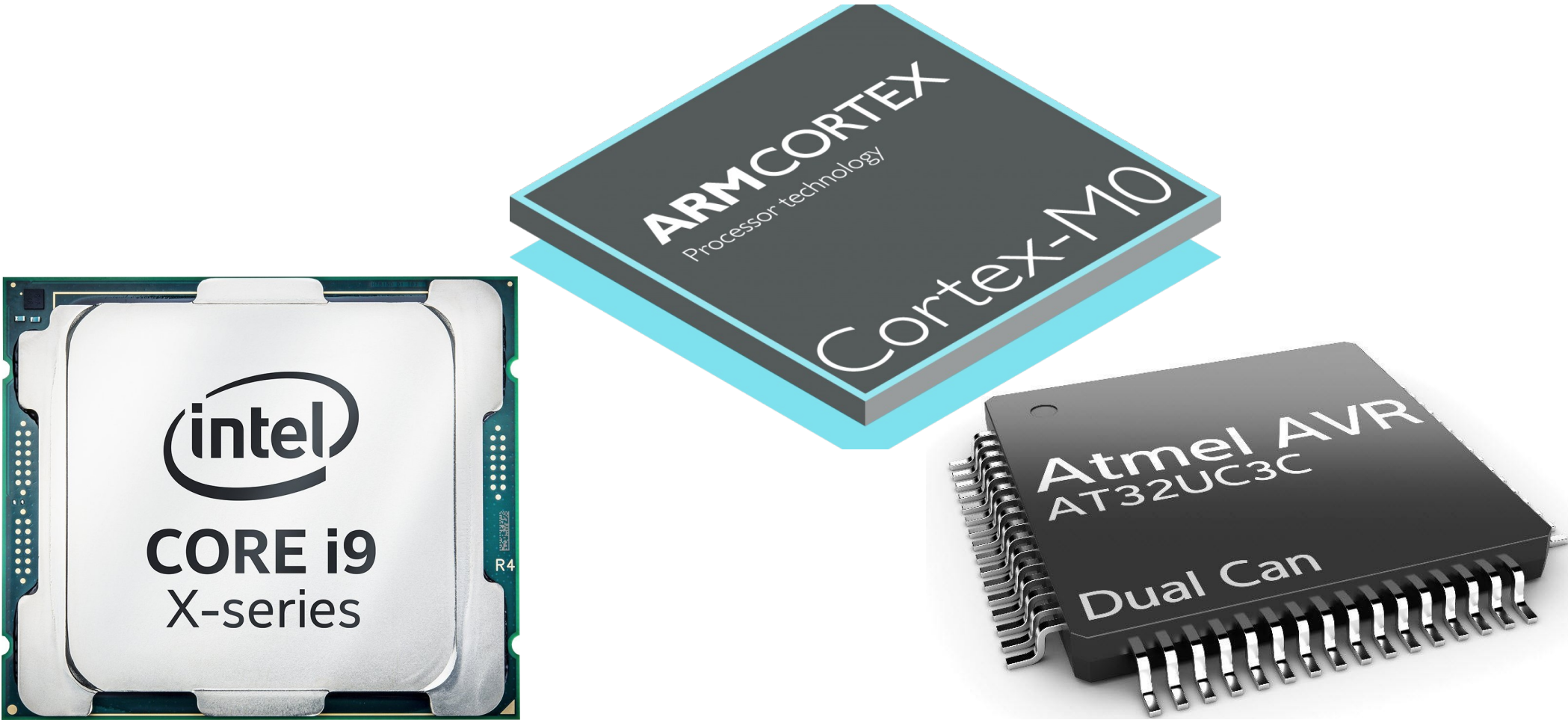


Архитектура процессоров ARM



История появления

ARM (Advanced RISC Machine, усовершенствованная RISC-машина; иногда Acorn RISC Machine) — система команд и семейство описаний и готовых топологий 32-битных и 64-битных микропроцессорных/микроконтроллерных ядер разрабатываемых компанией ARM Limited.

Готовые топологии ядер ARM лицензируют компании:

AMD, Apple, Analog Devices, Atmel, Xilinx, Cirrus Logic, Intel (до 27 июня 2006 года), Marvell, NXP, STMicroelectronics, Samsung, LG, MediaTek, Qualcomm, Sony, Texas Instruments, Nvidia, Freescale, Миландр, ЭЛВИС, HiSilicon, Байкал электроникс.

Значимые семейства процессоров: ARM7, ARM9, ARM11 и Cortex.

Многие лицензиаты проектируют собственные топологии ядер на базе системы команд ARM:

DEC StrongARM, Freescale i.MX, Intel XScale, NVIDIA Tegra, ST-Ericsson Nomadik, Krait и Kryo в Qualcomm Snapdragon, Texas Instruments OMAP, Samsung Hummingbird, LG H13, Apple A6 и HiSilicon K3.

Официальный проект Acorn RISC Machine был начат в октябре 1983 года.

Представлена в 1985

Архитектура ARM

Архитектура развивалась с течением времени и, начиная с ARMv7, были определены 3 профиля:

- *A (application) — для устройств, требующих высокой производительности (смартфоны, планшеты);*
- *R (real time) — для приложений, работающих в реальном времени;*
- *M (microcontroller) — для микроконтроллеров и недорогих встраиваемых устройств.*

Профили могут поддерживать меньшее количество команд (команды определенного типа).

