



РУКОВОДСТВО ПО ARM-АССЕМБЛЕРУ

1. ПРОГРАММНАЯ МОДЕЛЬ МИКРОПРОЦЕССОРА СЕМЕЙСТВА ARM ...	3
1.1 Состояния ядра ARM	3
1.2 РЕЖИМЫ РАБОТЫ ядра ARM.....	3
1.3 Типы данных.....	3
1.4 РЕГИСТРОВАЯ МОДЕЛЬ ПРОЦЕССОРА	4
1.4.1 Для состояние ARM.....	4
1.4.2 Для состояния THUMB.....	5
1.4.3 ВЗАИМОСВЯЗЬ МЕЖДУ РЕГИСТРАМИ В ARM И THUMB РЕЖИМАХ	5
1.5 РЕГИСТР СТАТУСА	7
1.6 СПОСОБЫ АДРЕСАЦИИ.....	8
1.7 ОБРАБОТКА ПРЕРЫВАНИЙ.....	8
1.8 Сдвиги.....	10
1.8.1 ЛОГИЧЕСКИЙ СДВИГ ВЛЕВО (LSL).....	10
1.8.2 ЛОГИЧЕСКИЙ СДВИГ ВПРАВО (LSR).....	10
1.8.3 АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО (ASR)	11
1.8.4 ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО (ROR)	11
1.8.5 РАСШИРЕННЫЙ ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО (RRX).....	11
2. СИСТЕМА КОМАНД В СОСТОЯНИИ ARM	12
2.1 ОСОБЕННОСТИ СИСТЕМЫ КОМАНД В СОСТОЯНИИ ARM	12
2.2 ФОРМАТЫ КОМАНД.....	13
2.3 КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ	14
2.4 КОМАНДЫ АРИФМЕТИЧЕСКИХ И ЛОГИЧЕСКИХ ОПЕРАЦИЙ.....	16
2.5 КОМАНДЫ ПЕРЕСЫЛКИ	17
2.6 КОМАНДЫ ПОДДЕРЖКИ СОПРОЦЕССОРА	18
3. ТАБЛИЦА КОМАНД	19

1.1 Состояния ядра ARM

Ядро ARM может функционировать в двух состояниях: ARM и THUMB. В состоянии ARM процессор выполняет 32-разрядные команды, в состоянии THUMB — 16-разрядные команды. Встроенный декодер переводит команды, записанные по системе THUMB, в команды ARM. Ввиду компактности используемых форматов система команд THUMB имеет ряд особенностей и ограничений.

Переход процессора из состояния ARM в THUMB и обратно реализуется с помощью команды **BX**.

1.2 Режимы работы ядра ARM

В состоянии ARM процессор может функционировать в одном из следующих режимов:

1. **User** - выполнение программ пользователя.
2. **Supervisor** - работа под управлением операционной системы (ОС), которая оперирует данными, недоступными программам пользователя.
3. **System** - режим выполнения системных программ, при котором ОС работает с данными пользователя.
4. **IRQ (Interrupt Request)** - режим обработки прерываний, в который попадает процессор при поступлении запроса прерывания низшего уровня на вход IRQ.
5. **FIQ (Fast Interrupt Request)** - режим быстрой реакции на прерывания, в который попадает процессор при поступлении запроса высшего уровня на вход FIQ. IRQ может быть прервано FIQ, но IRQ не может прервать FIQ. FIQ имеет больше теневых регистров и не может вызывать SWI (Software Interrupt – Программное прерывание).
6. **Abort** - режим, который реализуется при ошибке обращения к памяти (ошибки такого рода — обращение по несуществующему адресу, попытка записи в ПЗУ и другие, фиксируются контроллером прерываний, который выдаёт процессорному ядру запрос Abort).
7. **Undefined** - режим реализуется при выборке неправильного кода команды.

1.3 Типы данных

Микропроцессор работает с данными следующих разрядностей:

- 8 бит (четверть слова);
- 16 бит (полслова);
- 32 бита (слово).

Слово должно быть ограничено 4 байтами, полслова – 2 байтами.

ВНИМАНИЕ

Далее словом для ARM режима будет считаться последовательность из 32 бит, и из 16 бит для режима THUMB соответственно.

