB
Bcond label (inside IT block)	Внутри IT-блока: от −16 MB до +16 MB
BL{cond} label	от −16 MB до +16 MB
BX{cond} Rm	Переход по любому адресу в регистре Rm
BLX{cond} Rm	Переход по любому адресу в регистре Rm

Вы можете использовать в оптокоде команды суффикс «.W», для задания максимально большого диапазона смещений. В этом случае будет сгенерирована 32-разрядная команда с максимальным числом разрядов для размещения непосредственного смещения относительно текущего значения счетчика команд PC.

Ограничения

■ Нельзя использовать счетчик команд PC в командах косвенного вызова подпрограмм BLX.
■ Для корректного выполнения процессором команд косвенной передачи управления BX и BLX, нулевой бит (bit[0]) регистра Rm должен содержать 1. При загрузке данных в PC он будет автоматически сброшен в 0.
■ Если любая из этих команд включена в блок условного выполнения IT, то она должна быть последней инструкцией внутри блока IT.

Замечание.

Команда Bcond label является единственной из команд передачи управления, которая не требует включения в IT-блок. Однако, если она включена в IT-блок, то будет иметь максимально возможный диапазон относительных смещений.

Выработка флагов

Команда не изменяет значения флагов

Примеры

```
B loopA          ; Переход по метке loopA. Максимальное
                 ; смещение ±16MB.
BLE ng           ; Переход на метку ng при выполнении
                 ; условия LE («Меньше или эквивалентно»).
                 ; Максимальное смещение ±1MB
B.W target       ; Безусловный переход на метку target
                 ; с максимально возможным смещением ±16MB.
                 ; Аналог просто B target.
BEQ target       ; Переход на метку target при выполнении условия
                 ; EQ «Эквивалентно» с максимальным смещением
                 ; ±1MB
BEQ.W target     ; То же, но с максимальным смещением ±16MB
BL funC          ; Вызов функции funC с сохранением адреса
                 ; возврата в регистре связи LR.
                 ; Фактически - команда Call.
BX LR            ; Возврат из подпрограммы
                 ; Фактически команда Return.
BXNE R0          ; Косвенная передача управления по содержимому
                 ; регистра R0 при выполнении условия NE («Не
                 ; эквивалентно»).
BLX R0           ; Косвенный вызов подпрограммы (Call)по адресу,
                 ; находящемуся в регистре R0, с сохранением
                 ; адреса возврата в регистре связи LR.
```

9.2 Сравнение и переход по нулю или не нулю

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
CBZ и CBNZ	Compare and Branch if Zero, Compare and Branch if Non-Zero. Сравнение и переход, если ноль Сравнение и переход, если не ноль	V	V	V

Синтаксис

CBZ Rn, label **Сравнение и переход по нулю**
CBNZ Rn, label **Сравнение и переход по не нулю**

Где:	**Rn** Регистр, содержащий исходный операнд, значение которого тестируется на ноль
	label – Метка назначения при передаче управления. Относительное выражение по отношению к текущему значению счетчика команд PC.

Операция

Используйте команды CBZ и CBNZ для проверки состояния какого-либо регистра на ноль и передачи управления на нужную метку в зависимости от результата проверки (без изменения исходного состояния флагов).

Команда **CBZ Rn, label** не изменяет флагов результатов предыдущих операций, но эквивалентна выполнению следующих двух команд:

CMP Rn, #0
BEQ label

Команда **CBNZ Rn, label** не изменяет флагов результатов предыдущих операций, но эквивалентна выполнению следующих двух команд:

CMP Rn, #0
BNE label

Ограничения

- Регистр исходного операнда может быть только одним из младших регистров от R0 до R7.
- Переход возможен только вперед на расстояние от 4-х до 130 байт.
- Команда не может быть включена в блок Если-То (IT-блок).

