Архитектура x86-64

М. А. Соловьёв

9 октября 2013, ВМК МГУ

Историческая справка

- 2000: оригинальная спецификация от **AMD**.
- 2003: первая реализация **AMD**: **Opteron**.
- 2004: первая реализация **Intel**: **Xeon Nocona**.
- 2008: первая реализация **VIA**: **Nano**.

Поддержка в процессорах

- AMD
  - Athlon 64
  - Athlon II
  - Opteron
  - Turion 64
  - Sempron
  - Phenom
  - Phenom II
  - FX
  - Fusion

Поддержка в процессорах

- Intel
  - NetBurst
    - Xeon, Celeron, Pentium 4, Pentium D, Pentium Extreme Edition
  - Core
    - Xeon, Core 2, Pentium Dual-Core, Celeron
  - Atom
    - 200, 300, N4xx, N5xx, Dxxx
  - Nehalem
    - Core i3, Core i5, Core i7
  - Sandy Bridge
    - Core i3, Core i5, Core i7

Поддержка в процессорах

- VIA
  - Nano

Название

- Всё семейство реализаций:
  - **x86-64**;
  - **x86_64**;
  - **x64**.
- Реализация AMD:
  - **AMD64**;
  - **AA64**.
- Реализация Intel:
  - **Intel 64**;
  - **IA-32e**;
  - **EM64T**.

Название

- **IA-64** — это **Intel Itanium**, другая 64-разрядная процессорная архитектура от **Intel**.

Регистры общего назначения

- Расширение 32-разрядных GPR до 64 разрядов:
  - RAX > EAX > AX > AL (AH);
  - RCX > ECX > CX > CL (CH);
  - RDX > EDX > DX > DL (DH);
  - RBX > EBX > BX > BL (BH);
  - RSP > ESP > SP > SPL;
  - RBP > EBP > BP > BPL;
  - RSI > ESI > SI > SIL;
  - RDI > EDI > DI > DIL.

Регистры общего назначения

- Дополнительные 8 GPR:
  - R8 > R8D > R8W > R8B;
  - R9 > R9D > R9W > R9B;
  - R10 > R10D > R10W > R10B;
  - R11 > R11D > R11W > R11B;
  - R12 > R12D > R12W > R12B;
  - R13 > R13D > R13W > R13B;
  - R14 > R14D > R14W > R14B;
  - R15 > R15D > R15W > R15B.

Специальные регистры

- Регистр счётчика инструкций:
  - `RIP` > `EIP` > `IP`.
- Регистр флагов:
  - `RFLAGS` > `EFLAGS` > `FLAGS`.
- Управляющие регистры расширены до 64 разрядов:
  - `CR`*n*, `DR`*n*, `GDTR`, `LDTR`, `IDTR`, `TR`, `MSR_`*nnnn*`_`*nnnn*.

Базовые команды

- `ADD RAX, QWORD [RSI + 8 * R8]`
- `ADD EAX, DWORD [ESI + 4 * R8W]`
- `JMP RDI`
- `JMP QWORD [RDI]`

- При выработке 32-разрядного значения в GPR происходит неявное расширение нулём до 64 разрядов.

Базовые команды

- RAX = 0002 0001 8000 2201;
  RBX = 0002 0002 0123 3301.

- ADD RBX, RAX
  → RBX = 0004 0003 8123 5502.
- ADD EBX, EAX
  → RBX = 0000 0000 8123 5502.
- ADD BX, AX
  → RBX = 0002 0002 0123 5502.
- ADD BL, AL
  → RBX = 0002 0002 0123 3302.

Базовые команды

- Команды пересылки
  - Пересылка
    - MOV, MOVSX(D), MOVZX, MOVD, MOVNTI
  - Условная пересылка
    - CMOV*cc*
  - Операции со стеком
    - POP, POPA(D), PUSH, PUSHA(D), ENTER, LEAVE

Базовые команды

- Команды преобразования данных
  - Знаковое расширение
    - `CBW, CWDE, CDQE, CWD, CDQ, CQO`
  - Извлечение знаковой маски
    - `MOVMSKPS, MOVMSKPD`
  - Трансляция
    - `XLAT`
  - ASCII и BCD
    - `AAA, AAD, AAM, AAS`
    - `DAA, DAS`
  - Порядок байтов
    - `BSWAP`