1.4 Регистровая модель процессора

1.4.1 Для состояние ARM

ARM режим содержит 16 регистров прямого доступа: R0 – R15. Все эти регистры являются регистрами общего назначения (**general-purpose**) и могут быть использованы для хранения, как адресов, так и данных. В дополнение к ним имеется семнадцатый регистр (CPSR), использующийся для хранения слова состояния процессора (**status information**).

14 регистр (R13)	Чаще всего является указателем на стек (Stack Pointer - SP).
15 регистр (R14)	Используется для хранения адреса возврата при вызове подпрограмм (Link Register - LR). Содержит копию 15-го регистра (R14) при выполнении операций ветвления и перехода. Все остальное время он может быть использован как регистр общего назначения.
16 регистр (R15)	Используется для хранения адреса текущей команды (Program Counter - PC). В ARM режиме биты [1:0] равны 0, а биты [31:2] содержат адрес команды (PC). В THUMB режиме бит [0] равен 0, а биты [31:1] содержат адрес команды (PC).
17 регистр (CPSR)	Регистр статуса (Current Program Status Register - CPSR). Используется для хранения флагов состояния после выполнения команд.

Для каждого режима функционирования имеется соответствующая регистровая модель, которая содержит набор 32-разрядных теневых регистров (обозначены серым цветом), доступных в этом режиме (Таблица 1.1). Например, при переходе в режим **FIQ** (поступление запроса прерывания на вход FIQ) сохраняется текущее содержимое регистров R8 - R14, вместо которых в этом режиме используются регистры R8_fiq - R14_fiq. Поэтому при обработке данного прерывания нет необходимости сохранять содержимое этих регистров в стеке. В ряде случаев это обеспечивает более быстрый переход к обработчику прерываний и возврат из него.

Таблица 1.1 – Регистровая модель процессора в состоянии ARM

User, System	FIQ	Supervisor	Abort	IRQ	Undefined
R0 (a1)	R0	R0	R0	R0	R0
R1 (a2)	R1	R1	R1	R1	R1
R2 (a3)	R2	R2	R2	R2	R2
R3 (a4)	R3	R3	R3	R3	R3
R4 (v1)	R4	R4	R4	R4	R4
R5 (v2)	R5	R5	R5	R5	R5
R6 (v3)	R6	R6	R6	R6	R6
R7 (v4)	R7	R7	R7	R7	R7
R8 (v5)	R8_fiq	R8	R8	R8	R8
R9 (v6)	R9_fiq	R9	R9	R9	R9
R10 (v7)	R10_fiq	R10	R10	R10	R10
R11 (v8)	R11_fiq	R11	R11	R11	R11
R12 (IP)	R12_fiq	R12	R12	R12	R12
R13 (SP)	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14 (LR)	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15	R15	R15	R15	R15
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

При переходе в другой режим текущее содержимое регистра CPSR (слово состояния) переписывается в регистр SPSR, соответствующий наступившему режиму. При возвращении к исходному режиму содержимое CPSR восстанавливается.

Все регистры имеют дополнительные имена (указаны в скобках), определённые стандартом фирмы ARM. Этот документ регламентирует использование регистров при вызове подпрограмм и организации передачи данных между ними и предназначен, главным образом, для разработчиков компиляторов:

- Регистры a1 - a4 (argument 1 - 4) используются для передачи параметров подпрограмм, результат возвращается в регистр R0 (a1);
- Регистры v1 - v8 (variable 1 - 8) используются для хранения локальных переменных;
- Регистр IP (Intra-Procedure-call scratch register) - служит для хранения промежуточных данных между вызовами процедур.

1.4.2 Для состояния THUMB

Набор регистров в состоянии THUMB сокращён (Таблица 1.2) – он является подмножеством регистрового банка в состоянии ARM. В формате большинства команд THUMB под номер регистра отведено только 3 бита, поэтому прямое обращение возможно только к регистрам R0-R7. Регистры R8-R12 доступны только через специальные команды загрузки. Регистры SP и LR выполняют фиксированные функции как указатель стека и регистр связи. В состоянии THUMB имеются специальные команды, ориентированные на работу с этими регистрами – загрузка в стек проходит только через SP, адрес возврата из подпрограммы сохраняется только в LR.