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
CBZ R5, target  ; Переход вперед на метку target,
                ; если содержимое R5 равно 0.
CBNZ R0, target ; То же, если содержимое R0 не равно 0.
```

9.3 Команда начала блока условно выполняемых команд

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
IT	If-Then. Блок условного выполнения «Если-То»	v	v	v

Синтаксис

IT{x{y{z}}} cond

Где:

- **cond** Специфицирует *условие выполнения* для первой команды в блоке.
- **x** Специфицирует *переключатель условия выполнения* для второй команды в блоке IT.
- **y** Специфицирует *переключатель условия выполнения* для третьей команды в блоке IT.
- **z** Специфицирует *переключатель условия выполнения* для четвертой команды в блоке IT.
- Возможные значения переключателей условия выполнения второй, третьей и четвертой команд:

T	**Then** (То) – По тому же условию cond, что и первая команда блока IT.
E	**Else** (В противном случае) – По противоположному (оппозитному) условию относительно условия выполнения cond первой команды блока IT.

Замечание.
Можно в качестве основного условия cond указать AL (при всех условиях). В этом случае все команды внутри IT-блока должны быть безусловными, а все переключатели условий x, y, z должны быть установлены в состояние T или вообще отсутствовать, но ни в коем случае не быть E (в противном случае).

Операция

Команда IT-блока делает возможным условное выполнение вплоть до 4-х последующих команд с однотипными условиями (прямым и оппозитным ему). Например, в блоке IT могут быть команды условного выполнения с суффиксами GT и LE. Команды условного выполнения с «непарными» условиями должны быть включены в другой IT-блок.
В блоке указывается основное условие, которое является условием выполнения первой команды блока. Для последующих команд (2-й, 3-й и 4-й) указывается переключатель условия: T – то же самое условие, как и для первой команды, или E – логически противоположное.
В состав IT-блока будут входить команды условного выполнения, расположенные сразу после команды IT. Число таких команд определяется числом переключателей условий, которые указаны в качестве суффиксов в команде IT. Это могут быть, в том числе, команды передачи управления

(branches), которые имеют в мнемокоде команды суффикс условного выполнения {cond}. Команда перехода обязательно должна быть последней в IT-блоке.

Замечание.
Ваш транслятор с языка Ассемблер может быть в состоянии автоматически генерировать инструкцию блока условного выполнения IT, когда встретит в ассемблерном тексте команды условного выполнения, так что Вам нет необходимости писать эти инструкции самостоятельно. Детали смотрите в описании на Ассемблер, который используете!

Команда точки останова BKPT внутри IT-блока всегда выполняется, даже в том случае, когда ее условие выполнения ложно.
Исключение или прерывание может быть запущено на обработку между командой IT-блока и собственно «телом» блока, а также внутри IT-блока. Это означает, что блок условного выполнения IT является прерываемым. Когда управление передается соответствующему обработчику прерывания, адрес возврата сохраняется в регистре связи LR или стеке.
При этом обычные команды возврата из процедуры обслуживания прерывания/исключения обеспечивают корректный возврат управления внутрь IT-блока, который успешно завершается после обработки прерывания. Это единственный путь модификации содержимого счетчика команд PC, для того, чтобы перейти к команде внутри IT-блока.

Ограничения

Не допускается использование внутри блока IT следующих команд:
■ IT - не допускаются вложенные IT-блоки;
■ CBZ и CBNZ - команды проверки «на ноль» или на «не ноль» и передачи управления;
■ CPSID и CPSIE - команды изменения состояния процессора - запрещения или разрешения прерываний.

Для других команд, которые используются внутри блока IT, должны соблюдаться следующие ограничения:
■ Любая команда передачи управления (branch) или команда модификации текущего содержимого счетчика команд PC должна располагаться либо вне блока, либо должна быть последней командой IT-блока:
– ADD PC, PC, Rm
– MOV PC, Rm
– B, BL, BX, BLX
– Любая команда загрузки регистра LDR, множественной загрузки регистров LDM или восстановления содержимого регистров из стека POP, если в качестве регистра-назначения используется счетчик команд PC
– TBB и TBH (табличной передачи управления)

■ Запрещается использовать команды передачи управления на команды внутри IT-блока, за исключением восстановления адреса возврата в счетчик команд в конце процедуры обслуживания исключения/прерывания.
■ Все команды условного выполнения за исключением команды условной передачи управления **Bcond label** должны располагаться внутри IT-блоков. Об этом автоматически заботится хороший транслятор.
■ Команда условной передачи управления Bcond label может располагаться как во вне, так и внутри IT-блока. В последнем случае она имеет максимально возможный диапазон смещений адреса ± 16 MB относительно адреса текущей команды.
■ Каждая команда внутри IT-блока должна быть командой условного выполнения и содержать суффикс условного выполнения либо точно такой же, как в условии cond блока IT для первой команды, либо логически противоположный. То есть, если основное условие EQ, то все четыре команды внутри блока должны содержать либо суффикс EQ, либо NE.

Замечание.
Некоторые трансляторы с Ассемблера накладывают дополнительные ограничения на использование блоков условного выполнения IT, например, запрещают использовать внутри блоков IT директив Ассемблера.

Выработка флагов

Команда не изменяет значения флагов

Примеры

