Моделирование центрального процессора с помощью интерпретации

Курс «Программное моделирование вычислительных систем»

Григорий Речистов
grigory.rechistov@phystech.edu

18 февраля 2015 г.

На прошлой лекции

Требования к симуляторам:

- точность,
- скорость,
- совместимость/расширяемость

Вопросы

- В чём измеряется скорость симуляции?

Вопросы

- В чём измеряется скорость симуляции?
- Как соотносятся скорости функционального и потактового симуляторов?

Вопросы

- В чём измеряется скорость симуляции?
- Как соотносятся скорости функционального и потактового симуляторов?
- Для чего необходимо предоставлять API симулятора?

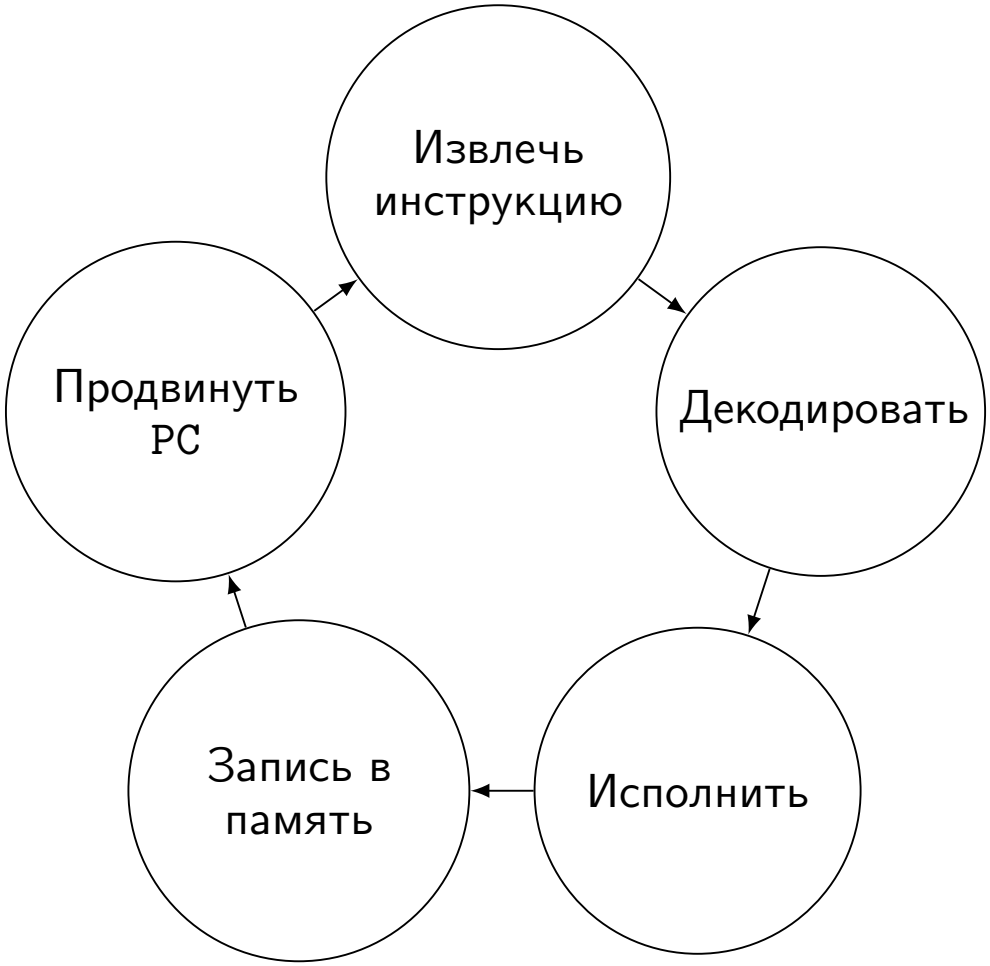
Вопросы

- В чём измеряется скорость симуляции?
- Как соотносятся скорости функционального и потактового симуляторов?
- Для чего необходимо предоставлять API симулятора? Чтобы пользователи могли с ним поиграть.