Таблица 1.2 - Регистровая модель процессора в состоянии THUMB

User, System	FIQ	Supervisor	Abort	IRQ	Undefined
R0 (a1)	R0	R0	R0	R0	R0
R1 (a2)	R1	R1	R1	R1	R1
R2 (a3)	R2	R2	R2	R2	R2
R3 (a4)	R3	R3	R3	R3	R3
R4 (v1)	R4	R4	R4	R4	R4
R5 (v2)	R5	R5	R5	R5	R5
R6 (v3)	R6	R6	R6	R6	R6
R7 (v4)	R7	R7	R7	R7	R7
SP	SP_fiq	SP_svc	SP_abt	SP_irq	SP_und
LR	LR_fiq	LR_svc	LR_abt	LR_irq	LR_und
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

1.4.3 Взаимосвязь между регистрами в ARM и THUMB режимах

Регистры в THUMB режиме связаны с регистрами в ARM режиме следующим образом:

- Регистры R0 – R7 в THUMB и ARM режиме идентичны;
- Регистр CPSR в THUMB и ARM режиме идентичен;
- Регистр SP (Stack Pointer) в режиме THUMB соответствует регистру R13 в режиме ARM;
- Регистр LR (Link Register) в режиме THUMB соответствует регистру R14 в режиме ARM;
- Регистр PC (Program Counter) в режиме THUMB соответствует регистру R15 в режиме ARM;

В THUMB режиме старшие регистры (Hi) имеют ограниченный доступ и могут быть использованы для быстрого временного хранения. Значение может быть отправлено из регистров R0 - R7 (младших - Lo) в старшие регистры (Hi) с помощью специальных вариантов команды MOV. Старшие регистры (Hi) также могут сравниваться или добавляться с младшими (Lo) регистрами с помощью команд CMP и ADD. Соответствие регистров в THUMB и ARM режимах изображено на рисунке 1.1.

