ТЕОРИЯ (с 299 стр учебника)

Способы адресации операндов. Процессор использует способы адресации операндов, представленные в табл. 17.4.

При *непосредственной адресации* операнд Imm входит в состав команды. При *регистровой адресации* в команде задается имя регистра Rn, содержимое которого является операндом или результатом операции.

Особенностью системы команд ARM является наличие *регистровой адресации с масштабированием*. При масштабировании содержимое регистра Rn сдвигается на число разрядов n, указанное в команде (от 1 до 31). Вместо shift в ассемблерном тексте используется одно из обозначений, задающих вид реализуемого сдвига: LSL — логический сдвиг влево, LSR — логический сдвиг вправо, ASR — арифметический сдвиг вправо, ROR — циклический сдвиг вправо.

Таким образом, различные сдвиги реализуют с помощью команды пересылки MOV, используя регистровую адресацию с масштабированием. При выполнении арифметических и логических сдвигов

последний выдвигаемый разряд поступает в регистр CPSR в качестве признака переноса C.

При циклическом сдвиге бит C включается в цепь сдвига, только если число разрядов сдвига задано равным 0. В этом случае выполняется циклический сдвиг операнда на один разряд вправо через бит C в регистре CPSR.

При *косвенно-регистровой адресации* указанный регистр Rn содержит адрес ячейки памяти, где хранится операнд или результат.

Индексная адресация имеет две разновидности — преиндексная и постиндексная. В случае *преиндексной адресации* адресом служит содержимое базового регистра Rn, которое индексируется перед выполнением операции путем сложения или вычитания непосредственного операнда Imm, содержимого регистра Rm или масштабированного содержимого Rm. Если в поле операнда содержится символ {!}, то индексированное содержимое Rn сохраняется после выполнения операции. В случае *постиндексной адресации* адресом служит содержимое базового регистра Rn, которое индексируется после выполнения операции.

Таблица 17.4. **Способы адресации**

Обозначение	Название
#Imm	Непосредственная
Rn	Регистровая
Rn, shift #n	Регистровая с масштабированием
[Rn]	Косвенно-регистровая
[Rn, ±Imm] {!}	Преиндексная с непосредственным смещением
[Rn, ±Rm] {!}	Преиндексная с регистровым смещением
[Rn, ±Rm, shift #n]	Преиндексная с масштабированным регистровым смещением
[Rn], ±Rm	Постиндексная с регистровым смещением
[Rn], ±Rm, shift #n	Постиндексная с масштабированным регистровым смещением

Таблица 17.5. **Суффиксы для организации условных операций**

Суффикс	Логическое выражение	Условие
EQ	Z = 1	Равно
NE	Z = 0	Не равно
CS	C = 1	Выше или равно
CC	C = 0	Ниже
MI	N = 1	Отрицательный результат
PL	N = 0	Положительный результат
VS	V = 1	Переполнение
VC	V = 0	Нет переполнения
HI	C = 1, Z = 0	Выше
LS	C = 0, Z = 1	Ниже или равно
GE	N = V	Больше или равно
LT	N ≠ V	Меньше
GT	Z = 0, N = V	Больше
LE	Z = 1, N ≠ V	Меньше или равно
AL	—	Всегда

Таблица 17.6. **Команды пересылки**

Мнемокод	Команда	Операция
LDM	Групповая загрузка содержимого регистров из памяти	—
LDR	Загрузка регистра из памяти	Rd := <адрес>
MOV	Пересылка регистра или константы	Rd := Op2
MRS	Пересылка из CPSR или SPSR в регистр	Rn := CPSR (SPSR)
MSR	Пересылка из регистра в CPSR или SPSR	CPSR (SPSR) := Rm
MVN	Пересылка с побитовой инверсией	Rd := HE Op2
STM	Групповое сохранение содержимого регистров в памяти	—
STR	Пересылка из регистра в память	<адрес> := Rn
SWP	Обмен содержимого регистра и памяти	Rd := [Rn], [Rn] := Rm