```
      ITTE NE                 ; Условное выполнение трех следующих команд.
      ANDNE R0, R0, R1        ; В 1-й команде ANDNE должно использоваться
                              ; основное условие NE блока IT.
      ADDSNE R2, R2, #1       ; Во 2-й команде ADDSNE основное условие
                              ; подтверждается – суффикс «T» в команде
                              ; блока IT.
                              ; Эта команда имеет в оптокоде
                              ; дополнительный суффикс «S», меняющий
                              ; состояние флагов.
      MOVEQ R2, R3            ; В 3-й команде используется инверсное
                              ; основному условие EQ.

; Конвертирование Hex-значения в регистре R0 в диапазоне от 0 до 15
; в ASCII-код соответствующего символа('0'-'9') или ('A'-'F')
; Для числа в диапазоне '0'-'9' – добавить #55
; Для числа в диапазоне 'A'-'F' – добавить #48
      CMP R0, #9              ; Число больше 9?, т.е. ('A'-'F')

      ITE GT                  ; Следующие две команды включены в блок
                              ; условного выполнения IT.
      ADDGT R1, R0, #55       ; Получить ASCII-код 0xA -> 'A'.
      ADDLE R1, R0, #48       ; Получить ASCII-код 0x0 -> '0'.

      IT GT                   ; IT-блок автоматически создается
                              ; транслятором даже для одной единственной
                              ; команды условного выполнения.
      ADDGT R1, R1, #1        ; Условное инкрементирование регистра R1.

      ITTEE EQ                ; Блок условного выполнения из 4-х команд.
      MOVEQ R0, R1            ; Условная загрузка R0 из R1 (1-я команда).
      ADDEQ R2, R2, #10       ; Условное сложение (2-я команда).
      ANDNE R3, R3, #1        ; Логическое И по оппозитному условию (3-я).
      BNE.W dloop             ; Условная передача управления (4-я).
                              ; Суффикс «.W» расширяет диапазон возможных
                              ; смещений до максимального ± 16 MB.

      IT NE                   ; Следующая команда должна быть командой
                              ; условного выполнения
      ADD R0, R0, R1          ; Так как суффикс условного выполнения «NE»
                              ; отсутствует, а команда IT-блока есть, то
                              ; выдается сообщение о синтаксической ошибке
                              ; Syntax error:
                              ; «no condition code used in IT block».
                              ; не указан код условия, использованный в
                              ; IT-блоке
```

9.4 Команды табличной передачи управления

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
TBB и TBH	Table Branch Byte and Table Branch Halfword. Табличный относительный переход по байтовому смещению. Табличный относительный переход по смещению-полуслову.	v	v	v

Синтаксис