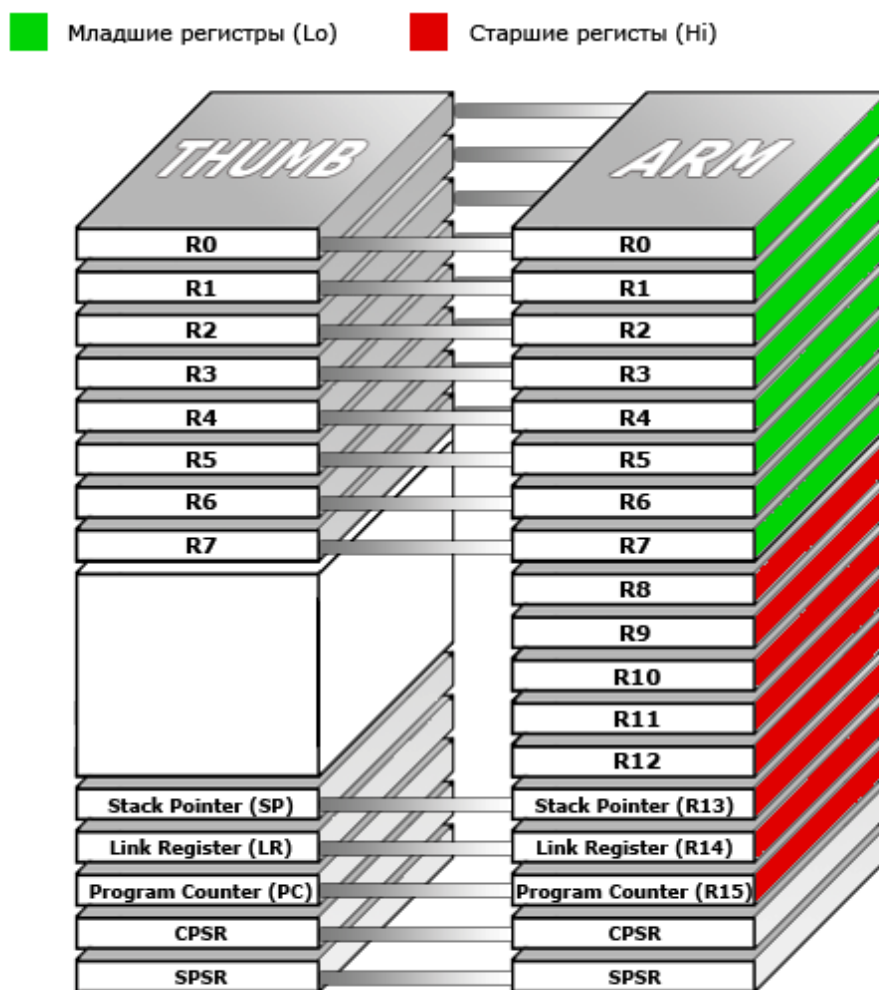


Рисунок 1.1 – Соответствие регистров в THUMB и ARM режимах

1.5 Регистр статуса

Формат регистра статуса представлен на рисунке 1.2.



Рисунок 1.2 – Формат регистра статуса

Биты N, Z, C, V являются флагами условия. Они могут быть изменены в результате выполнения арифметических или логических операций и использованы для определения способа выполнения команды.

Младшие 8 бит регистра статуса (I, F, T, M[4:0]) содержат биты управления. Они изменяются при возникновении исключительных ситуаций (например прерываний).

- Бит T отражает режим работы процессора. Когда он установлен в единицу, процессор работает в режиме THUMB, в противном случае процессор работает в режиме ARM. Этот бит реагирует на внешний сигнал **TBIT**.
- Биты запрета прерываний I, F запрещают IRQ и FIQ прерывания соответственно.
- Биты режима доступа M[4:0] определяют режим работы процессора в соответствии с Таблицей 1.3.
- Биты [27-8] являются зарезервированными.

Таблица 1.3 – Кодирование режимов работы процессора

M[4:0]	Режим работы	Видимые регистры в THUMB режиме	Видимые регистры в ARM режиме
10000	User	R7..R0 LR, SP PC, CPSR	R14..R0 PC, CPSR
10001	FIQ	R7..R0 LR_fiq, SP_fiq PC, CPSR, SPSR_fiq	R7..R0 R14_fiq..R8_fiq PC, CPSR, SPSR_fiq
10010	IRQ	R7..R0 LR_irq, SP_irq PC, CPSR, SPSR_irq	R12..R0 R14_irq..R13_irq PC, CPSR, SPSR_irq
10011	Supervisor	R7..R0 LR_svr, SP_svr PC, CPSR, SPSR_svr	R12..R0 R14_svr..R13_svr PC, CPSR, SPSR_svr
10111	Abort	R7..R0 LR_abt, SP_abt PC, CPSR, SPSR_abt	R12..R0 R14_abt, R13_abt PC, CPSR, SPSR_abt
11011	Undefined	R7..R0 LR_und, SP_und PC, CPSR, SPSR_und	R12..R0 R14_und, R13_und PC, CPSR, SPSR_und
11111	System	R7..R0 LR, SP PC, CPSR	R14..R0 PC, CPSR

1.6 Способы адресации

Процессор ARM использует следующие способы адресации (Таблица 1.4)

Таблица 1.4 – Способы адресации	
Обозначение	Название
#Imm	Непосредственная
Rn	Регистровая
Rn, shift #n	Регистровая с масштабированием
[Rn]	Косвенно-регистровая
[Rn, ±Imm] {!}	Пре-индексная с непосредственным смещением
[Rn, ±Rm] {!}	Пре-индексная с регистровым смещением
[Rn, ±Rm, shift #n]	Пре-индексная с масштабированным регистровым смещением
[Rn], ±Rm	Пост-индексная с регистровым смещением
[Rn], ±Rm, shift #n	Пост-индексная с масштабированным регистровым смещением

- 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.
- При регистровой адресации с масштабированием в команде задается имя регистра Rn, содержимое которого является операндом или результатом операции с учетом того, что при масштабировании содержимое регистра Rn сдвигается на число разрядов n, указанное в команде (от 1 до 31). Вместо "shift" в ассемблерном тексте используется один из четырех символов, задающих вид реализуемого сдвига (см. п. 1.8):
 1. LSL – логический сдвиг влево;
 2. LSR – логический сдвиг вправо;
 3. ASR – арифметический сдвиг вправо;
 4. ROR – циклический сдвиг вправо.
- При косвенно-регистровой адресации содержимое указанного регистра Rn содержит адрес ячейки памяти, где хранится операнд или результат.
- В случае пре-индексной 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 пост-индексной адресации адресом служит содержимое базового регистра Rn, которое индексируется после выполнения операции. Если в поле операнда содержится символ {!}, то индексированное содержимое Rn сохраняется после выполнения операции.