Таблица 17.7. **Команды арифметических и логических операций**

Мнемоника	Описание команды	Операция
ADC	Сложение с учетом переноса	Rd := Rn + Op2 + C
ADD	Сложение	Rd := Rn + Op2
AND	Логическое И	Rd := Rn AND Op2
BIC	Очистка битов (маскирование)	Rd := Rn AND (HE Op2)
CMN	Сравнение с отрицательным числом	CPSR flags := Rn + Op2
CMP	Сравнение	CPSR flags := Rn – Op2
EOR	Исключающее ИЛИ	Rd := Rn XOR Op2
MLA	Умножение с накоплением	Rd := (Rm × Rs) + Rn
MUL	Умножение	Rd := Rm × Rs
ORR	Логическое ИЛИ	Rd := Rn OR Op2
RSB	Обратное вычитание	Rd := Op2 – Rn
RSC	Обратное вычитание с заемом	Rd := Op2 – Rn – 1 + C
SBC	Вычитание с заемом	Rd := Rn – Op2 – 1 + C
SUB	Вычитание	Rd := Rn – Op2
TEQ	Побитовое сравнение	CPSR flags := Rn XOR Op2
TST	Тестирование битов	CPSR flags := Rn AND Op2

Таблица 17.8. **Команды передачи управления**

Мнемокод	Команда	Операция
B	Переход	PC := PC + (rel << 1)
BL	Переход с сохранением адреса возврата в LR	LR := PC, PC := PC + (rel << 1)
BX	Переход с возможностью смены состояния	PC := Rn, T := Rn[0]
SWI	Программное прерывание	Режим Supervisor, LR := PC, PC := 0x0008

Таблица 17.9. **Команды поддержки сопроцессора**

Мнемокод	Команда	Операция
CDP	Выполнение команды сопроцессором	—
LDC	Загрузка данных в сопроцессор из памяти	cRn := <адрес>
MCR	Пересылка из регистра процессора в регистр сопроцессора	cRn := Rn {cRm}
MRC	Пересылка из регистра сопроцессора в регистр процессора	Rn := cRn {cRm}
STC	Сохранение регистра сопроцессора в памяти	<адрес> := cRn

Если хочется добавить красивых слов:

1) отсутствие аппаратной поддержки стека. Стек организуется программно, причем в качестве указателя стека рекомендуется использовать регистр R13, хотя в принципе в этой роли может быть любой другой регистр общего назначения. Обычно обращение к стеку производится с помощью команд групповой пересылки STM и LDM;

2) установка в регистре CPSR признаков N, Z, C, V по результатам выполнения операций производится при наличии в команде суффикса S. При его отсутствии признаки в регистре CPSR не изменяются;

3) любая команда может быть условной, если ее снабдить соответствующим суффиксом. Виды условий и их суффиксы приведены в табл. 17.5. При наличии суффикса AL команда выполняется безусловно — при любых значениях признаков. Условия «Выше», «Ниже», «Выше или равно», «Ниже или равно» используются при сравнении чисел без знака, условия «Больше», «Меньше», «Больше или равно», «Меньше или равно» — при сравнении чисел со знаком.

Из группы команд пересылки (табл. 17.6) интересны команды групповой загрузки (сохранения) содержимого регистров LDM и STM. Они позволяют пересылать содержимое нескольких регистров, перечисленных в команде. В формате команды есть 16-битовое поле, в котором каждый бит соответствует одному из регистров R15 — R0. Если бит равен 1, то содержимое данного регистра сохраняется в памяти (по команде STM) или загружается из памяти (по команде LDM).

При записи в регистр CPSR с помощью команды MSR в режиме *User* изменяются только признаки N, Z, V, C. Остальные биты сохраняют ранее установленное значение. В любом режиме изменение бита T с помощью команды MSR не реализуется. Для программной смены состояния процессора используется команда перехода с возможностью смены состояния BX.