На этой лекции

1. Конвейер процессора
2. Fetch
3. Decode
4. Execute
5. Write Back
6. Advance PC
7. Исключения

Конвейер процессора



Переключаемый интерпретатор (switched)

```
while (run) {
    raw_code = fetch(PC);
    (opcode, operands) = decode(raw_code);
    switch (opcode) {

    case opcode1:
        func1(operands); PC++; break;

    case opcode2:
        func2(operands); PC++; break;

    /*...*/
    }
}
```

Чтение инструкции из памяти

```
data = mem[pc];
```

Чтение инструкции из памяти

```
data = mem[pc];
```

Скорее,

```
data = read_mem(pc);
```

Порядок байт при доступах

- Порядок от младшего к старшему (*англ.* little-endian);
- Порядок от старшего к младшему (*англ.* big-endian);
- Смешанный порядок (*англ.* middle-endian).

Представление	D4 + C3 * 100 + B2 * 10000 + A1 * 1000000
Little-endian	D4, C3, B2, A1
Big-endian	A1, B2, C3, D4

Бит, байт, слово

- Бит

Бит, байт, слово

- Бит — ?
- Байт

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации
- Октет

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации
- Октет — восемь бит

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации
- Октет — восемь бит
- Машинное слово

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации
- Октет — восемь бит
- Машинное слово — максимальный объём информации, который ЦПУ может обработать единовременно

Бит, байт, слово

- Бит — ?
- Байт — минимальная адресуемая (в данной архитектуре) единица хранения информации
- Октет — восемь бит
- Машинное слово — максимальный объём информации, который ЦПУ может обработать единовременно

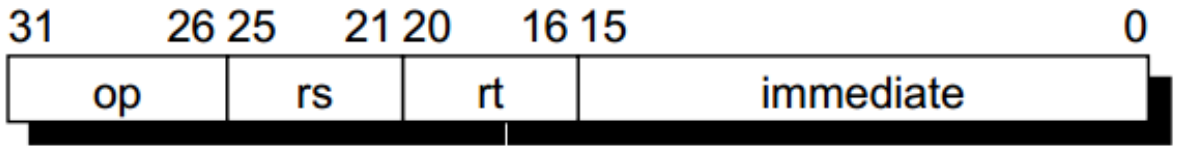
Intel: word — 16 бит, dword — 32 бит, qword — 64 бит.

Декодирование

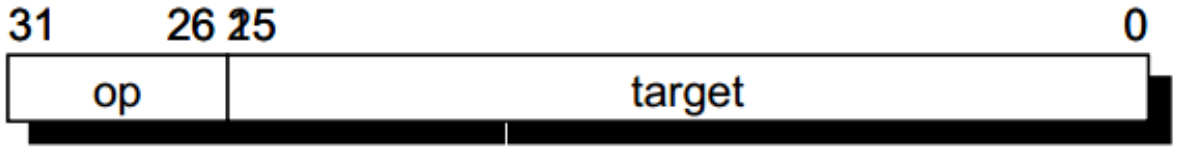
Перевод данных об инструкции из машинного представления во внутреннее (высокоуровневое), удобное для последующего анализа

Пример: MIPS

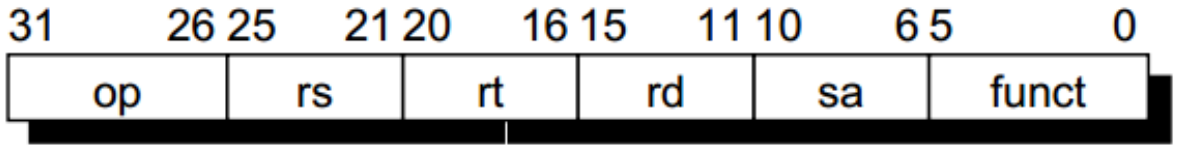
I-Type (Immediate)



J-Type (Jump)