1.7 Обработка прерываний

При поступлении внешнего запроса прерывания или обнаружении ошибки процессор автоматически переходит в состояние ARM и начинает работу в соответствующем режиме **IRQ**, **FIQ**, **Abort** или **Undefined**. Последовательность его действий такова:

1. Адрес возврата сохраняется в **LR**.
2. Содержимое **CPSR** копируется в соответствующий регистр **SPSR**.
3. В **CPSR** значения битов **M[4-0]** устанавливаются в соответствии с новым режимом работы (**IRQ**, **FIQ**, **Abort** или **Undefined**).
4. Из размещенной в памяти таблицы векторов прерываний в PC загружается адрес первой команды обработчика прерывания.

В зависимости от алгоритма работы контроллера прерываний, могут быть также замаскированы некоторые исключения, возникновение которых может нарушить процесс обработки.

В системе команд ARM отсутствует специальная команда для выхода из обработчика прерывания. Для выполнения возврата программа должна выполнить следующие действия:

1. Восстановить содержимое **CPSR** из **SPSR**.
2. Разрешить прерывания, запрещённые при входе в обработчик.
3. Загрузить в **PC** адрес возврата из регистра **LR** с помощью команды выхода из прерывания (Таблица 1.5).

В случаях выхода из режимов **FIQ**, **IRQ**, **Abort** может потребоваться коррекция адреса возврата. После обработки этих исключений процессор должен вернуться к выполнению команды, вызвавшей прерывание. В регистре **LR** хранится адрес следующей команды, поэтому при выходе из обработчика прерываний содержимое этого регистра следует скорректировать с помощью команды вычитания **SUBS**.

Ядро **ARM** реализует 7 видов прерываний, которые перечислены ниже в порядке снижения их приоритета.

1. **Reset** – реализуется при подаче сигнала запуска на внешний вывод процессора. Процессор переходит в режим **Supervisor** и начинает выполнение программы с адреса, размещённого в первой ячейке памяти (адрес ячейки 0x00000000).
2. **Data abort** – ошибка при обращении к данным (фиксируется контроллером прерываний). Процессор переходит в режим **Abort**.
3. **FIQ** – реализуется при подаче сигнала прерывания на один из выводов **nFIQ**. Процессор переходит в режим **FIQ**. При поступлении данного прерывания в регистровом банке происходит переключение 7 регистров R8 - R14 (Рисунок 1.1), что позволяет сократить или исключить операции сохранения содержимого регистров в стеке. За счёт этого переход к обработке прерывания происходит быстрее.
4. **IRQ** – возникает при подаче сигнала прерывания на один из выводов **nIRQ**. Процессор переходит в режим **IRQ**.
5. **Prefetch abort** – ошибка при выборке команды (фиксируется контроллером прерываний). Процессор переходит в режим **Abort**.
6. **Undefined instruction** – выборка неправильного кода команды. Процессор переходит в режим **Undefined**.
7. **Software interrupt** – программное прерывание по команде SWI. Процессор переходит в режим **Supervisor**. Программные прерывания используются, как правило, для вызова функций ОС.

Вектора прерываний располагаются в памяти последовательно, начиная с нулевого адреса (Таблица 1.5). Каждый вектор содержит 4 байта, которые являются адресами первой команды обработчика прерываний.

Таблица 1.5 - Размещение векторов прерываний и команды выхода

Адрес	Вектор	Команда выхода из прерывания
0x00000000	Запуск (Reset)	-
0x00000004	Неправильная команда	MOVS PC, R14_und
0x00000008	Программное прерывание	MOVS PC, R14_svc
0x0000000C	Ошибка выборки инструкции	SUBS PC, R14_abt, #4
0x00000010	Ошибка выборки данных	SUBS PC, R14_abt, #8
0x00000014	Зарезервировано	-
0x00000018	IRQ	SUBS PC, R14_irq, #4
0x0000001C	FIQ	SUBS PC, R14_fiq, #4

В состав микроконтроллера с ядром ARM вводится контроллер прерываний, который транслирует запрос от любого периферийного устройства в запрос IRQ или FIQ. Далее программа-обработчик должна самостоятельно определить источник запроса, используя регистры контроллера прерываний.