Справочное руководство по архитектуре ARM, разграничивает все типы интерфейсов, которые поддерживает ARM, так как детали реализации каждого типа процессора могут различаться.

Чтобы сохранить устройство простым и быстрым, оригинальное изготовление ARM было исполнено без микрокода.

Архитектура ARM. Режимы

Процессор может находиться в одном из следующих операционных режимов:

- **User mode** — обычный режим выполнения программ. В этом режиме выполняется большинство программ.
- **Fast Interrupt (FIQ)** — режим быстрого прерывания (меньшее время срабатывания).
- **Interrupt (IRQ)** — основной режим прерывания.
- **System mode** — защищённый режим для использования операционной системой.
- **Abort mode** — режим, в который процессор переходит при возникновении ошибки доступа к памяти (доступ к данным или к инструкции на этапе *prefetch* конвейера).
- **Supervisor mode** — привилегированный пользовательский режим.
- **Undefined mode** — режим, в который процессор входит при попытке выполнить неизвестную ему инструкцию.

Переключение режима процессора происходит при возникновении соответствующего исключения или же модификацией регистра статуса.

Архитектура ARM. Регистры

31 РОН разрядностью 32 бит.

В зависимости от режима и состояния процессора пользователь имеет доступ только к строго определённым набору регистров.

В ARM state разработчику постоянно доступны 17 регистров:

- 13 **РОН** (r0..r12).
- **Stack Pointer** (r13, SP) — содержит указатель стека выполняемой программы.
- **Link register** (r14, LR) — содержит адрес возврата в инструкциях ветвления.
- **Program Counter** (r15, PC) — биты [31:1] содержат адрес выполняемой инструкции.
- **Program Status Register** (PSR) — содержит флаги, описывающие текущее состояние процессора. Модифицируется при выполнении многих инструкций: логических, арифметических, и др.

Имя регистра	Второе обозначение	Использование	
R0	-	Младший банк регистров общего назначения ЦПУ	Нет никаких ограничений в использовании этих регистров в качестве операндов-источников или операндов-приемников
R1	-		
R2	-		
R3	-		
R4	-		
R5	-		
R6	-		
R7	-		
R8	-	Старший банк регистров общего назначения ЦПУ	Для некоторых команд имеются ограничения в использовании (см. <u>приложение 1</u>).
R9	-		
R10	-		
R11	-		
R12	-		
R13	SP		
		PSP	SP_process – указатель стека процесса пользователя (программ нижнего уровня)
		MSP	SP_main – указатель стека обработчиков исключений и программ верхнего уровня
R14	LR	Link register – линк-регистр, регистр связи Содержит адрес возврата в основную программу после команды вызова подпрограммы	
R15	PC	Program counter – программный счетчик (счетчик команд) Всегда содержит адрес очередной, подлежащей выполнению команды.	
	PSR	Program Status Register – Регистр статуса программы Состоит из трех регистров:	
		APSR	Application Program Status Register – регистр статуса программы-приложения
		IPSR	Interrupt Program Status Register – регистр статуса системы прерываний/исключений
		EPSR	Execution Program Status Register – регистр статуса системы выполнения команд

Архитектура ARM. Счетчик команд

Регистр r15, pc (Program Counter) – указатель на исполняемые команды.

Над его содержимым можно выполнять разные арифметические и логические операции, тем самым исполнение программы будет переходить на другие адреса.

Кроме собственно указателя на исполняемые команды содержится информация:

- Биты 31:28 – флаги результата выполнения арифметической или логической операции { Negative, Zero, Carry, oVerflow }
- Биты 27 – маска IRQ прерывания, прерывания запрещены, когда бит установлен.
- Биты 26 – маска FIRQ прерывания, быстрые прерывания запрещены, когда бит установлен.
- Биты 25:2 – собственно указатель на команды программы занимает только 26 бит.
- Биты 1:0 – текущий режим работы процессора.