```
TBB [Rn, Rm]
TBH [Rn, Rm, LSL #1]
```

Где:

Rn *Регистр базы*, содержащий базовый адрес таблицы длин переходов(table of branch lengths).
Если этот регистр является счетчиком команд Program Counter (PC - R15), то базовый адрес таблицы - это адрес байта, расположенного сразу за командой табличного перехода TBB или TBH.Таблица смещений должна быть расположена сразу за командой TBB, TBH

Rm *Индексный регистр*, содержащий индекс - порядковый номер в таблице смещений. Для таблицы смещений-полуслов (TBH) индекс должен быть удвоен, что делается попутной операцией логического сдвига влево на один разряд (Rm, LSL #1)= i x 2.

Операция

Эти команды выполняют относительный переход вперед по отношению к текущему состоянию счетчика команд PC на величину смещения, заданного в таблице 8-разрядных байтовых смещений для команды TBB или в таблице 16-разрядных смещений – полуслов для команды TBH.
Отличие команд друг от друга – диапазон возможных смещений (для TBH – вдвое больший).
Регистр Rn является базовым регистром – указателем на начало таблицы смещений.
Регистр Rm является индексным регистром, позволяющим выбрать из таблицы смещений нужную величину смещения по индексу.
Для команды TBB: из таблицы извлекается байт без знака и умножается на 2:
offset=byte*2.
Для команды TBH: из таблицы извлекается полуслово без знака и умножается на 2:
offset=halfword*2.
Полученное значение смещения добавляется к текущему значению счетчика команд PC и выполняется относительный переход на нужный блок (фрагмент) программы
B [PC, #offset].

Ограничения

- Rn не должен быть указателем стека SP.
- Rm не должен быть указателем стека SP или счетчиком команд PC.
- Если любая из этих двух команд табличной передачи управления находится внутри блока условного выполнения IT, то она должна быть последней командой блока.

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
; Загрузить начальный адрес байтовой таблицы смещений в регистр R0
      ADR.W R0, BranchTable_Byte
; Выполнить переход на программный блок в зависимости от номера блока,
; ранее записанного в индексный регистр R1 (i - номер блока)
      TBB [R0, R1]
```

```
Case0
;     …
; Последовательность команд для i=0
Case1
;     …
; Последовательность команд для i=1
Case2
;     …
; Последовательность команд для i=2

; Таблица байтовых смещений
BranchTable_Byte
      DCB 0                       ; Вычислить смещение для Case0
      DCB ((Case1-Case0)/2)       ; Вычислить смещение для Case1.
      DCB ((Case2-Case0)/2)       ; Вычислить смещение для Case2.
; Смещение, полученное из таблицы, будет автоматически удваиваться,
; поэтому адресный диапазон делится на два.
; Полученное значение (путем вычисления на стадии трансляции)
; сохраняется в кодовой памяти как байтовая константа с помощью
; директивы Ассемблера DCB (определить код байта).

; Табличная передача управления по таблице смещений-полуслов,
; расположенных сразу за командой TBH. Базовый адрес таблицы
; смещений – содержимое счетчика команд PC.
      TBH [PC, R1, LSL #1]

BranchTable_H
      DCI ((CaseA - BranchTable_H)/2) ; Вычислить смещение для CaseA.
      DCI ((CaseB - BranchTable_H)/2) ; Вычислить смещение для CaseB
      DCI ((CaseC - BranchTable_H)/2) ; Вычислить смещение для CaseC.
CaseA
;     …
; Тело программного блока CaseA
CaseB
;     …
; Тело программного блока CaseB
CaseC
;     …
; Тело программного блока CaseC

; Директива Ассемблера DCI размещает константы-полуслова в кодовой
; памяти. Такую же функцию выполняет директива DCW.
```

10 КОМАНДЫ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

Оглавление

10.1. Точка останова. Переход в режим отладки 170
10.2. Изменение состояния процессора.....171
10.3. Синхронизация доступа к данным.....172
10.4. Пересылка данных из регистров специального назначения в регистр общего назначения и обратно 173
10.5. Пустая команда.....175
10.6. Посылка события.....175
10.7. Вызов супервизора 176
10.8. Ожидание события или прерывания 177

10.1 Точка останова. Переход в режим отладки

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
BKPT	Breakpoint – Точка останова	v	v	v

Синтаксис

BKPT #imm

Где:	**imm** Выражение, значение которого может быть целым числом без знака в диапазоне от 0 до 255.