При выполнении арифметических и логических операций (табл. 17.7) один из операндов размещается в регистре, второй Op2 — в регистре,

ячейке памяти или задается непосредственно. Результат всегда размещается в регистре Rd.

Команда обратного вычитания RSB позволяет изменить порядок записи операндов в команде вычитания. Она соответствует команде «Вычесть из операнда Op2 содержимое регистра Rn и записать в регистр Rd». Операция CMN позволяет сравнить два операнда, у одного из которых при сравнении изменяется знак.

Ядро ARM выполняет несколько разновидностей операций умножения. Существуют две основные команды — простое умножение MUL и умножение с накоплением MLA. Любая из этих команд может иметь либо короткую форму (результат записывается в один регистр с потерей младших разрядов), либо длинную (произведение записывается в два регистра). При записи команды длинная форма обозначается суффиксом L. В первом случае операции умножения можно производить как со знаковыми, так и с беззнаковыми целыми числами. Длинная форма команды умножения имеет два варианта — знаковый и беззнаковый, которые отличаются префиксом S и U соответственно. Таким образом, команда умножения имеет следующие разновидности: MUL, SMULL, UMULL. Аналогично, команда умножения с накоплением имеет разновидности MLA, SMLAL, UMLAL.

Отметим, что процессор не выполняет операцию деления, которая реализуется программным путем (обычно путем вызова соответствующей подпрограммы).

Команды передачи управления (табл. 17.8) служат для изменения хода программы. Команда перехода B с соответствующим суффиксом (см. табл. 17.5) обеспечивает выполнение условных или безусловных переходов. Переход к подпрограмме осуществляется командой BL, при этом текущее содержимое программного счетчика PC (адрес возврата) сохраняется в регистре связи LR. В качестве операнда в команде задается 24-разрядное смещение rel, которое сдвигается на один разряд влево и после знакового расширения добавляется к текущему содержимому PC. При вызове вложенных подпрограмм необходимо программным путем организовать сохранение промежуточных значений адресов возврата (содержимого LR) в стеке, используя регистр SP в качестве указателя. Также программным путем обеспечивается, в случае необходимости, сохранение в стеке содержимого регистров. Заданный адрес перехода должен быть четным, если процессор находится в состоянии Thumb, или кратным четырем — в состоянии ARM.

Команда BX позволяет осуществлять переход с одновременным изменением состояния процессора. Адрес перехода (четный или кратный четырем) определяется разрядами 31 — 1 или 31 — 2 содержимого регистра Rn, заданного в команде, а состояние процессора — нулевым битом этого регистра, который копируется в регистр CPSR в качестве бита T.

Команда программного прерывания SWI используется для доступа к функциям операционной системы. При выполнении данной команды процессор переходит в режим Supervisor, запоминает адрес возврата в регистре LR и переходит на адрес 0x0008. По этому адресу располагается команда перехода к обработчику прерывания. Отметим, что выполнение команды SWI — единственный способ, позволяющий перевести процессор из режима User в привилегированный режим Supervisor.

Специальная группа команд (табл. 17.9) обеспечивает совместную работу процессора с сопроцессором, который обычно представляет собой специализированный блок, изготовленный на одном кристалле с процессором. К процессорному ядру ARM может быть подключено до 16 сопроцессоров, каждый из которых может иметь до 16 собственных регистров cRn. Данные, передаваемые сопроцессору по внутренней шине, включают в себя номер сопроцессора, номера регистров сопроцессора, код операции и поле дополнительной информации. Сопроцессор, приняв команду, исполняет ее и выдает на шину результат. При этом один из регистров сопроцессора cRn выполняет функцию регистра команд, используя записанные в него данные как код операции. После вывода в этот регистр кода операции процессор должен послать в сопроцессор команду CDP. Ввод-вывод данных в регистры сопроцессора реализуется с помощью команд LDC, STC, MCR, MRC.

```
add r0,r1,r2     @ поместить в регистр r0 сумму значений регистров r1 и r2
sub r5,r4,#7     @ поместить в регистр r5 разность (r4-7)
```