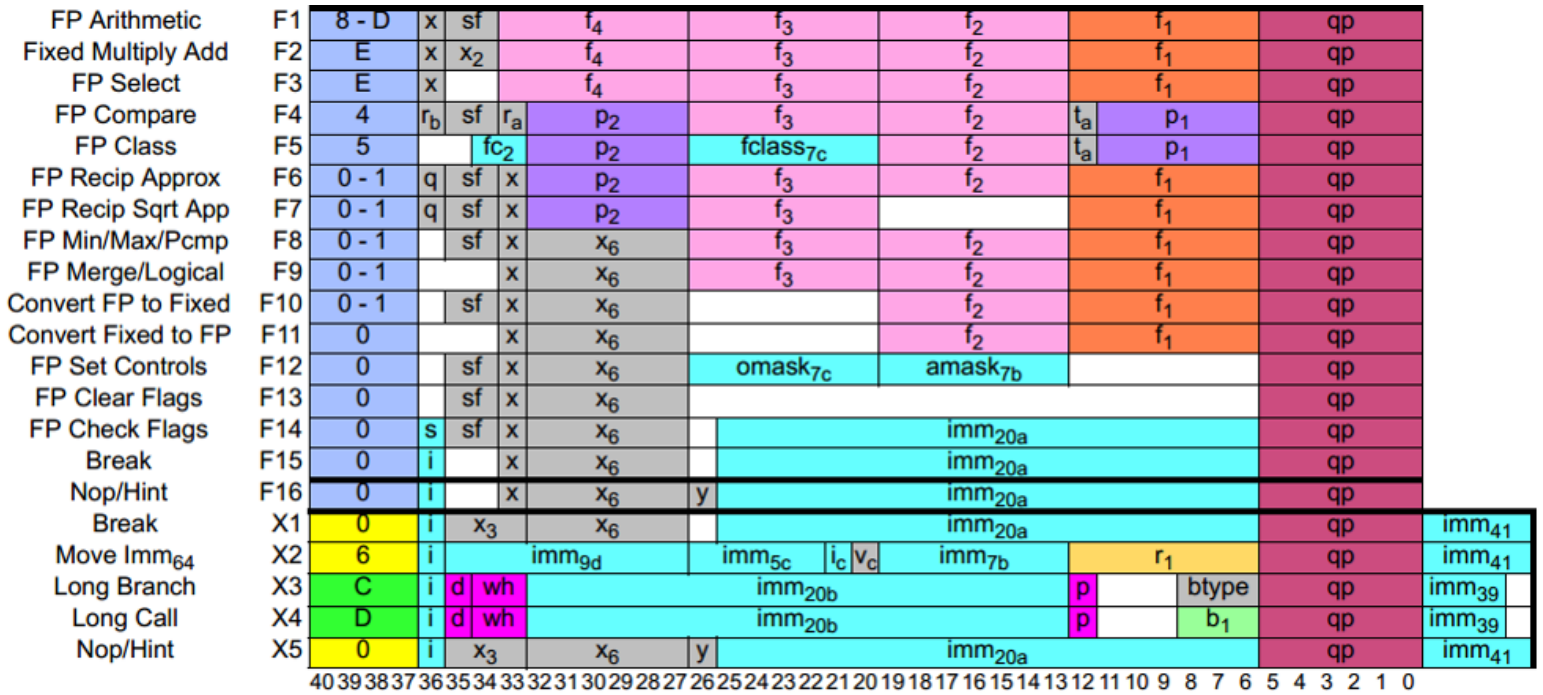
R-Type (Register)