Операция

Команда точки останова BKPT вызывает переход процессора в состояние отладки Debug state. Программно-аппаратные средства отладки могут использовать эту команду для исследования текущего состояния системы, когда точка останова будет достигнута.
Значение непосредственного операнда imm игнорируется процессором. Оно используется отладчиком для получения дополнительной информации о точке останова.
Команда BKPT может быть включена в IT-блок, однако она будет выполнена безусловно, да она и не имеет опции условного выполнения. Условие, указанное в команде IT-блока никакого влияния на эту команду не оказывает.

Ограничения

Нет ограничений

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
BKPT 0xAB  ; Точка останова с непосредственным операндом 0xAB
           ; Отладчик может прочитать значение этой переменной
           ; с использованием адреса в счетчике команд PC
           ; для идентификации типа точки останова
```

10.2 Изменение состояния процессора

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
CPS	Change Processor State – Изменить состояние процессора	v	v	v

Синтаксис

CPS*effect iflags*

Где: ***effect*** Суффикс разрешения/запрещения прерываний.

IE	Clears the special-purpose register – Очистить регистр специального назначения – Разрешить прерывания
ID	Sets the special-purpose register – Установить регистр специального назначения – Запретить прерывания

iflags Последовательность из одного или более флагов:

i	Set or clear the Priority Mask Register (**PRIMASK**). Установить или очистить регистр масок приоритетов.
f	Set or clear the Fault Mask Register (**FAULTMASK**). Установить или очистить регистр масок ошибок

Операция

Команда CPS изменяет состояние процессора, модифицируя значения в регистрах специального назначения PRIMASK (масок прерываний) и FAULTMASK (масок ошибок). Более подробно см. в «Stellaris® Data Sheet».

Ограничения

■ Используйте команду CPS исключительно в привилегированном программном обеспечении. Если она используется в непривилегированном режиме, то не оказывает никакого эффекта.
■ Команда CPS не может быть условной и поэтому не может быть включена в IT-блок условного выполнения.

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
CPSID i     ; Запрещение прерываний (Disable interrupts) и
            ; конфигурируемых обработчиков ошибок
            ; configurable fault handlers (set PRIMASK).
CPSID f     ; Запрещение прерываний (Disable interrupts) и
            ; всех обработчиков ошибок
            ; all fault handlers (set FAULTMASK).
CPSIE i     ; Разрешение прерываний (Enable interrupts) и
            ; конфигурируемых обработчиков ошибок
            ; configurable fault handlers (clear PRIMASK).
CPSIE f     ; Разрешение прерываний (Enable interrupts) и
            ; обработчиков ошибок
            ; fault handlers (clear FAULTMASK).
```

10.3 Синхронизация доступа к данным

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
DMB	Data Memory Barrier. Барьер на доступ к памяти данных.	v	v	v

Синтаксис

```
DMB{cond}
```

Где: `cond` Опциональный суффикс условного выполнения команды

Операция

Команда DMB выставляет барьер доступа к памяти данным. Ее выполнение гарантирует, что все сделанные ранее обращения к памяти (до выполнения команды постановки барьера DMB) будут завершены в программном порядке, прежде чем новая команда доступа к памяти, расположенная после команды DMB, будет выполнена (в программном порядке).
Команда DMB не влияет на порядок выполнения других команд, которые не связаны с доступом к памяти.

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
      DMB   ; Установить барьер на доступ к памяти данным
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
DSB	Data Synchronization Barrier. Барьер синхронизации данных.	v	v	v

Синтаксис

```
DSB{cond}
```

Где: `cond` Опциональный суффикс условного выполнения команды

Операция

Команда DSB устанавливает специальный барьер синхронизации данных при доступе к памяти. Все команды, которые расположены за этой командой (в программном порядке) не будут выполняться до тех пор, пока выполнение команды синхронизации DSB не завершится полностью. Это произойдет только после того, как все явные доступы к памяти до этой инструкции DSB будут выполнены.