Базовые команды

- Команды загрузки сегментных регистров
  - `LDS, LES, LFS, LGS, LSS`
  - `MOV`
  - `POP`
- Команда загрузки эффективного адреса
  - `LEA`

Базовые команды

- Арифметические команды
  - Сложение и вычитание
    - `ADC, ADD, SBB, SUB, NEG`
  - Умножение и деление
    - `MUL, IMUL, DIV, IDIV`
  - Инкремент и декремент
    - `DEC, INC`
- Команды сдвига и вращения
  - Вращение
    - `RCL, RCR, ROL, ROR`
  - Сдвиг
    - `SAL, SAR, SHL, SHR, SHLD, SHRD`

Базовые команды

- Команды сравнения и тестирования
  - Сравнение
    - `CMP`
  - Тестирование
    - `TEST`
  - Сканирование битов
    - `BSF, BSR`
  - Подсчёт битов
    - `POPCNT, LZCNT`
  - Тестирование битов
    - `BT, BTC, BTR, BTS`
  - Условная установка значения
    - `SET`*`cc`*
  - Проверка принадлежности диапазону
    - `BOUND`

Базовые команды

- Логические команды
  - AND, OR, XOR, NOT
- Строковые команды
  - CMPS, CMPSB, CMPSW, CMPSD, CMPSQ
  - SCAS, SCASB, SCASW, SCASD, SCASQ
  - MOVS, MOVSB, MOVSW, MOVSD, MOVSQ
  - LODS, LODSB, LODSW, LODSD, LODSQ
  - STOS, STOSB, STOSW, STOSD, STOSQ

Базовые команды

- Команды передачи управления
  - Прыжок
    - `JMP`
  - Условный прыжок
    - `J`*`cc`*`, JCXZ, JECXZ, JRCXZ`
  - Цикл
    - `LOOP`*`cc`*
  - Вызов подпрограммы
    - `CALL`
  - Возврат из подпрограммы
    - `RET`
  - Прерывания и исключения
    - `INT, INTO, IRET, IRETD, IRETQ`

Базовые команды

- Команды работы с флагами
  - Стек
    - `POPF, POPFD, POPFQ, PUSHF, PUSHFD, PUSHFQ`
  - Установка и сброс
    - `CLC, CMC, STC, CLD, STD, CLI, STI`
  - Загрузка и сохранение
    - `LAHF, SAHF`

Базовые команды

- Команды ввода/вывода
  - Общий ввод/вывод
    - IN, OUT
  - Строковый ввод/вывод
    - INS, INSB, INSW, INSD
    - OUTS, OUTSB, OUTSW, OUTSD
- Семафорные команды
  - CMPXCHG, CMPXCHG8B, CMPXCHG16B
  - XADD
  - XCHG

Базовые команды

- Команды запроса информации
  - CPUID
- Команды управления кешем
  - LFENCE, SFENCE, MFENCE
  - PREFETCH*l*, PREFETCH, PREFETCHW
  - CLFLUSH
- Команда, которая ничего не делает
  - NOP
- Команды системных вызовов
  - SYSENTER, SYSEXIT
  - SYSCALL, SYSRET

Регистры XMM

- 128-разрядные регистры из набора SSE, добавлено 8 новых.
  - `XMM0, XMM1, XMM2, XMM3, XMM4, XMM5, XMM6, XMM7, XMM8, XMM9, XMM10, XMM11, XMM12, XMM13, XMM14, XMM15.`
- Скалярные форматы данных:
  - беззнаковое целое: 1, 8, 16, 32, 64 или 128 разрядов;
  - число с плавающей точкой: 32 или 64 разряда.

Регистры XMM

- Векторные форматы данных:
  - беззнаковые целые: 8 (16 шт.), 16 (8 шт.), 32 (4 шт.) или 64 (2 шт.) разряда;
  - знаковые целые: 8 (16 шт.), 16 (8 шт.), 32 (4 шт.) или 64 (2 шт.) разряда;
  - числа с плавающей точкой: одинарной точности (4 шт.) или двойной точности (2 шт.).