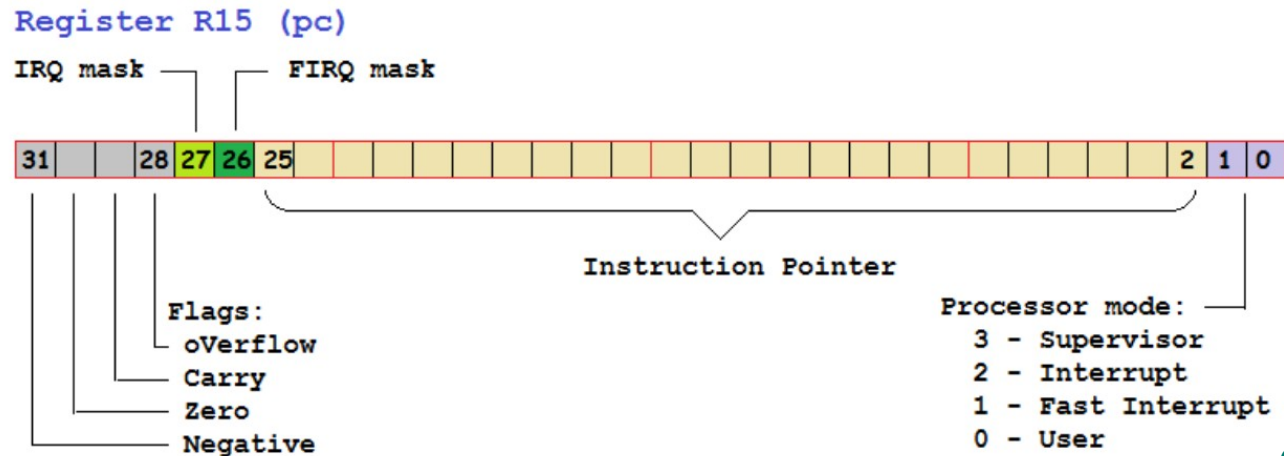
3 - Supervisor

2 - Interrupt

1 - Fast Interrupt

0 – User

В старших процессорах ARM все флаги и служебные биты расположены в отдельных регистрах Current Program Status Register (cpsr) и Saved Program Status Register (spsr), для доступа к которым есть отдельные специальные команды. Это сделано для того, что бы расширить доступное адресное пространство для программ.



Архитектура ARM. Набор команд

Режим 32-разрядного набора команд

ARM Base Instruction Set:

ADC, ADD, AND, B/BL, BIC, CMN, CMP, EOR, LDM, LDR/LDRB, MLA, MOV, MUL, MVN, ORR, RSB, RSC, SBC, STM, STR/STRB, SUB, SWI, SWP, TEQ, TST

Набор команд Thumb

Для улучшения плотности кода. Альтернативный набор 16-битных команд. Уменьшение длины команды за счёт сокрытия некоторых операндов и ограничения возможностей адресации.

Меньшие коды операций обладают меньшей функциональностью. Ветвления могут быть только условными. Многие коды операций имеют ограничение в виде доступа только к половине главных регистров процессора.

Более короткие коды операций в целом дают большую плотность кода, но требуют дополнительных команд.

Набор команд Jazelle

Позволяет байткоду Java исполняться прямо в архитектуре ARM в качестве 3-го состояния исполнения (и набора команд) наряду с обычными командами ARM и режимом Thumb. Поддерживается, начиная с архитектуры ARMv6.

Архитектура ARM. Набор команд

Мнемоника	Команда	Действие
ADC	Сложение с переносом	$Rd := Rn + Op2 + \text{перенос}$
ADD	Сложение	$Rd := Rn + Op2$
AND	Логическое И	$Rd := Rn \text{ AND } Op2$
B	Переход	$R15 := \text{адрес}$
BIC	Очистить бит	$Rd := Rn \text{ AND NOT } Op2$
BL	Переход со ссылкой	$R14 := R15, R15 := \text{адрес}$
BX	Переход и переключение режима ядра	$R15 := Rn, \text{Т бит} := Rn[0]$
CDP	Обработать данные сопроцессором	(зависит от типа сопроцессора)
CMN	Сравнить с отрицательным операндом	$CPSR \text{ флаги} := Rn + Op2$
CMP	Сравнение	$CPSR \text{ флаги} := Rn - Op2$
EOR	Исключающее ИЛИ	$Rd := (Rn \text{ AND NOT } Op2) \text{ OR } (Op2 \text{ AND NOT } Rn)$
LDC	Загрузить в сопроцессор из памяти	Загрузить в сопроцессор
LDM	Загрузить сразу несколько регистров	Манипуляции со стеком (Pop)
LDR	Загрузить регистр из памяти по указанному адресу	$Rd := (\text{адрес})$
MCR	Скопировать регистр CPU в регистр сопроцессора	$cRn := rRn \{<op>cRm\}$
MLA	Умножение со сложением	$Rd := (Rm * Rs) + Rn$
MOV	Загрузить в регистр константу	$Rd := Op2$
MRC	Скопировать регистр сопроцессора в регистр CPU	$Rn := cRn \{<op>cRm\}$
MRS	Переместить регистр статуса/флагов PSR в регистр Rn	$Rn := PSR$
MSR	Загрузить в PSR статус/флаги указанный регистр	$PSR := Rm$
MUL	Умножение	$Rd := Rm * Rs$
MVN	Загрузить регистр отрицательной константой	$Rd := 0xFFFFFFFF \text{ EOR } Op2$
ORR	Логическое ИЛИ	$Rd := Rn \text{ OR } Op2$
RSB	Обратное вычитание	$Rd := Op2 - Rn$
RSC	Обратное вычитание с переносом	$Rd := Op2 - Rn - 1 + \text{Перенос}$
SBC	Вычитание с переносом	$Rd := Rn - Op2 - 1 + \text{Перенос}$
STC	Сохранить регистр сопроцессора в памяти	$\text{адрес} := CRn$
STM	Сохранить сразу несколько регистров	Манипуляции со стеком (Push)
STR	Сохранить регистр в памяти	$\langle \text{адрес} \rangle := Rd$
SUB	Вычитание	$Rd := Rn - Op2$
SWI	Программное прерывание	Вызывается операционной системой
SWP	Обменять местами содержимое регистра и памяти	$Rd := [Rn], [Rn] := Rm$
TEQ	Побитовая проверка на равенство	$CPSR \text{ флаги} := Rn \text{ EOR } Op2$
TST	Проверка битов	$CPSR \text{ флаги} := Rn \text{ AND } Op2$

Архитектура ARM. Набор команд

Набор команд ARM 64 бит (ARMv8)

Назван: набор команд **A64**.