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
      DSB   ; Установить барьер синхронизации данных
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
ISB	Instruction Synchronization Barrier. Барьер синхронизации инструкций.	v	v	v

Синтаксис

```
ISB{cond}
```

Где: `cond Опциональный суффикс условного выполнения команды`

Операция

Команда ISB устанавливает барьер синхронизации инструкций. После завершения ее выполнения происходит «смыв» - очистка конвейера команд и все последующие команды (после команды ISB) выбираются не с конвейера, а снова из кэш-памяти (если она есть) или напрямую из памяти.

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
ISB ; Барьер синхронизации инструкций
```

10.4 Пересылка данных из регистров специального назначения в регистр общего назначения и обратно

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MRS	Move the contents of a special register to a general-purpose register. Пересылка содержимого регистра специального назначения в регистр общего назначения.	v	v	v

Синтаксис

```
MRS{cond} Rd, spec_reg
```

Где:
- `cond - Опциональный суффикс условного выполнения команды`
- `Rd - Регистр общего назначения - приемник данных`
- `spec_reg - Регистр специального назначения, источник данных, один из следующих регистров:`
  APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR,
  PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, CONTROL

Операция

Используйте последовательность команд MRS … MSR – «Чтение-Модификация-Запись» для обновления PSR (регистра статуса программы), например, для очистки флага насыщения Q.
В процессе обновления кода в некоторых битах регистра статуса, остальные биты, которые отвечают за состояние программного процесса, должны быть сохранены. Это достигается тем, что сначала состояние регистра специального назначения считывается в регистр общего назначения процессора, затем нужные биты модифицируются с использованием логических команд, а уже после этого, - модифицированное значение сохраняется в регистре специального назначения.
Команда MRS выполняет функцию сохранения текущего состояния (state-saving), а команда MSR – восстановления состояния после модификации (state-restoring)
Замечание.
Имя регистра BASEPRI_MAX является псевдонимом имени BASEPRI при использовании команды чтения из регистра специального назначения MRS. См. также описание команды MSR.

Ограничения

Регистр-приемник данных Rn не может быть указателем стека SP или счетчиком команд PC.

Выработка флагов

Команда не изменяет состояния флагов

Примеры

```
MRS R0, PRIMASK ; Считать содержимое регистра масок приоритетов
                ; прерываний PRIMASK в регистр R0.
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
MSR	Move the contents of a general-purpose register to a special register. Пересылка содержимого регистра общего назначения в регистр специального назначения.	V	V	V

Синтаксис

MSR{cond} spec_reg, Rn

Где:	cond – Опциональный суффикс условного выполнения команды
	Rd – Регистр общего назначения – источник данных
	spec_reg – Регистр специального назначения, приемник данных, один из следующих регистров: **APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, CONTROL**

Операция

Доступ к регистрам специального назначения по записи с использованием команды MSR зависит от текущего уровня привилегий (privilege level):

- Если процессор работает в непривилегированном режиме Unprivileged (пользовательском режиме), то доступ по записи возможен только к Регистру Статуса Программы Приложения Application Program Status Register (APSR).
- Если процессор работает в привилегированном режиме Privileged, то доступ возможен ко всем регистрам специального назначения.

Замечание.
Если запись производится в регистр BASEPRI_MAX, то запись выполняется в регистр BASEPRI лишь в следующих случаях:
■ Содержимое регистра источника Rn не равно нулю и текущее значение в регистре BASEPRI равно нулю.
■ Содержимое регистра источника Rn не равно нулю и меньше (less) чем текущее значение в регистре BASEPRI.

Ограничения

Регистр-источник данных Rn не может быть указателем стека SP или счетчиком команд PC.

Выработка флагов

Команда обновляет состояние флагов в соответствии со значениями в регистре-источнике Rn.