Примеры команд

- MOVD XMM0, RAX
- MOVDQA XMM8, OWORD [RSI]
- MOVDQU OWORD [RDI], XMM8

- CVTDQ2PS XMM1, XMM0

- PADDD XMM1, XMM0
- PSADBW XMM1, XMM0

Наборы инструкций

- SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4a, SSE4.2
- Intel AVX, AVX2
- Intel AES
- Мелкие наборы инструкций

Режимы адресации

- Режим адресации — способ кодирования операнда.
- Различные инструкции поддерживают различные режимы адресации.
- Режим адресации может описывать константу, регистр или данное в памяти.

Режимы адресации памяти

- Адресация памяти:
  - *сегментный* регистр (есть умолчание);
  - регистр *базы* (опционально);
  - регистр *индекса* и *множитель* (опционально);
  - *смещение* (опционально).

Режимы адресации памяти

- Формат инструкции:
  - ноль или более legacy-префиксов;
  - необязательный REX-префикс;
  - 1- или 2-байтовый код операции;
  - необязательный байт ModRM или пара байтов ModRM и SIB;
  - необязательное смещение;
  - необязательное значение операнда-константы.

Режимы адресации памяти

- REX:
  - имеет коды с `0x40` по `0x4F`;
  - заменяет однобайтовые кодировки `INC` и `DEC`;
  - содержит четыре бита:
    - `REX.W` — 64-разрядный размер операнда;
    - `REX.R` — доп. разряд к полю ModRM `reg`;
    - `REX.X` — доп. разряд к полю ModRM `index`;
    - `REX.B` — доп. разряд к полю ModRM `r/m`.

Режимы адресации памяти

- ModRM и SIB:
  - в совокупности задают 5 полей: `mod`, `r/m`, `reg`, `index`, `scale`;
  - если при инструкции есть REX-префикс, поля `r/m`, `reg`, `index` дополняются соответствующим старшим разрядом;
  - в **x86-64** эти поля могут ссылаться на все 16 регистров общего назначения или XMM;
  - в **x86** эти поля могли ссылаться на 8 регистров общего назначения или XMM.

Режимы адресации памяти

- В x86-64 разрешено использовать в качестве базы счётчик инструкций RIP.
- В качестве значения будет выступать адрес *следующей* инструкции, так как текущая уже была выбрана процессором.
- Пример:
  - `MOV RAX, QWORD [RIP + 0x3078]`

Режимы адресации памяти

- Ключевые слова NASM.
  - `ADD EAX, DWORD [REL foo]`
  - `ADD EAX, DWORD [ABS foo]`
- Поведение по умолчанию.
  - `DEFAULT ABS`
  - `DEFAULT REL`
    - Автоматически не распространяется на память, адресуемую в сегментах FS и GS.

Режимы адресации памяти

- Регистры SPL, BPL, SIL, DIL имеют те же номера, что и AH, CH, DH, BH.
- Если инструкция имеет REX-префикс, можно использовать только SPL, BPL, SIL, DIL; AH, CH, DH, BH недоступны.
- Наоборот, если инструкции не имеет REX-префикса, можно использовать только AH, CH, DH, BH; SPL, BPL, SIL, DIL недоступны.
- Пример:
  - `CMP CH, BYTE [R8]` — недопустимо;
  - `CMP CL, BYTE [R8]` — допустимо.

Режимы работы процессора

- Legacy режим
  - Реальный режим
  - Защищённый режим
  - Режим виртуального 8086
- Длинный режим
  - Режим совместимости
  - 64-разрядный режим
- Режим системного менеджмента

Режимы работы процессора

	64-разрядный режим	Режим совместимости
Адресация относительно RIP	+	–
Размер адреса	64	32
Размер операнда	32	32
Загрузка 64-битного числа с MOV	+	–
Поддержка V8086	–	–
Поддержка задач	–	–
64-битные виртуальные адреса	+	+
Сегменты	FS, GS	все
Дескрипторы	64-разрядные	64-разрядные

Соглашения о связях

- System V AMD64
- Microsoft

Соглашения о связях

- System V AMD64
  - Необходимо сохранять RBP, RBX, R12, R13, R14, R15.
  - При использовании инструкций MMX требуется выполнить FEMMS.
  - Флаг DF должен быть сброшен.
  - Управляющие биты MXCSR и FPU сохраняет вызываемый.
  - Статусные биты MXCSR и FPU сохраняет вызывающий.