Примеры

```
UMULL   R1,R4,R2,R3      ; R4,R1:=R2*R3
UMLALS  R1,R5,R2,R3      ; R5,R1:=R2*R3+R5,R1, также разрешить
                         ; воздействие на флаги CPSR выполнения (S)
```

Примеры

```
MUL     R1,R2,R3         ; R1 := R2*R3
MLAEQS  R1,R2,R3,R4      ; Выполнить по условию EQ: R1 := R2*R3 + R4
                         ; и разрешить воздействие на флаги CPSR выполнения (S)
```

Табл. 27. Команды условного перехода

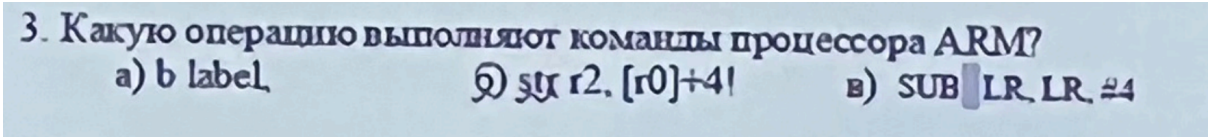
L	THUMB ассемблер	ARM эквивалент	Действия
0000	BEQ метка	BEQ метка	Переход, если установлен Z (равно)
0001	BNE метка	BNE метка	Переход, если сброшен Z (не равно)
0010	BCS метка	BCS метка	Переход, если установлен C (выше или равно)
0011	BCC метка	BCC метка	Переход, если сброшен C (ниже)
0100	BMI метка	BMI метка	Переход, если установлен N (минус)
0101	BPL метка	BPL метка	Переход, если сброшен N (плюс или равно)
0110	BVS метка	BVS метка	Переход, если установлен V (переполнение)
0111	BVC метка	BVC метка	Переход, если сброшен V (нет переполнения)
1000	BHI метка	BHI метка	Переход, если установлен C и сброшен Z (выше)
1001	BLS метка	BLS метка	Переход, если сброшен C или установлен Z (ниже или равно)
1010	BGE метка	BGE метка	Переход, если установлен N и установлен V или сброшен N и сброшен V (больше или равно)
1011	BLT метка	BLT метка	Переход, если установлен N и сброшен V или сброшен N и установлен V (меньше)
1100	BGT метка	BGT метка	Переход, если сброшен Z и либо установлен N и сброшен V, либо сброшен N и установлен V (больше)
1101	BLE метка	BLE метка	Переход, если установлен Z, или установлен N и сброшен V, или сброшен N и установлен V (меньше или равно)

В обеих командах умножения с накоплением (UMLAL и SMLAL) используют два 32-битных регистра, которые умножаются друг на друга, и к полученному результату добавляется 64-битное слагаемое: RdHi,RdLo := Rm*Rs + RdHi,RdLo. Младшие 32 бита добавляемого к результату умножения слагаемого вычитываются их регистра RdLo, а старшие 32 бита - из регистра RdHi. После чего младшие 32 бита результата заносятся в регистр RdLo, а старшие 32 бита - в регистр RdHi.

Команды MULL и MLAL позволяют выполнить целочисленное умножение двух 32-битных операндов и получить 64-битный результат. Умножение со знаком и без знака может быть как с накоплением, так и без него. Поэтому существуют четыре разновидности умножения с 64-битным результатом.