Примеры

```
MSR CONTROL, R1 ; Записать текущее значение в регистре R1
                ; в регистр управления CONTROL.
```

10.5 Пустая команда

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
NOP	No Operation. Нет операции	v	v	v

Синтаксис

NOP{cond}

Где: cond - Опциональный суффикс условного выполнения команды

Операция

Команда NOP не выполняет никаких действий. Не может использоваться для реализации программной задержки. Процессор может удалить ее с конвейера команд, прежде чем она достигнет стадии выполнения. Используйте ее исключительно в качестве шаблона, куда Вы в последующем можете вставить нужную инструкцию, например, для выравнивания следующей команды по 64-битной границе (границе двойного слова).

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов.

Примеры

```
        NOP   ; Нет операции
```

10.6 Посылка события

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SEV	Send Event. Посылка события.	v	v	v

Синтаксис

SEV{cond}

Где: cond - Опциональный суффикс условного выполнения команды

Операция

Команда SEV (посылка события) устанавливает первый бит в регистре событий event register (относится к регистрам управления питанием "Power Management"). Это сигнал для всех других процессоров, которые входят в мульти-микропроцессорную систему, посылаемый программным путем (уведомление).

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов.

Примеры

```
        SEV   ; Посылка события
```

10.7 Вызов супервизора

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
SVC	Supervisor Call. Вызов супервизора.	v	v	v

Синтаксис

SVC{cond} #imm

Где:
- cond - Опциональный суффикс условного выполнения команды
- imm - Выражение, значение которого может быть целым числом в диапазоне от 0 до 255 (8-разрядным целым без знака).

Операция

Команда SVC вызывает супервизор – подпрограмму-обработчик исключения по вызову супервизора. Непосредственный операнд imm игнорируется процессором. Если это необходимо, то он может расцениваться как номер сервиса, который должен быть предоставлен супервизором по текущему вызову.
Фактически – программный вызов исключения.

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов.

Примеры

```
SVC 0x32    ; Вызов программы-супервизора (Supervisor Call)
            ; Супервизор (SVC handler) может распаковать
            ; непосредственный операнд по значению PC, которое
            ; сохранено в стеке (считывая из кодовой памяти код
            ; операции и выделяя в нем значение 0x32).
```

10.8 Ожидание события или прерывания

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
WFE	Wait For Event. Ожидание события.	v	v	v

Синтаксис

WFE{cond}

Где: cond - Опциональный суффикс условного выполнения команды

Операция

Команда WFE является командой ожидания события. Если первый бит в регистре событий (event register) равен нулю, то выполнение программы приостанавливается до тех пор, пока не произойдет одно из следующих событий:

■ Возникнет прерывание/исключение, которое не замаскировано в регистрах масок прерываний/исключений (PRIMASK, FAULTMASK, и BASEPRI) или установленным текущим уровнем приоритетов прерываний.
■ Произойдет установка флага ждущего обслуживания прерывания (Pending state), если бит SEVONPEND (послать событие по флагу ждущего обслуживания прерывания) в регистре управления системой System Control Register (SCR) установлен.
■ Возникнет запрос по отладке (Debug request), если отладка (Debug) разрешена.
■ Возникнет событие, которое сигнализирует периферии или другим процессором в мульти-микропроцессорной системе о том, что событие произошло с помощью команды посылки события SEV.
Если содержимое регистра событий (event register) равно 1, то команда WFE очищает этот бит в 0 и возвращает управление следующей команде. Более подробно в разделе «Управление питанием» ("Power Management" в Stellaris® Data Sheet).

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов.

Примеры

```
        WFE   ; Ожидание события
```

Мнемокод	Название команды	Используется в:		
		M3	M4	M4F
WFI	Wait for Interrupt. Ожидание прерывания.	v	v	v

Синтаксис

WFI{cond}

Где: cond - Опциональный суффикс условного выполнения команды

Операция

Команда WFI приостанавливает выполнение программы до тех пор, пока не произойдет одно из следующих событий:

- Прерывание/Исключение.
- Запос на отладку (Debug request), независимо от того, разрешена ли отладка (Debug).

Ограничения

Нет

Выработка флагов

Команда не изменяет состояния флагов.

Примеры

```
        WFI   ; Ожидание прерывания
```