Соглашения о связях

- System V AMD64
  - Диаграмма стека при вызове:

Адрес	Содержимое	Фрейм
`RBP + 8n + 16`	Параметр в памяти *n*	Предыдущий
	…	
`RBP + 16`	Параметр в памяти 0	
`RBP + 8`	Адрес возврата	Текущий
`RBP`	Предыдущее значение RBP	
`RBP – 8`	Не определено	
	…	
`RSP`	(переменный размер)	
`RSP – 128`	Красная зона	

Соглашения о связях

- System V AMD64
  - Для передачи целочисленных параметров могут использоваться регистры:
    - `RDI`, `RSI`, `RDX`, `RCX`, `R8`, `R9`.
  - Для передачи SSE-параметров могут использоваться регистры:
    - `XMM0`, `XMM1`, `XMM2`, `XMM3`, `XMM4`, `XMM5`, `XMM6`, `XMM7`.
  - Два последовательных SSE-параметра могут быть упакованы в один регистр.
  - Все остальные параметры передаются через стек.

Соглашения о связях

- System V AMD64
  - Возврат целых чисел:
    - `RAX` и `RDX`.
  - Возврат SSE-значений:
    - `XMM0` и `XMM1`.
  - Два последовательных SSE-значения могут быть упакованы в один регистр.
  - Возврат FPU-значений:
    - `ST(0)` (`ST(1)` для комплексных чисел).
  - Два последовательных FPU-значения могут быть упакованы в один регистр.
  - Все остальные значения записываются в область памяти, передаваемую в качестве неявного параметра 0 (`RDI`); на выходе `RAX := RDI`.

Соглашения о связях

- System V AMD64
  - `typedef struct { int a, b; double d; } structparm;`

```
extern void
func(int e, int f, structparm s, int g, int h,
     long double ld, double m, __m128 y, double n,
     int i, int j, int k);
```

GPR		FPR		Стек	
RDI:	e	**XMM0:**	s.d	**0:**	ld
RSI:	f	**XMM1:**	m	**16:**	j
RDX:	s.a, s.b	**XMM2:**	y	**24:**	k
RCX:	g	**XMM3:**	n		
R8:	h				
R9:	i				

Соглашения о связях

- Microsoft
  - Необходимо сохранять RBP, RBX, RSI, RDI, R12, R13, R14, R15, XMM6..XMM15.
  - Флаг DF должен быть сброшен.
  - Управляющие биты MXCSR и FPU сохраняет вызываемый.
  - Статусные биты MXCSR и FPU сохраняет вызывающий.

Соглашения о связях

- Microsoft
  - Стек содержит место для локальных переменных и сохранённых регистров.
  - Стек содержит место для сохранения указателя фрейма.
  - Стек содержит место для `alloca()`.
  - Стек содержит место для параметров, переданных в функцию.
  - Стек содержит место для параметров, передаваемых из функции.

Соглашения о связях

- Microsoft
  - Для передачи целочисленных параметров могут использоваться регистры:
    - `RCX`, `RDX`, `R8`, `R9`.
  - Для передачи FP-параметров могут использоваться регистры:
    - `XMM0`, `XMM1`, `XMM2`, `XMM3`.
  - Все остальные параметры передаются через стек.

Соглашения о связях

- Microsoft
  - `func1(float a, double b, float c, double d, float e)`
    - `XMM0 (a), XMM1 (b), XMM2 (c), XMM3 (d), stack (e)`
  - `func2(__m64 a, __m128 b, struct c, float d)`
    - `RCX (a), RDX (&b), R8 (&c), XMM3 (d)`

Заключение

- Архитектура **x86-64** — эволюционный путь развития **x86**.
- Новые возможности добавлены с максимальным сохранением обратной совместимости.
- Убраны некоторые пережитки с 80-х годов: BCD-арифметика, сегменты, аппаратная поддержка задач и т. д.
- Наведён относительный порядок в наборе инструкций.

Заключение

- Появляющиеся вновь расширения набора инструкций нарушают порядок.
- Различные механизмы виртуализации у Intel, AMD и VIA не совместимы между собой.
- Набор инструкций AVX меняет модель программирования для SSE.
- Очень сложные инструкции (AES).