Поддержка 32-битных команд названа **A32** и выполняется на архитектурах AArch32.

Инструкции **Thumb** поддерживаются в режиме **T32**, только для 32-битных архитектур.

Допускается исполнение 32-битных приложений в 64-битной ОС, и запуск виртуализованной 32-битной ОС при помощи 64-битного гипервизора.

Особенности AArch64:

- Новый набор команд A64
- 31 регистр общего назначения, каждый длиной 64 бит
- Отдельные регистры SP и PC
- Инструкции имеют размер 32 бит и многие совпадают с командами A32
- Большинство инструкций работают как с 32, так и с 64-разрядными аргументами
- Адреса имеют размер 64 бит
- Улучшения Advanced SIMD (NEON) enhanced
- Количество 128-битных регистров увеличено с 16 до 32
- Вычисления с плавающей запятой двойной точности
- Полная совместимость с IEEE 754
- Новая система обработки исключений

Архитектура ARM. Набор команд

Архитек- тура	Процессоры	Разряд- ность	Поддерживаются наборы команд			
			ARM-32 bit ISA	Thumb -16 bit ISA	Thumb2	ARM
v4	ARM7TDMI SC100	32	V	V		
v5	ARM946 ARM968 ARM926	32	V	V		
v6	ARM1156T2 ARM1136 ARM1176 ARM11MP	32	V	V	V	
v7	Cortex-R4 Cortex-R5 Cortex-R7 Cortex-A5 Cortex-A8 Cortex-A9 Cortex-A12 Cortex-A15	32	V		V	
v6-M	Cortex-M0 Cortex-M1 SC000	32		V		
v7-M	Cortex-M3 Cortex-M4 SC300	32		V	V	
v8	Cortex-A53 Cortex-A57	64		V	V	V

Архитектура ARM. Формат команд

31 – 28	27 – 25	24 - 21	20	19 – 16	15 – 12	11 – 0
Condition	Operand type	OpCode	Set Condition Codes	Operand Register	Destination Register	Immediate Operand

Архитектура ARM. Формат команд

		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	
Обработка данных и передача FSR	Условие	0	0	1	Код оп.				S	Rn				Rd				Операнд 2																
	Умножение	Условие	0	0	0	0	0	0	A	S	Rd				Rn				Rs				1	0	0	1	Rm							
Длинное умножение	Условие	0	0	0	0	1	U	A	S	RdHi				RdLo				Rn				1	0	0	1	Rm								
Одиночный обмен данными	Условие	0	0	0	1	0	B	0	0	Rn				Rd				0	0	0	0	1	0	0	1	Rm								
Переход и обмен	Условие	0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	Rn						
Передача полу-слова данных, смещ. в регистре	Условие	0	0	0	P	U	0	W	L	Rn				Rd				0	0	0	0	1	S	H	1	Rm								
Передача полу-слова данных, пост. смещен.	Условие	0	0	0	P	U	1	W	L	Rn				Rd				Смещение				1	S	H	1	Смещение								
Одиноч.передача данных	Условие	0	1	1	P	U	B	W	L	Rn				Rd				Смещение																
Неопред.	Условие	0	1	1																			1											
Передача блока данных	Условие	1	0	0	P	U	S	W	L	Rn				Список регистров																				
Переход	Условие	1	0	1	L	Смещение																												
Передача данных сопроцессора	Условие	1	1	0	P	U	N	W	L	Rn				CRd				CP#				Смещение												
Операция над данными сопроцессора	Условие	1	1	1	0	CP Опс				CRn				CRd				CP#				CP		0	CRm									
Передача регистра сопроцессора	Условие	1	1	1	0	CP Опс				L	CRn				Rd				CP#				CP		1	CRm								
Программное прерывание	Условие	1	1	1	1	Игнорируется процессором																												

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

Архитектура ARM. Поле условия

Type		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				
Data Processing	REGOP	Cond				0	0	I	Opcode				S	Rn				Rd				shifter_operand															
Multiply	MULT	Cond				0	0	0	0	0	0	A	S	Rd				Rn				Rs				1	0	0	1	Rm							
Single Data Swap	SWAP	Cond				0	0	0	1	0	B	0	0	Rn				Rd				0	0	0	0	1	0	0	1	Rm							
Single Data Transfer	TRANS	Cond				0	1	I	P	U	B	W	L	Rn				Rd				Offset															
Block Data Transfer	MTRANS	Cond				1	0	0	P	U	S	W	L	Rn				Register List																			
Branch	BRANCH	Cond				1	0	1	L	Offset																											
Coprocessor Data Transfer	CODTRANS	Cond				1	1	0	P	U	N	W	L	Rn				CRd				CP#				Offset											
Coprocessor Data Operation	COREGOP	Cond				1	1	1	0	CP Opcode				CRn				CRd				CP#				CP		0	CRm								
Coprocessor Register Transfer	CORTTRANS	Cond				1	1	1	0	CP Opcode			L	CRn				Rd				CP#				CP		1	CRm								
Software Interrupt	SWI	Cond				1	1	1	1	Ignored by processor																											
		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				