1.8 Сдвиги

1.8.1 Логический сдвиг влево (LSL)

Во время этого сдвига содержимое Rm сдвигается влево на заданное число разрядов (**#n**) со входом нуля и выходом **32 - #n** бита в регистр статуса (CPSR) во флаг C. На рисунке 1.3 изображен пример операции **LSL #5**.



Рисунок 1.3 – Логический сдвиг влево

1.8.2 Логический сдвиг вправо (LSR)

Во время этого сдвига содержимое Rm сдвигается вправо на заданное число разрядов (**#n**) со входом нуля и выходом **#n - 1** бита в регистр статуса (CPSR) во флаг C. На рисунке 1.4 изображен пример операции **LSR #5**.



Рисунок 1.4 – Логический сдвиг вправо

1.8.3 Арифметический сдвиг вправо (ASR)

Во время этого сдвига содержимое Rm сдвигается вправо на заданное число разрядов (**#n**) со входом бита 31-го содержимого Rm и выходом **#n - 1** бита в регистр статуса (CPSR) во флаг C. На рисунке 1.5 изображен пример операции **ASR #5**.



Рисунок 1.5 – Арифметический сдвиг вправо

1.8.4 Циклический сдвиг вправо (ROR)

Во время этого сдвига содержимое Rm сдвигается вправо на заданное число разрядов (**#n**) со входом выдвигаемых из содержимого Rm бит в начало слова и выходом **#n - 1** бита в регистр статуса (CPSR) во флаг C. На рисунке 1.6 изображен пример операции **ROR #5**.

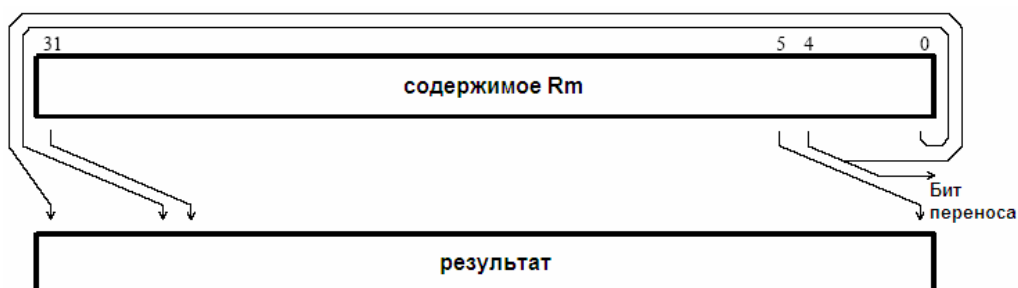


Рисунок 1.6 – Циклический сдвиг вправо

1.8.5 Расширенный циклический сдвиг вправо (RRX)

Расширенный циклический сдвиг вправо (Rotate Right Extended - RRX) является частным случаем сдвига **ROR – ROR #0**. Во время этого сдвига содержимое Rm сдвигается вправо со входом флага C из регистра статуса (CPSR) и выходом нулевого бита в регистр статуса (CPSR) во флаг C. На рисунке 1.7 изображен пример операции **ROR #0**.



Рисунок 1.7 – Расширенный циклический сдвиг вправо

2.1 Особенности системы команд в состоянии ARM

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 её снабдить соответствующим суффиксом. Виды условий и их суффиксы приведены в Таблице 2.1.

Таблица 2.1 – Виды условий и их суффиксы

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 либо ноль
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	$N \neq V$	Меньше
GT	$Z = 0, N = V$	Больше
LE	$Z = 1, N \neq V$	Меньше или равно
AL		Всегда

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.