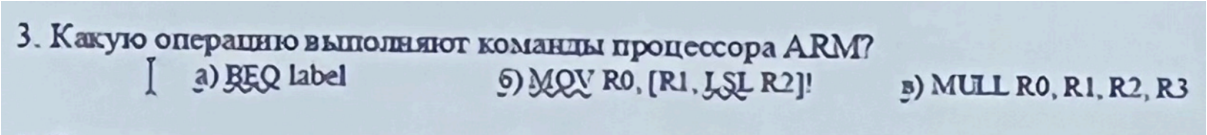
https://marsohod.org/projects/marsohod2/amber-arm-soc/226-arm-instr
http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/arm_thumb/18.htm
http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/asm_arm/mull_mlal.htm
система команд - http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/asm_arm/survey.htm
https://www.keil.com/support/man/docs/armasm/armasm_dom1361289861747.htm

ЗАДАНИЯ

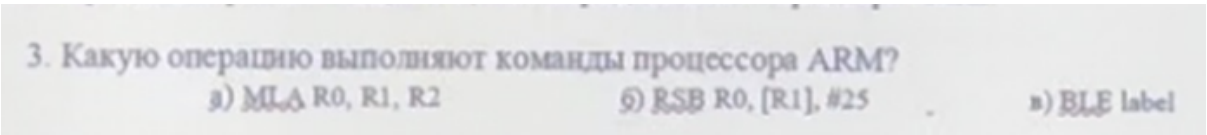


a) безусловный переход по метке label(24раздрядное смещение сдвигается на 1 разряд влево и после знакового расширения добавляется к PC)
b) пересылка из регистра в память, адрес памяти в регистре r0, который проиндексируется после выполнения операции тк постиндексная адресация

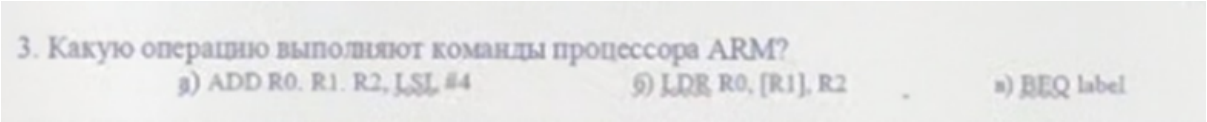
c) в регистр lr разность lr - 4



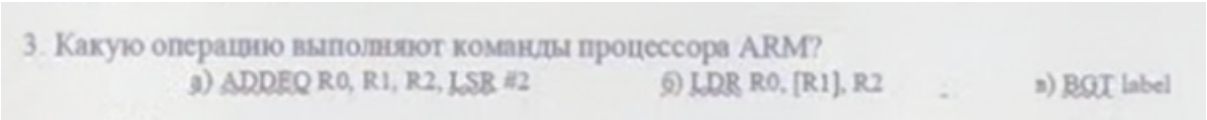
a) переход если равно
b) пересылка в регистр ro значения, адрес которого определяется содержимым регистра r1, сдвинутого влево на количество разрядов, равное содержимому регистра r2, при этом полученный адрес сохраняется
c) умножение в p0 пойдет старшая часть a в p1 младшая часть умножения p2 на p3



a) умножение с накоплением, к полученному умножению добавляется слагаемое
b) в регистр r0 помещается результат вычитания из 25 значения по адресу, равному содержимому регистра r1
c) переход, если меньше или равно (Z = 1, N != V)



a) в регистр r0 помещается результат сложения содержимого регистра r1 и содержимого регистра r2, сдвинутого влево на четыре разряда
b) загрузка в регистр p0 из памяти по адресу в p1 с постиндексным регистровым смещением из p2 (то есть прибавляется значение из регистра r2)
c) переход если равно



a) если равно (Z = 1), то в регистр r0 помещается результат сложения содержимого регистра r1 и содержимого регистра r2, сдвинутого вправо на два разряда
b) загрузка в регистр p0 из памяти по адресу в p1 с постиндексным регистровым смещением из p2 (то есть прибавляется значение из регистра r2)
c) переход, если больше (Z = 0, N = V)