

10. Процессорные архитектуры, ч. 4. Наборы инструкций MIPS, ARM.

MIPS

MIPS (англ. Microprocessor without Interlocked Pipeline Stages) — микропроцессор, разработанный компанией MIPS Computer Systems (в настоящее время MIPS Technologies) в соответствии с концепцией проектирования процессоров RISC (то есть для процессоров с упрощенным набором команд). Ранние модели процессора имели 32-битную структуру, позднее появились его 64-битные версии. Существует множество модификаций процессора, включая MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS32 и MIPS64, из них действующими являются MIPS32 (для 32-битной реализации) и MIPS64 (для 64-битной реализации). MIPS32 и MIPS64 определяют как набор регистров управления, так и набор команд.

В настоящее время различные реализации MIPS используются в основном во встроенных системах, например, в смартфонах, маршрутизаторах, шлюзах, а также в игровых консолях, таких, как Sony PlayStation 2 и Sony PlayStation Portable.

Формат инструкций MIPS I

Инструкции делятся на три типа: R, I и J. Каждая инструкция начинается с 6-битного кода. В дополнение к коду, инструкции R-типа определяют три регистра, область размера сдвига регистра, и область функции; инструкции I-типа определяют два регистра и непосредственное значение; инструкции J-типа состоят из кода операции и 26-битного адреса перехода.

Далее приведена таблица применения трех форматов инструкции в архитектуре ядра:




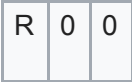


Тип	-31- формат (в битах) -0-					
R	код (6)	rs (5)	rt (5)	rd (5)	shamt (5)	функция (6)
I	код (6)	rs (5)	rt (5)	непосредственное (16)		
J	код (6)	адрес (26)				

Язык ассемблера MIPS

1) Целочисленные операции

MIPS имеет 32 регистра для целочисленных операций. Для выполнения арифметических вычислений данные должны находиться в регистрах. Регистр \$0 всегда хранит 0, а регистр \$1 резервируется для сборки (для хранения псевдоинструкций и больших констант).

Нижеприведенная таблица показывает, какие биты каким частям инструкции соответствуют. Дефис (-) обозначает нейтральное состояние. (полная таблица по ссылке [wiki](#))

Операция	Описание	Синтаксис инструкции	Описание	Адресат/код/функция	Замечания/Кодирование
Арифметическая		$Op, \$s, \t	$s + \$t$		Использует два регистра, выполняет прерывание при переполнении
Загрузка данных	word	$Op(\$s)$	$memory[\$s + C]$		Загружает word из: $MEM[\$s+C]$ и следующих 3 байтов.
Логическая		$Op, \$s, \t	$s \& \$t$		Логическая конъюнкция
Логический сдвиг	Shift logical	$Op, \$s, C$	$s \ll C$		Сдвигает C битов влево
Условное ветвление	if on equal	$Op, \$t, C$	$(s == \$t) \text{ go to } PC+4+4 \cdot C$		Переходит к инструкции по указанному адресу, если два регистра равны.
Безусловный переход			$PC+4+4[31:28] \cdot C$		Выполняет безусловный переход к инструкции по указанному адресу.

2) Операции над числами с плавающей запятой

MIPS имеет 32 регистра с плавающей запятой. Регистры соединены по 2 для двойной точности вычислений. Регистры с нечетными номерами не могут быть использованы для арифметических операций или ветвления, они могут лишь частично указывать двойную точность в паре регистров (табл по ссылке)

3) Псевдоинструкции

Эти инструкции принимаются языком ассемблера MIPS, однако они не являются реальными. Ассемблер переводит их в последовательности настоящих инструкций

Название	Синтаксис инструкции	Трансляция в обычную инструкцию	значение
Load Address	la \$1, LabelAddr	lui \$1, LabelAddr[31:16]; ori \$1,\$1, LabelAddr[15:0]	\$1 = Маркировка адреса
Load Immediate	li \$1, IMMED[31:0]	lui \$1, IMMED[31:16]; ori \$1,\$1, IMMED[15:0]	\$1 = 32-битное прямое значение
Branch if greater than	bgt \$rs,\$rt,Label	slt \$at,\$rt,\$rs; bne \$at,\$zero,Label	if(R[rs]>R[rt]) PC=Label
Branch if less than	blt \$rs,\$rt,Label	slt \$at,\$rs,\$rt; bne \$at,\$zero,Label	if(R[rs]<R[rt]) PC=Label
Branch if greater than or equal	bge	etc.	if(R[rs]>=R[rt]) PC=Label
Branch if less than or equal	ble		if(R[rs]<=R[rt]) PC=Label
Branch if greater than unsigned	bgtu		if(R[rs]>=R[rt]) PC=Label
Branch if greater than zero	bgtz		if(R[rs]>0) PC=Label
Multiplies and returns only first 32 bits	mul \$1, \$2, \$3	mult \$2, \$3; mflo \$1	\$1 = \$2 * \$3

Ссылки

[https://ru.wikipedia.org/wiki/MIPS_\(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0\)](https://ru.wikipedia.org/wiki/MIPS_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0))

ARM

Архитектура ARM (от англ. Advanced RISC Machine — усовершенствованная RISC-машина; иногда — Acorn RISC Machine) — семейство лицензируемых 32-битных и 64-битных микропроцессорных ядер разработки компании ARM Limited.