2.2 Форматы команд

Общая таблица форматов команд изображена на Рисунке 2.1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Условие	0	0	1					Код оп.	S		Rn		Rd	Операнд 2													Общий формат команд обработки данных					
Условие	0	0	0	0	0	0	0	A	S		Rd		Rn		Rs	1	0	0	1													Формат команды умножения
Условие	0	0	0	0	1	U	A	S		RdHi		RdLo		Rn	1	0	0	1														Умножение длинных чисел
Условие	0	0	0	1	0	B	0	0		Rn		Rd	0	0	0	0	1	0	0	1												Команда обмена содержимого регистров
Условие	0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1										Команда перехода и смены состояния
Условие	0	0	0	P	U	0	W	L		Rn		Rd	0	0	0	0	1	S	H	1												Передача 16-разрядного полуслова – регистровое смещение
Условие	0	0	0	P	U	1	W	L		Rn		Rd	Смещение				1	S	H	1	Смещение											Передача 16-разрядного полуслова – непосредственное смещение
Условие	0	1	1	P	U	B	W	L		Rn		Rd	Смещение													Передача 32-разрядного слова						
Условие	0	1	1																1				Неопределённые коды команд									
Условие	1	0	0	P	U	S	W	L		Rn	Список регистров													Команды групповой пересылки содержимого регистров								
Условие	1	0	1	L	Смещение																Команды перехода											
Условие	1	1	0	P	U	N	W	L		Rn		CRd		CP#	Смещение				Передача константы сопроцессору													
Условие	1	1	1	0	Код оп.					CRn		CRd		CP#		CP	0		CRm	Выполнение команды сопроцессором												
Условие	1	1	1	0	Код оп.				L	CRn		Rd		CP#		CP	1		CRm	Передача содержимого регистра в сопроцессор												
Условие	1	1	1	1	Игнорируется процессором																Программное прерывание											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Рисунок 2.1 – Таблица форматов команд

	Условные обозначения
Условие	Поле условия (табл. 4)
Rn, Rm	Поля номеров регистров операндов
Rd	Поле регистра назначения
Код оп.	Поле кода операции
Смещение	Поле смещения
S	Разрешение установки флагов условий
L	Флаг направление передачи данных (в память / из памяти)
B	Флаг байтовой операции
W	Разрешение индексации операнда
P	Флаг пре- или пост-индексной адресации
U	Направление изменения содержимого индексного регистра
A	Флаг накопления в команде умножения
N	Вид смещения (короткое – длинное)

2.3 Команды передачи управления

Команды передачи управления служат для изменения хода программы. В таблице 2.1 приведен список команд передачи управления.

Таблица 2.1 – Список команд 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	$PC := R_n, T := R_n[0]$
SWI	Программное прерывание	режим Supervisor, $LR := PC, PC := 0x0008$

Команда перехода B с соответствующим условием обеспечивает выполнение условных или безусловных переходов. Переход к подпрограмме осуществляется командой BL (бит 24 установлен в 1), при этом текущее содержимое программного счётчика PC (адрес возврата) сохраняется в регистре связи LR (R14). В качестве операнда в команде задаётся 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 путём организовать сохранение промежуточных значений адресов возврата (содержимого LR) в стеке, используя регистр SP в качестве указателя. Также программным путём обеспечивается, в случае необходимости, сохранение в стеке содержимого регистров. Заданный адрес перехода должен быть чётным, если процессор находится в состоянии THUMB, или кратным четырём, если процессор в состоянии ARM. Формат команд данного типа представлен на рисунке 2.2

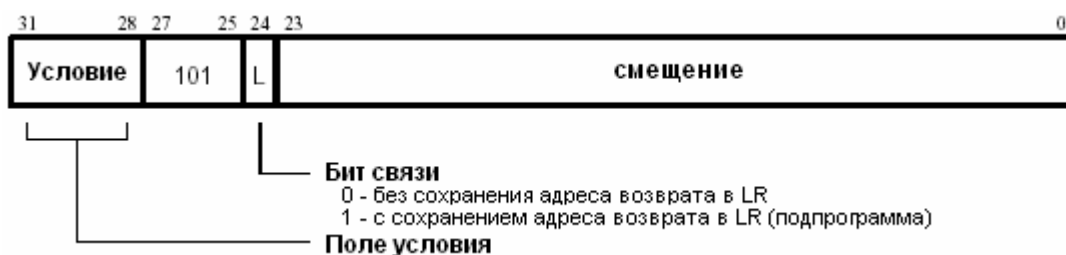


Рисунок 2.2 – Формат команд B, BL

Синтаксис ассемблера

$B\{L\}\{\text{условие}\} <\text{выражение}>$

- {L} – бит связи, используется для сохранения адреса возврата в LR
- {условие} – двухсимвольная мненоника условия (см. таблицу 2.1)
- {выражение} – адрес перехода (смещение)

Команда BX позволяет осуществить переход с одновременным изменением состояния процессора. Адрес перехода определяется содержимым регистра R_n , заданного в команде, а состояние процессора – нулевым битом этого регистра, который копируется в регистр CPSR в качестве бита T. Формат команды BX представлен на рисунке 2.3.