Архитектура ARM. Формат команд

Флаг	Название. Порядок формирования
N	Флаг отрицательного результата N (Negative). Устанавливается в 1 (N=1), если в результате операции получено число, старший бит которого (31-й, знаковый) равен 1. Если старший бит равен нулю, то флаг N сбрасывается в 0 (N=0).
Z	Флаг нуля Z (Zero). Если в результате арифметической или логической операции получен нулевой результат (все 32 бита результата нулевые), то флаг выставляется (Z=1), в противном случае – сбрасывается (Z=0).
C	Флаг переноса C (Carry). Устанавливается (C=1), если в результате операции сложения двух 32-разрядных чисел произошел <i>перенос</i> за пределы разрядной сетки. В противном случае – сбрасывается (C=0). Свидетельствует о <i>переполнении при сложении чисел без знака</i> . Сбрасывается (C=0), если в результате операции вычитания двух 32-разрядных чисел произошел « <i>заем</i> » и выставляется (C=1) в противном случае (нет «заема»).
V	Флаг переполнения V (Overflow). Выставляется (V=1) в том случае, когда при выполнении арифметической операции сложения или вычитания чисел, рассматриваемых процессором как числа со знаком в дополнительном коде, возникло <i>переполнение</i> , т.е. полученный 32-разрядный результат вышел за пределы диапазона 32-разрядных чисел со знаком, в противном случае – сбрасывается (V=0).
Q	Флаг насыщения Q (saturation). Вырабатывается только двумя командами процессора – насыщение содержимого регистра как числа без знака USAT и насыщение содержимого регистра как числа со знаком SSAT. Если насыщение (ограничение на уровне максимального значения) имело место, то флаг устанавливается (Q=1), в противном случае – сбрасывается (Q=0).

Архитектура ARM. Поле условия

Все команды выполняются в зависимости состояния регистра PSR и поля условия самой команды (биты 31:28), содержащее условие, при которых команда будет выполнена. Если флаги **C, N, Z, V** *установлены* (сброшены) согласно коду поля условия, то команда будет **выполнена**, в противном случае эта команда будет **проигнорирована**.

Существуют 16 вариантов условий, каждое из которых определяется двухсимвольным суффиксом, добавляемым к мнемонике команды. Например, команда переход (Branch - "B" в ассемблере) становится командой перейти, если равно - BEQ (Branch if Equal), которая означает, что переход будет выполнен, если установлен флаг Z.

Практически можно использовать 15 различных условий, 16-е условие (1111) зарезервировано и не применяется.

При отсутствии в мнемонике команды условия выполнения или установлен суффикс AL, то эта команда будет выполнена безусловно, независимо от состояния флагов условия (регистр CPSR).

Архитектура ARM. Поле условия

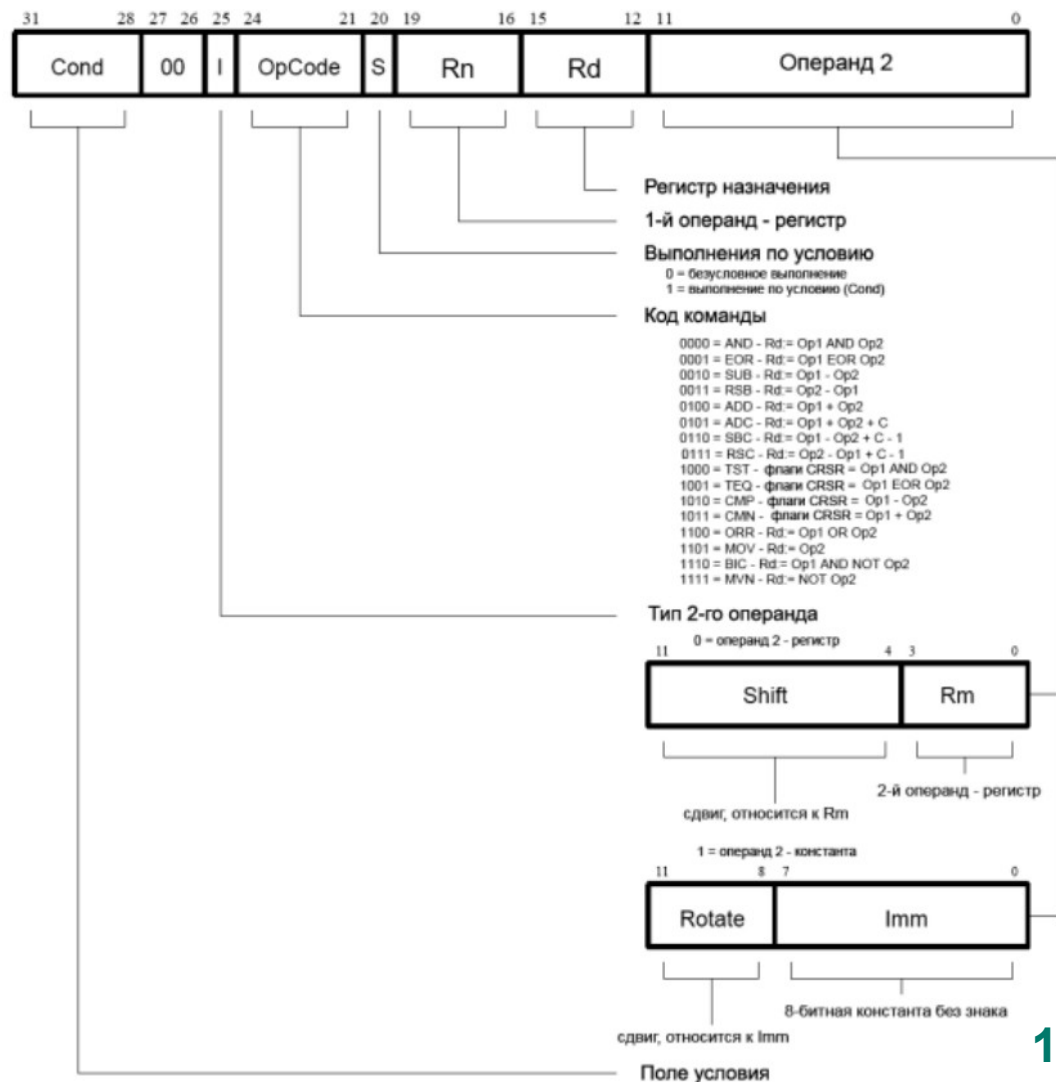
Суффикс	Флаги	Значение
EQ	Z=1	«Эквивалентно» (Equal)
NE	Z=0	«Не эквивалентно» (Not Equal)
HS или CS	C=1	«Выше или то же самое» (Higher or same) или «Флаг Carry установлен» (Carry Set) Больше или равно (\geq) при сравнении чисел без знака (unsigned)
LO или CC	C=0	«Ниже» (Lower) или «Флаг Carry сброшен» (Carry Clear) Строго меньше ($<$) при сравнении чисел без знака (unsigned)
MI	N=1	«Минус» (Negative) - Отрицательное
PL	N=0	«Плюс» (Positive or Zero) – Положительное или ноль
VS	V=1	«Переполнение знаковое» (Overflow)
VC	V=0	«Нет знакового переполнения» (No Overflow)
HI	(C=1) and (Z=0)	«Выше» (Higher) Строго больше ($>$) при сравнении чисел без знака (unsigned)
LS	(C=0) or (Z=1)	«Ниже или то же самое» (Lower or same) Меньше или равно (\leq) при сравнении чисел без знака (unsigned)
GE	N=V	«Больше или эквивалентно» (Greater then or equal) Больше или равно (\geq) при сравнении чисел со знаком (signed)
LT	N!=V	«Меньше чем» (Less then) Строго меньше ($<$) при сравнении чисел со знаком (signed)
GT	(Z=0) and (N=V)	«Строго больше» (Greater then) Строго больше ($>$) при сравнении чисел со знаком (signed)
LE	(Z=1) or (N!=V)	«Меньше чем или эквивалентно» (Less then or equal) Меньше или равно (\leq) при сравнении чисел со знаком (signed)
AL	Любые значения флагов	«Всегда». Суффикс по умолчанию, который используется всегда, когда код условного выполнения не специфицируется.