Среди лицензиатов - AMD, Apple, Analog Devices, Atmel, Xilinx, Cirrus Logic, Intel, Marvell, NXP, STMicroelectronics, Samsung, LG, MediaTek, Qualcomm, Sony, Texas Instruments, Nvidia, Freescale, Миландр, ЭЛВИС, HiSilicon.

Процессоры ARM

В настоящее время значимыми являются несколько семейств процессоров ARM:

- ARM7 (с тактовой частотой до 60-72 МГц), предназначенные, например, для недорогих мобильных телефонов и встраиваемых решений средней производительности. В настоящее время активно вытесняется новым семейством Cortex.
- ARM9 (англ.)русск., ARM11 (с частотами до 1 ГГц) для более мощных телефонов, карманных компьютеров и встраиваемых решений высокой производительности.
- Cortex A — новое семейство процессоров на смену ARM9 и ARM11.
- Cortex M — новое семейство процессоров на смену ARM7, также призванное занять новую для ARM нишу встраиваемых решений низкой производительности. В семействе присутствуют четыре значимых ядра:
- Cortex-M0, Cortex-M0+ (более энергоэффективное) и Cortex-M1 (оптимизировано для применения в ПЛИС) с архитектурой ARMv6-M;
- Cortex-M3 с архитектурой ARMv7-M;

- Cortex-M4 (добавлены SIMD-инструкции, опционально FPU) и Cortex-M7 (FPU с поддержкой чисел одинарной и двойной точности) с архитектурой ARMv7E-M;
- Cortex-M23 и Cortex-M33 с архитектурой ARMv8-M ARMv8-M .

В 2010 году производитель анонсировал процессоры Cortex-A15 под кодовым названием Eagle, ARM утверждает, что ядро Cortex A15 на 40 процентов производительнее на той же частоте, чем ядро Cortex-A9 при одинаковом числе ядер на чипе. Изделие, изготовленное по 28-нанометровому техпроцессу, имеет 4 ядра, может функционировать на частоте до 2,5 ГГц и будет поддерживаться многими современными операционными системами.

Архитектура

Уже давно существует справочное руководство по архитектуре ARM, которое разграничивает все типы интерфейсов, которые поддерживает ARM, так как детали реализации каждого типа процессора могут отличаться. 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).

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.

Набор команд ARM

Режим, в котором исполняется 32-битный набор 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

Регистры

ARM предоставляет 31 регистр общего назначения 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 регистров общего назначения (r0..r12).
- Stack Pointer (r13) — содержит указатель стека выполняемой программы.
- Link register (r14) — содержит адрес возврата в инструкциях ветвления.
- Program Counter (r15) — биты [31:1] содержат адрес выполняемой инструкции.
- Current Program Status Register (CPSR) — содержит флаги, описывающие текущее состояние процессора. Модифицируется при выполнении многих инструкций: логических, арифметических, и др.

Во всех режимах, кроме User mode и System mode, доступен также Saved Program Status Register (SPSR). После возникновения исключения регистр CPSR сохраняется в SPSR. Тем самым фиксируется состояние процессора (режим, состояние; флаги арифметических, логических операций, разрешения прерываний) на момент непосредственно перед прерыванием.

Краткое описание инструкций ARM (некраткое)

Операции		Синтаксис Ассемблера
Пересылка	Пересылка	MOV {cond}{S} Rd, <Oprnd2>
	Пересылка NOT	MVN {cond}{S} Rd, <Oprnd2>