Рисунок 2.3 – Формат команды BX

Синтаксис ассемблера

BX {условие} Rn

{условие} – двухсимвольная мненоника условия (см. таблицу 2.1)

Rn – регистр

Команда программного прерывания SWI используется для доступа к функциям ОС. 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 на обработчик прерывания. Выполнение команды SWI - единственный способ, позволяющий перевести процессор из режима User в режим Supervisor. Формат команды представлен на рисунке 2.4.



Рисунок 2.4 – Формат команды SWI

Синтаксис ассемблера

SWI {условие} <выражение>

{условие} – двухсимвольная мненоника условия (см. таблицу 2.1)

{выражение} – игнорируется процессором

2.4 Команды арифметических и логических операций

В стадии разработки

2.5 Команды пересылки

В стадии разработки

2.6 Команды поддержки сопроцессора

В стадии разработки

МНЕМОНИКА	СИНТАКСИС	ОПКОД	ДЕЙСТВИЕ	ОПИСАНИЕ
КОМАНДЫ ПЕРЕХОДА				
B	B <смещение>	0xEAxxxxxx	R15 = <смещение>	Безусловный переход
BL	BL < смещение>	0xEBxxxxxx	R14 = <смещение следующей инструкции>, R15 = <смещение>	Вызов функций
BEQ	BEQ < смещение>	0x0Axxxxxx	Если флаг Z установлен	Переход, если операнды равны между собой
BNE	BNE < смещение>	0x1Axxxxxx	Если флаг Z сброшен	Переход, если операнды не равны между собой
BGE	BGE < смещение>	0xAAxxxxxx	Если установлены/сброшены флаги N и V	Переход, если результат операции больше или равен 0
BGT	BGT < смещение>	0xCAxxxxxx	Если установлены/сброшены флаги N и V и сброшен флаг Z	Переход, если результат операции больше 0
BVS	BVS < смещение>	0x6Axxxxxx	Если установлен флаг V после арифметической операции	Переход, если произошло переполнение
BVC	BVC < смещение>	0x7Axxxxxx	Если сброшен флаг V после арифметической операции	Переход, если переполнения не было
BHI	BHI < смещение>	0x8Axxxxxx	Если установлен флаг C и сброшен флаг Z (беззнаковое сравнение)	Переход, если один операнд больше другого
BLS	BLS < смещение>	0x9Axxxxxx	Если сброшен флаг C или установлен флаг Z (беззнаковое сравнение)	Переход, если один операнд меньше или равен другому
BLEQ	BLEQ < смещение>	0x0Bxxxxxx	Если флаг Z установлен	Вызов функций, если операнды равны между собой
BLNE	BLNE < смещение>	0x1Bxxxxxx	Если флаг Z сброшен	Вызов функций, если операнды не равны между собой
BLGE	BLGE < смещение>	0xABxxxxxx	Если установлены/сброшены флаги N и V	Вызов функций, если результат операции больше или равен 0
BLGT	BLGT < смещение>	0xCBxxxxxx	Если установлены/сброшены флаги N и V и сброшен флаг Z	Вызов функций, если результат операции больше 0
BLVS	BLVS < смещение>	0x6Bxxxxxx	Если установлен флаг V после арифметической операции	Вызов функций, если произошло переполнение
BLVC	BLVC < смещение>	0x7Bxxxxxx	Если сброшен флаг V после арифметической операции	Вызов функций, если переполнения не было
BLHI	BLHI < смещение>	0x8Bxxxxxx	Если установлен флаг C и сброшен флаг Z (беззнаковое сравнение)	Вызов функций, если один операнд больше другого
BLLS	BLLS < смещение>	0x9Bxxxxxx	Если сброшен флаг C или установлен флаг Z (беззнаковое сравнение)	Вызов функций, если один операнд меньше или равен другому