Архитектура ARM. Команды обработки данных

Команда формирует результат, выполняя указанную арифметическую или логическую операцию с одним или двумя операндами.

Первый операнд - всегда регистр (Rn).
Второй операнд может быть регистром со сдвигом (Rm), или 8-битной константой с циклическим сдвигом (Imm) (в зависимости от значения бита "I" в команде).
Флаги CPSR могут оставаться без изменений (S = 0) или выставлены в зависимости от результата выполнения этой команды (S = 1).

Некоторые операции в команде (TST, TEQ, CMP, CMN) не записывают результат в регистр Rd. Они используются только для того, чтобы в зависимости от результата установить/сбросить флаги CPSR (только при S = 1).



Архитектура ARM. Команды обработки данных

SUB, RSB, ADD, ADC, SBC, RSC, CMP, CMN обрабатывают каждый операнд как 32-битное целое число (беззнаковое или знаковое с дополнением до 2-х, что). Если "**S**" установлен (и Rd не R15) и произойдет переполнение, то флаг "**V**" в PSR будет установлен. Его можно игнорировать, если операнды беззнаковые, но нужно учитывать для знаковых операндов.

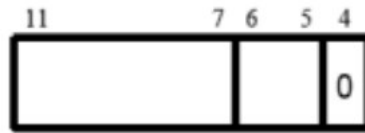
Флаг "**C**" будет установлен при переносе бита 31 из ALU (переполнение).

Флаг "**Z**" будет установлен, только если результат нулевой.

Флаг "**N**" будет установлен в зависимости от значения бита 31 результата операции (указывает на отрицательный результат, если операнды знаковые с дополнением до 2-х).

Мнемоника ассемблера	Код	Действие
AND	0000	операнд1 AND операнд2
EOR	0001	операнд1 EOR операнд2
SUB	0010	операнд1 - операнд2
RSB	0011	операнд2 - операнд1
ADD	0100	операнд1 + операнд2
ADC	0101	операнд1 + операнд2 + перенос
SBC	0110	операнд1 - операнд2 + перенос - 1
RSC	0111	операнд2 - операнд1 + перенос - 1
TST	1000	как AND, но результат не будет записан
TEQ	1001	as EOR, но результат не будет записан
CMP	1010	as SUB, но результат не будет записан
CMN	1011	as ADD, но результат не будет записан
ORR	1100	операнд1 OR операнд2
MOV	1101	операнд2 (операнд1 игнорируется)
BIC	1110	операнд1 AND NOT операнд2 (бит сброшен)
MVN	1111	NOT операнд2 (операнд1 игнорируется)

Архитектура ARM. Команды сдвига

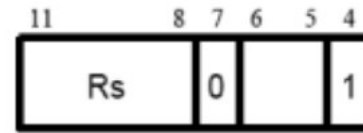


Тип сдвига

00 = логический влево
01 = логический вправо
10 = арифметический вправо
11 = циклический вправо

Число сдвигов

5-битное целое беззнаковое



Тип сдвига

00 = логический влево
01 = логический вправо
10 = арифметический вправо
11 = циклический вправо

Сдвигаемый регистр

Число сдвигов, указанное
в младшем байте регистра Rs

Второй операнд определен как сдвиговый регистр.