Пример: код

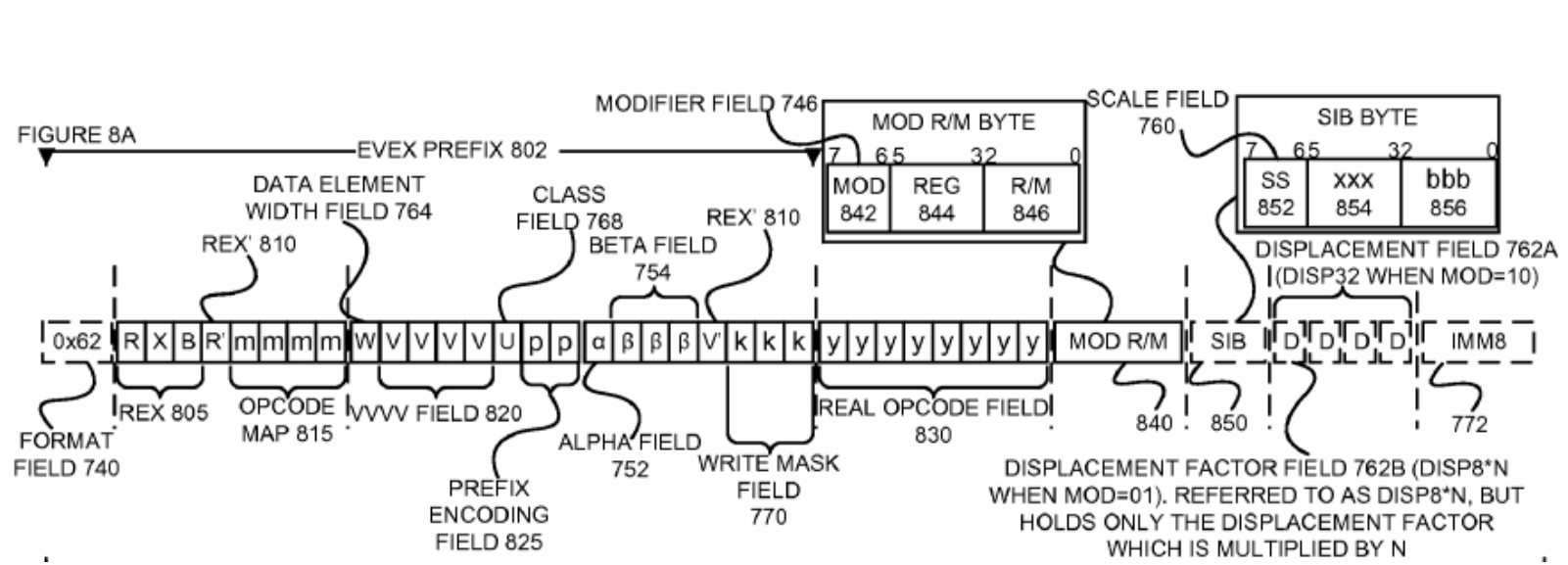
```
#define BITS(v, s, e) (v >> s) & (( 1 << (e-s+1)) - 1)
typedef struct decode {/* ... */} decode_t;
static inline int32_t sign_extend(uint32_t v, int width)
    {/* ... */};
decode_t decode(uint32_t raw) {
    uint32_t op = BITS(raw, 26, 31);
    uint32_t rs = BITS(raw, 21, 25);
    uint32_t rt = BITS(raw, 16, 20);
    int32_t  imm = sign_extend(BITS(raw, 0, 15));
    int32_t  tgt = sign_extend(BITS(raw, 0, 25));
    /* ... */
    uint32_t funct = BITS(raw, 0, 5);
    return decode_t{op, rs, rt, imm, tgt, funct};
}
```

Более сложный пример: Intel IA-64 2.3



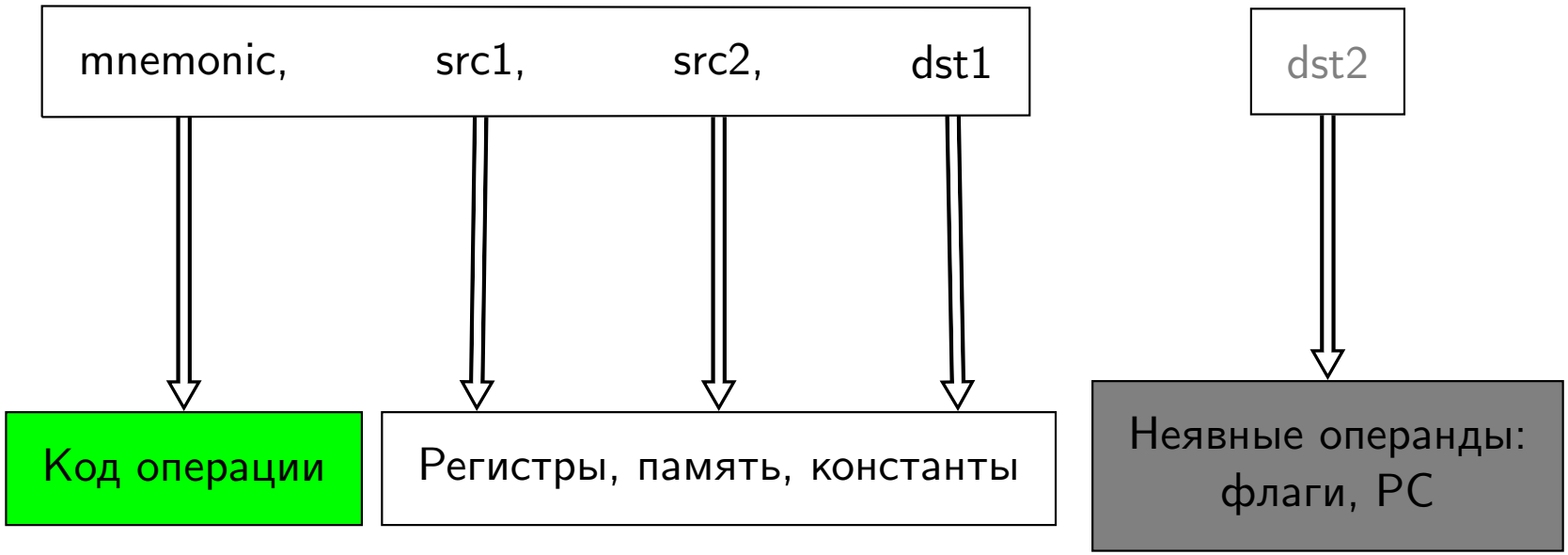
Intel® Itanium® Architecture Software Developer's Manual, p 3:296

