Анализ кода и информационная безопасность

Дополнительные материалы

Архитектура x86-64: краткая справка

МГУ / ВМК / СП

Историческая справка

- 2000 — оригинальная спецификация 64-разрядного расширения архитектуры x86 от AMD.
- 2003 — первая реализация «в кремнии» AMD — Opteron.
- 2004 — первая реализация «в кремнии» Intel — Xeon Nocona.
- 2008 — первая реализация «в кремнии» VIA — Nano.

Название

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

IA-64 — это Intel Itanium, совсем другая 64-разрядная архитектура.

Режимы работы процессора

- Наследственные режимы:
  - реальный режим (real mode):
    - «настоящий» реальный режим;
    - режим V8086;
    - режим системного менеджмента (SMM);
  - защищённый режим (protected mode):
    - 16-разрядный;
    - 32-разрядный.
- Длинный режим:
  - режим совместимости (compatibility mode);
  - 64-разрядный режим.

Длинный режим

- Длинный режим имеет два подрежима:
  - режим совместимости — для «старых» 32-разрядных программ;
  - 64-разрядный режим — для «новых» 64-разрядных программ.
- Режим совместимости нужен, чтобы можно было в рамках одной системы запускать и 32-, и 64-разрядные программы.
- Общие характеристики длинного режима:
  - 64-разрядные (в теории) физические адреса;
  - расширенный формат системных таблиц дескрипторов, позволяющий использовать 64-разрядные адреса.
- В остальном нововведения касаются только 64-разрядного режима.

64-разрядный режим

- Удвоенное количество регистров общего назначения.
- Удвоенный размер регистров общего назначения, счётчика команд, слова состояния, системных регистров.
- 64-разрядная (в теории) адресация виртуальной памяти:
  - сегментная трансляция отключена (кроме сегментов `FS` и `GS`);
  - канонические адреса;
  - новый формат таблиц страниц;
  - страницы по 4 кбайт, 2 Мбайт, 1 Гбайт.
- Расширенная кодировка команд:
  - REX-, VEX-префиксы;
  - некоторые кодировки недоступны в 64-разрядном режиме.
- Новые возможности для организации обработки прерываний и системных вызовов.

Регистры общего назначения

Удвоенный размер

- Расширение 32-разрядных РОН 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
Удвоенное количество

- Добавлено восемь новых регистров:
  - `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

- Счётчик команд:
  - `RIP` > `EIP` > `IP`.
- Слово состояния (регистр флагов):
  - `RFLAGS` > `EFLAGS` > `FLAGS`.
- Системные регистры расширены до 64 разрядов:
  - `CR`*n*;
  - `DR`*n*;
  - `GDTR`, `LDTR`, `IDTR`, `TR`;
  - `MSR_`*nnnn*`_`*nnnn*.

Базовые команды

- Примеры команд:
  - `ADD RAX, QWORD [RSI + 8 * R8];`
  - `ADD EAX, DWORD [ESI + 4 * R8W];`
  - `JMP RDI;`
  - `JMP QWORD [RSP].`

- Прямое кодирование 64-разрядной константы возможно только в команде `MOV`.
- Прямая адресация 64-разрядного адреса возможна только в команде `MOV`.
- При выработке 32-разрядного значения в РОН процессор неявно расширяет результат нулём до 64 разрядов.
- Зачем?

Базовые команды

Правило неявного расширения

- Рассмотрим следующее состояние процессора:
  - RAX = 0x0002_0001_8000_2201;
  - RBX = 0x0002_0002_0123_3301.
- Тогда:
  - ADD RBX, RAX:
    - RBX = 0x0004_0003_8123_5502.
  - ADD EBX, EAX:
    - RBX = 0x0000_0000_8123_5502.
  - ADD BX, AX:
    - RBX = 0x0002_0002_0123_5502.
  - ADD BL, AL:
    - RBX = 0x0002_0002_0123_3302.

Базовые команды

Изменения в наборе команд

- Команды пересылки и расширения:
  - `CDQE` — знаковое расширение EAX до RAX;
  - `CQO` — знаковое расширение RAX до пары RDX:RAX;
  - `MOVSXD`— знаковое расширение до 64-разрядного числа.
- Команды работы со стеком:
  - `PUSHA`, `PUSHAD`, `POPA`, `POPAD` — загрузка и сохранение всего регистрового файла на стеке.
  - `PUSHFQ`, `POPFQ` — загрузка и сохранение слова состояния.
- Команды передачи управления:
  - `JRCXZ` — передача управления по условию RCX == 0;
  - `IRETQ` — возврат из прерывания, обслуживаемого в длинном режиме;
  - `SYSCALL`, `SYSRET` — команды «быстрого» обслуживания системных вызовов.
- Прочие команды:
  - `AAA`, `AAD`, `AAM`, `AAS`, `DAA`, `DAS` — работа с BCD-значениями;
  - `BOUND` — проверка принадлежности числа диапазону;
  - `CMPSQ`, `SCASQ`, `MOVSQ`, `LODSQ`, `STOSQ` — строковые команды;
  - `CMPXCHG16B` — 128-разрядный примитив “compare-and-swap.”