	Пересылка SPSR в регистр	MRS {cond} Rd, SPSR
	Пересылка CPSR в регистр	MRS {cond} Rd, CPSR
	Пересылка регистра SPSR	MSR {cond} SPSR{field}, Rm
	Пересылка CPSR	MSR {cond} CPSR{field}, Rm
	Пересылка константы во флаги SPSR	MSR {cond} SPSR_f, #32bit_Imm
	Пересылка константы во флаги CPSR	MSR {cond} CPSR_f, #32bit_Imm
Арифметические	Сложение	ADD {cond}{S} Rd, Rn, <Oprnd2>
	Сложение с переносом	ADC {cond}{S} Rd, Rn, <Oprnd2>
	Вычитание	SUB {cond}{S} Rd, Rn, <Oprnd2>
	Вычитание с переносом	SBC {cond}{S} Rd, Rn, <Oprnd2>
	Вычитание обратного вычитания	RSB {cond}{S} Rd, Rn, <Oprnd2>
	Вычитание обратного вычитания с переносом	RSC {cond}{S} Rd, Rn, <Oprnd2>
	Умножение	MUL {cond}{S} Rd, Rm, Rs
	Умножение-накопление	MLA {cond}{S} Rd, Rm, Rs, Rn
	Умножение длинных беззнаковых чисел	UMULL {cond}{S} RdLo, RdHi, Rm, Rs
	Умножение - беззнаковое накопление длинных значений	UMLAL {cond}{S} RdLo, RdHi, Rm, Rs
	Умножение знаковых длинных	SMULL {cond}{S} RdLo, RdHi, Rm, Rs
	Умножение - знаковое накопление длинных значений	SMLAL {cond}{S} RdLo, RdHi, Rm, Rs
	Сравнение	CMP {cond} Rd, <Oprnd2>
	Сравнение отрицательное	CMN {cond} Rd, <Oprnd2>
Логические	Проверка	TST {cond} Rn, <Oprnd2>
	Проверка на эквивалентность	TEQ {cond} Rn, <Oprnd2>
	Лог. И	AND {cond}{S} Rd, Rn, <Oprnd2>
	Искл. ИЛИ	EOR {cond}{S} Rd, Rn, <Oprnd2>
	ORR	ORR {cond}{S} Rd, Rn, <Oprnd2>
	Сброс бита	BIC {cond}{S} Rd, Rn, <Oprnd2>>
Переход	Переход	B {cond} label

	Переход по ссылке	BL {cond} label
	Переход и изменение набора инструкций	BX {cond} Rn
Чтение	слова	LDR {cond} Rd, <a_mode2>
	слова с преимуществом режима пользователя	LDR {cond}T Rd, <a_mode2P>
	байта	LDR {cond}B Rd, <a_mode2>
	байта с преимуществом режима пользователя	LDR {cond}BT Rd, <a_mode2P>
	байта со знаком	LDR {cond}SB Rd, <a_mode3>
	полуслова	LDR {cond}H Rd, <a_mode3>
	полуслова со знаком	LDR {cond}SH Rd, <a_mode3>
	операции с несколькими блоками данных	-
	 с предварительным инкрементом	LDM {cond}IB Rd{!}, <reglist>{^}
	 с последующим инкрементом	LDM {cond}IA Rd{!}, <reglist>{^}
	 с предварительным декрементом	LDM {cond}DB Rd{!}, <reglist>{^}
	 с последующим декрементом	LDM {cond}DA Rd{!}, <reglist>{^}
	 операция над стеком	LDM {cond}<a_mode4L> Rd{!}, <reglist>
	 операция над стеком и восстановление CPSR	LDM {cond}<a_mode4L> Rd{!}, <reglist+pc>^
	операция над стеком с регистрами пользователя	LDM {cond}<a_mode4L> Rd{!}, <reglist>^
Запись	слова	STR {cond} Rd, <a_mode2>
	слова с преимуществом режима пользователя	STR {cond}T Rd, <a_mode2P>
	байта	STR {cond}B Rd, <a_mode2>
	байта с преимуществом режима пользователя	STR {cond}BT Rd, <a_mode2P>
	полуслова	STR {cond}H Rd, <a_mode3>
	операции над несколькими блоками данных	-
	 с предварительным инкрементом	STM {cond}IB Rd{!}, <reglist>{^}
	 с последующим инкрементом	STM {cond}IA Rd{!}, <reglist>{^}
	 с предварительным декрементом	STM {cond}DB Rd{!}, <reglist>{^}
	о с последующим декрементом	STM {cond}DA Rd{!}, <reglist>{^}

	■ операция над стеком	STM {cond}<a_mode4S> Rd{!}, <reglist>
	■ операция над стеком с регистрами пользователя	STM {cond}<a_mode4S> Rd{!}, <reglist>^
Обмен	слов	SWP {cond} Rd, Rm, [Rn]
	байт	SWP {cond}B Rd, Rm, [Rn]
Сопроцессор	Операция над данными	CDP {cond} p<cpnum>, <op1>, CRd, CRn, CRm, <op2>
	Пересылка в ARM-регистр из сопроцессора	MRC {cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>
	Пересылка в сопроцессор из ARM-регистра	MCR {cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>
	Чтение	LDC {cond} p<cpnum>, CRd, <a_mode5>
	Запись	STC {cond} p<cpnum>, CRd, <a_mode5>
Программное прерывание		SWI 24bit_Imm

Ссылки

https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0

http://www.gaw.ru/html.cgi/txt/doc/micros/arm/arh_7dtmi/insruct.htm