Пример посложнее: Intel IA-32



J.C.S. Adrian et al. Systems, Apparatuses, and Methods for Blending Two Source Operands into a Single Destination Using a Writemask. US Patent Application Publication. № 2012/0254588 A1

Что извлекать из машинного кода инструкции



На выходе декодера: успех, неуспех, недостаточно данных.

Декодирование

Код декодера редко пишется вручную, чаще он генерируется по описанию.

A5 YX 0Z 00 $\Rightarrow$ MOD RX, RY, RZ

В общем случае: классическаяя задача построения синтаксического анализатора.

На практике: специализированные инструменты и языки.

Декодирование: суровая реальность

- Переменная длина инструкций. IA-32: от 8 до 120 бит. Сколько байт пытаться декодировать за один раз?
- Зависимость смысла от префикса, режима работы процессора. Пример: 0x48
- Полное несоответствие какому-либо здравому смыслу

Дизассемблирование

Дизассемблирование — перевод инструкций из машинного представление в понятный человеку вид (мнемонику).

Дизассемблирование

Дизассемблирование — перевод инструкций из машинного представление в понятный человеку вид (мнемонику).
(За)кодирование (encoding) — перевод инструкций из мнемонической записи в машинный код.

Исполнение

Базовая единица — функция-эмулятор одной инструкции (service routine).

s.r. пишутся на языке высокого уровня — переносимость кода между хозяйскими платформами, компиляторами.

Используются генераторы кода.
Пример: SimGen — из одного описания генерируются декодер, дизассемблер и s.r.

Пример: ADD

```
void add32_rr(cpu_t *cpu, int src1, int src2, int dst) {
    cpu->regs[dst] = cpu->regs[src1] + cpu->regs[src2];
```

Пример: ADD

```
void add32_rr(cpu_t *cpu, int src1, int src2, int dst) {
    cpu->regs[dst] = cpu->regs[src1] + cpu->regs[src2];

    cpu->z_flag = cpu->regs[dst] == 0;
    cpu->n_flag = cpu->regs[dst] & (1 << 31);
    cpu->o_flag = cpu->regs[dst] <
            MAX(cpu->regs[src1], cpu->regs[src2]);
    cpu->c_flag = calc_c_flag(cpu->regs[src1],
                              cpu->regs[src2]);
}
```

Пример посложнее

Запись результата в память

<<Обычная>> запись в память:

Запись результата в память

<<Обычная>> запись в память:

- Невыровненные адрес,
- Граница страниц,
- Попытка изменить регион памяти доступный только для чтения,
- Часть результата может быть записана, а потом случится исключение.

Продвижение $PC

- Для большинства команд увеличение счетчика на длину обработанной инструкции.
  Ислючение: `REP MOVS`.

Продвижение $PC

- Для большинства команд увеличение счетчика на длину обработанной инструкции.
  Ислючение: `REP MOVS`.

- Явное изменение `$PC` — команды управления исполнением:
  - (Un)conditional (In)direct Jump/Branch,
  - Call/Return (subroutine).

Уточнённый цикл работы процессора

Классификация

Interruptions (термин из документации IA-64) — вмешательство, перерыв, приостановка Exception — синхронное исключение, без повторения текущей инструкции Fault — синхронное, с повторением текущей инструкции Trap — синхронные, общий случай для некоторой инструкции Interrupt — внешнее асинхронное прерывание Abort — внешние асинхронное с отсутствием информации о точке возврата

На следующей лекции

Спасибо за внимание!

Замечание: все торговые марки и логотипы, использованные в данном материале, являются собственностью их владельцев. Представленная здесь точка зрения отражает личное мнение автора, не выступающего от лица какой-либо организации.