Режимы адресации

- Режим адресации — способ кодирования операнда и, как следствие, допустимый вид операнда.
- Различные команды имеют различные режимы адресации операндов.
- Режим адресации может описывать константное значение, регистр или участок памяти.
- Адресация памяти:
  - сегментный регистр;
  - базовый регистр;
  - регистр индекса и множитель;
  - смещение.

Формат команды

- Команда считывается «слева направо» — от младших адресов к старшим и имеет переменный размер от 1 до 15 байтов.
- После чтения очередного байта декодер в процессоре «знает», нужно ли считывать ещё один.
- Примерная последовательность полей:
  - наследственные префиксы (ноль или более);
  - VEX-префикс или REX-префикс (необязательно);
  - 1-, 2-, или 3-байтовый код операции;
  - в зависимости от режимов адресации операндов — байт ModRM, пара байтов ModRM/SIB или ничего;
  - в зависимости от режимов адресации операндов — смещение переменного размера;
  - в зависимости от режимов адресации операндов — значение операндов-констант переменного размера.

REX-префикс

- REX-префикс — это байт со значением от 0x40 до 0x4F.
- Работает только в 64-разрядном режиме, в остальных режимах эти байты кодируют команды `INC` и `DEC`.
- Имеет четыре изменяемых бита:
  - *REX.W* — признак 64-разрядного размера операнда;
  - *REX.R* — дополнительный разряд к полю ModRM *REG*;
  - *REX.X* — дополнительный разряд к полю ModRM *INDEX*;
  - *REX.B* — дополнительный разряд к полю ModRM *R/M*.

Байты ModRM и SIB

- Один или два байта (определяется по байту ModRM), которые в совокупности задают пять полей:
  - *MOD* (2 бита);
  - *R/M* (3 бита);
  - *REG* (3 бита);
  - *INDEX* (3 бита);
  - *SCALE* (2 бита).
- Если при команде есть REX- или VEX-префикс, поля *R/M*, *REG*, *INDEX* расширяются до 4 битов и могут кодировать 16 регистров.
- Байт ModRM может кодировать в качестве базы регистр `rIP`:
  - значение счётчика команд соответствует адресу следующей команды, т.к. фаза выборки уже прошла;
  - пример: `MOV RAX, QWORD [RIP + 0x2019]`;
  - зачем это сделано?

Байты ModRM и SIB

Неочевидные особенности

- Регистры `SPL`, `BPL`, `SIL`, `DIL` имеют те же номера, что и `AH`, `CH`, `DH`, `BH`.
- Если команда имеет REX- или VEX-префикс, они декодируются как `SPL`, `BPL`, `SIL`, `DIL`; в противном случае — как `AH`, `CH`, `DH`, `BH`.
- Это означает, что в команде с REX- или VEX-префиксом нельзя использовать регистры `AH`, `CH`, `DH`, `BH`.
- Примеры:
  - `CMP CL, BYTE [R8]` — корректная команда;
  - `CMP CH, BYTE [R8]` — некорректная команда.

Соглашения о вызовах

- Используется два основных соглашения:
  - System V ABI AMD64;
  - Microsoft ABI.
- Соглашения имеют много общего:
  - они являются разновидностью *fastcall*: часть параметров передаётся на регистрах (— почему *fastcall*?);
  - фрейм, как правило, не оформляется явно (с использованием цепочки RBP).

Соглашение System V ABI

- Вызываемая подпрограмма обязана:
  - сохранить значения RBP, RBX, R12, R13, R14, R15;
  - сохранить управляющие биты MXCSR и FPU;
  - сбросить флаг DF перед выходом.
- Стек выравнивается по границе 16 байт.
- Область размером 128 байт в отрицательную сторону от RSP – «красная зона».
- Аргументы передаются по возможности на регистрах в порядке слева направо, остальные размещаются на стеке:
  - регистры для целых чисел и указателей – RDI, RSI, RDX, RCX, R8, R9;
  - «слоты» аргументов в стеке выровнены по 8 байт.
- Возвращаемые значения (целые числа и указатели) – RAX (и RDX при необходимости).

Соглашение System V ABI

Пример

```
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);
```

- Регистры общего назначения:
  - `RDI = e`, `RSI = f`, `RDX = s.a : s.b`, `RCX = g`, `R8 = h`, `R9 = i`.
- Векторные регистры с плавающей точкой:
  - `XMM0 = s.d`, `XMM1 = m`, `XMM2 = y`, `XMM3 = n`.
- Стек:
  - `[RSP + 8 + 0] = ld`, `[RSP + 8 + 16] = j`, `[RSP + 8 + 24] = k`.