Операцией регистрового циклического сдвига управляет поле Shift в самой команде, указывая тип сдвига: логический сдвиг влево или вправо, арифметический сдвиг право, циклический сдвиг вправо.

Число бит, на которое будет произведен сдвиг, находится или в старшей части операнда 2 (сдвиг регистра кроме R15 при I = 0) или его в старших 4-х битах (сдвиг константы при I = 1).

Архитектура ARM. Особенности ассемблера

Команды с одним операндом: MOV,MVN

<opcode>{cond}{S} Rd, <Op2>

Команды без записи результата (без Rd): (CMP,CMN,TEQ,TST)

<opcode>{cond} Rn, <Op2>

Команды AND,EOR,SUB,RSB,ADD,ADC,SBC,RSC,ORR,BIC

<opcode>{cond}{S} Rd, Rn, <Op2>

- <Op2> - это Rm{,<shift>} или {,<#выражение>}
- {cond} - двухсимвольная мнемоника условия (см. Табл. слайд 16).
- {S} - разрешить условное выполнение (для команд CMP, CMN, TEQ, TST).
- Rd, Rn и Rm - выражения, результат вычисления которых - номер регистра общего назначения.
- <#выражение> - если это поле используется, то ассемблер попытается сгенерировать команду сдвига 8-битной константы с соответствии с указанным выражением. Если это невозможно, то возникнет ошибка.
- <shift> - <тип_сдвига> <регистр> или <тип_сдвига> #выражение, или RRX (расширенный циклический сдвиг вправо).
- <shiftname> - ASL, LSL, LSR, ASR, ROR. (так как ASL - синоним LSL, то для обеих команд будет сгенерирован одинаковый машинный код).

Архитектура ARM. Примеры ассемблерных команд

; умножение на 9 – это умножение числа на 8
; путем сдвига влево на 3 бита плюс еще число
`add r0, r1, r1, lsl #3` ; $r0 = r1 + (r1 \ll 3) = r1 * 9$

; умножение на 15 – это умножение на 16 минус число
`rsb r0, r1, r1, lsl #4` ; $r0 = (r1 \ll 4) - r1 = r1 * 15$

; доступ к таблице 4-х байтовых слов, где
; r1 – это базовый адрес таблицы
; r2 – это индекс элемента в таблице
`ldr r0, [r1, r2, lsl #2]`

Архитектура ARM. Примеры ассемблерных команд

ADDEQ	R2,R4,R5	; Если $Z = 1$, то $R2 := R4 + R5$
TEQS	R4,#3	; Проверить R4 на равенство 3 ; (ассемблер автоматически установит ; в единицу бит S)
SUB	R4,R5,R7,LSR R2	; Логический сдвиг вправо R7 на число бит, ; которое указано в старшем байте регистра R2, ; затем вычесть результат из R5 и разместить ; его в R4.
MOV	PC,R14	; Возврат из подпрограммы.
MOVS	PC,R14	; Возврат из исключения и восстановление PSR.

Архитектура ARM. Примеры ассемблерных команд

; Применение условий для логического ИЛИ

```
CMP      Rn,#p      ; Если Rn=p OR Rm=q то перейти на Label.  
BEQ      Label  
CMP      Rm,#q  
BEQ      Label
```

; Эквивалентный код:

```
CMP      Rn,#p  
CMPNE    Rm,#q      ; Если условие не выполнено, то другой тест.  
BEQ      Label
```

; Абсолютное значение

```
TEQ      Rn,#0      ; Проверить знак  
RSBMI    Rn,Rn,#0    ; и, если необходимо, дополнить до 2-х.
```

; Алгоритм Евклида:

```
loop CMP Ri, Rj          ; проверка условий NE ( $i \neq j$ ),  
                           ; GT ( $i > j$ ),  
                           ; и LT ( $i < j$ );  
      SUBGT Ri, Ri, Rj    ; если GT, выполняется  $i = i - j$ ;  
      SUBLT Rj, Rj, Ri    ; если LT -  $j = j - i$ ;  
      BNE loop           ; если NE - переход на  
                           ; метку loop.
```

Используемые источники

Википедия. ARM (архитектура):

[https://ru.wikipedia.org/wiki/ARM_\(архитектура\)](https://ru.wikipedia.org/wiki/ARM_(архитектура))

Изучение системы команд процессора ARM:

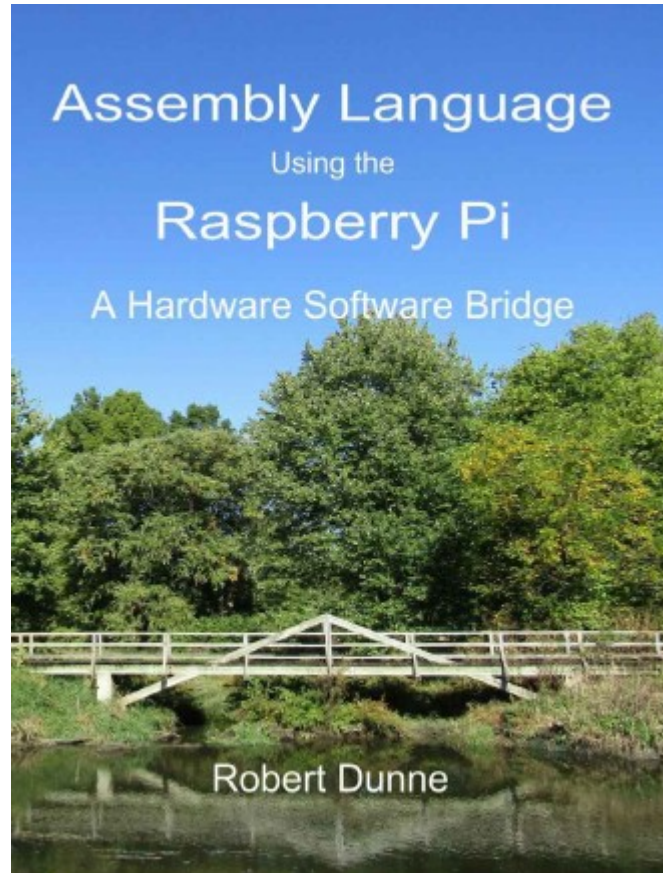
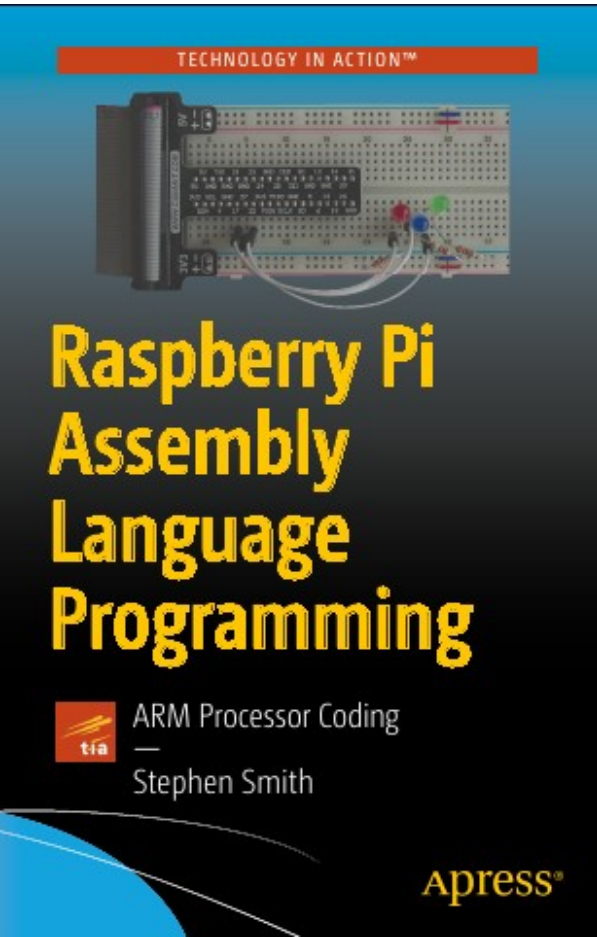
<https://marsohod.org/index.php/prodmarsohod2/amber-arm-soc/226-arm-instr>

Система команд в режиме ARM

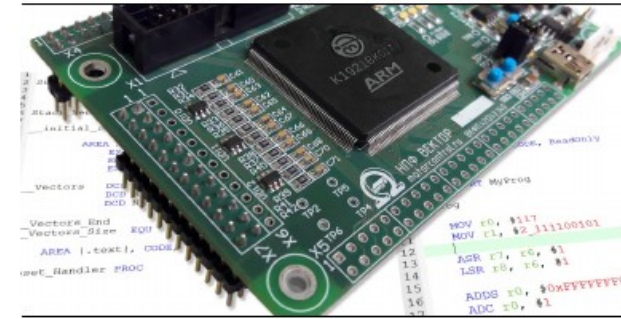
http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/asm_arm/

Практический курс микропроцессорной техники на базе процессорных ядер ARM-Cortex-M3/M4/M4F [электронный ресурс]: учебное пособие – электрон. текстовые дан. (12 Мб) / В.Ф. Козаченко, А.С. Анучин, Д. И. Алямкин и др.; под общ. ред. В.Ф. Козаченко. – М.: Издательство МЭИ, 2019.

Используемые источники



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

Архитектура, система команд, разработка и отладка
программного обеспечения на Ассемблере в интегрированной
среде Keil μVision

Цифровая схемотехника и архитектура компьютера



Дэвид М. Харрис и Сара Л. Харрис

Цветное издание



Цифровая схемотехника
и архитектура компьютера
Дополнение
по архитектуре **ARM**



Дэвид М. Харрис
Сара Л. Харрис



Дэвид М. Харрис, Сара Л. Харрис

Цифровая схемотехника и архитектура компьютера. – М.: ДМК Пресс, 2018. – 792 с.

Дэвид М. Харрис, Сара Л. Харрис

Цифровая схемотехника и архитектура компьютера. Дополнение по архитектуре ARM – М.: ДМК Пресс, 2019. – 356 с.