Архитектура вычислительных систем

Семинары № 8

Математический сопроцессор

План семинарского занятия

Цель и задачи

Познакомиться со способом хранение вещественных чисел в памяти. Рассмотреть основные задачи и возможности сопроцессора (FPU).

Основные вопросы

1. Хранение вещественных чисел в памяти по IEEE 754.
2. Инструкции FPU-сопроцессора.
3. Условные операторы и управляющий регистр fcsr

Вещественные числа и их запись

Несколько слов о представлении вещественных чисел

Числа с "фиксированной точкой"

Мы имеем дело **не с реальными вещественными числами**, а с их (рациональным) представлением.

$1234.5 = 1.2345*10^{3}$ $0.012345 = 1.2345*10^{-2}$

Таким образом вещественное число состоит из двух частей: **мантиссы** (1.2345) и показателя степени — **порядка.**

155.625:

- ✓ *155.625E0*
- ✓ *1.55625E2*
- ✓ *0.001555625E5*
- ✓ *155625E-3*

IEEE 754

155.625 в виде двоичного числа с плавающей точкой

= 10011011,101

в экспоненциальном виде: **$1{,}55625 \cdot exp_{10}^{+2} = 1{,}0011011101 \cdot exp_2^{+111}$**

Мантисса 1.0011011101, а порядок — $exp_2 = {}^{+111}$

Согласно стандарту IEEE 754

```
S[  E   ][          M              ]
01110101111100010110101110101111
```

S - бит знака

E - смещенный порядок двоичного числа; для 32-битного представления — 8 битов

IEEE 754

В 32 битном представлении **на мантиссу** приходится **23 бита.**

Низкая точность: 2^{23}=8388608

В IEEE 754 используется хитрость, позволяющая сэкономить ещё один бит мантиссы:

Число	1 бит	8 бит	23 бит	Шестнадцатеричное
	знак числа	смещённый порядок	мантисса	
155.625	0	10000110	00110111010000000000000	431BA000
-5.23E-39	1	00000000	01110001111001100011101	8038f31d

24 бита на мантиссу — это очень мало.

Например десять миллиардов представлены с точностью +/- полтысячи.

NaN

0 11111111 X *********************

- x=0 — NaN «по-тихому», для вычислений, возвращающих NaN
- x=1 — «сигнальный» NaN, для исключений
- RARS: ± 11111111 0 00000000000000000000 — ±∞

Инструмент для изучения числа в формате IEEE 754

Tools → Floating Point Representation

0 00000000 0 00000000000000000000 — это просто +0

Сопроцессоры

FPU

Сопроцессоры

Различные, почти не пересекающиеся задачи ⇒ сопроцессоры:

Сопроцессор - FPU

- Другая математика
- Свои регистры
- Почти не смешивается с целочисленной арифметикой
- «Тяжёлые» вычисления

Сопроцессор 0 – управления

Макросы с локальными метками

1. Расширение F поддерживает числа с плавающей точкой одинарной и двойной точности в IEEE-формате.
2. Архитектура предусматривает наличие тридцати двух регистров для чисел с плавающей точкой "f0–f31".
3. Если в архитектуре поддерживается "D" или "Q", для представления в этих регистрах чисел меньшей разрядности используется т. н. «NaN boxing» — заполнение старших частей регистра специальной константой NaN — «Not A Number»

Общее название	Мнемоника ABI	Назначение	Переживают ли вызов подпрограммы?
f0 - f7	ft0 - ft7	Временные	Нет
f8 - f9	fs0 - fs1	Сохраняемые при вызове	Да
f10 - f17	fa0 - fa7	Параметры и возвращаемые значения	Нет
f18 - f27	fs2 - fs11	Сохраняемые при вызове	Да
f28 - f31	ft8 - ft11	Временные	Нет

Инструкции FPU-сопроцессора

- **Регистры f* не имеют отношения к регистрам x***,
  но количество их такое же, и значит, они могут встречаться
  в командах типа **S** (запись в память, fs*),
  I (чтение из памяти, fl*) и **R** (вычисления).
- ***Процессор* ничего не знает о формате IEEE754**, так что в командах работы с памятью используется терминология «word» / «double word»: **flw, fsw, fld, fsd**
- Точность арифметических команд *сопроцессора* указывается суффиксом инструкции (**s**, **d**, q, h)
- Арифметические команды: fCMD.P,
  CMD — мнемоника инструкции, P — точность
  - CMD: add, sub, mul, div, sqrt, min, max

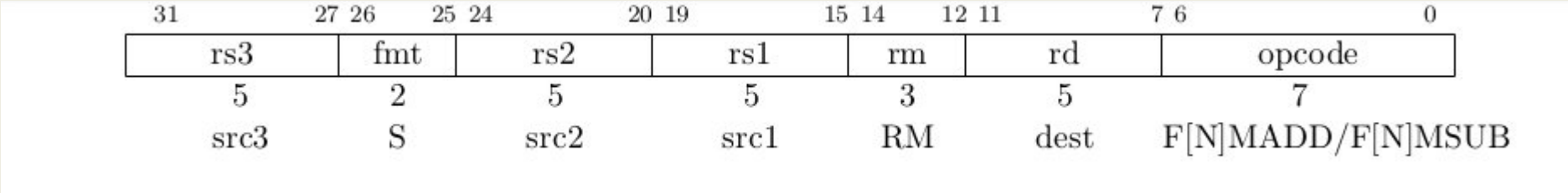
Псевдо-инструкция flw

```
1  .data
2  a:        .float  123.456
3  b:        .float  654.321
4  _2:       .float  2
5  .text
6            flw     ft0 a t0
7            flw     ft1 b t0
8            flw     ft2 _2 t0
9            fadd.s  ft3 ft2 ft1
10           fdiv.s  fa0 ft3 ft2
11           li      a7 2
12           ecall
```

Что это за t0
(регистр общего назначения)
в псевдо-инструкции flw?

Четырёхместные команды «умносложения»

Используются, например, для подсчёта многочленов в форме a0 + a1(x + a2(x + …))



f[n]madd / f[n]msub по формуле a*b+c: **новый тип команд R4**:

Можно обмениваться с регистрами общего назначения

- В мнемонике используются два суффикса
- **Перемещать** машинное слово из одного регистра в другой **умеет центральный процессор**: fmv.s.x и fmv.x.s
  - *Не преобразованное целое, лежащее в вещественном регистре и вещественное, лежащее в целом, нельзя осмысленно обработать, но можно зачем-то там хранить*
- **Преобразовывать** из вещественного формата в целый и обратно (а также из двойного в одинарный и обратно) **умеет только FPU**: fcvt.d.s fcvt.s.d, fcvt.P.w[u] и fcvt.w[u].P

Пример: $(x - 1)^2$

```
1  .data
2  x:      .double 12.34
3  .text
4          li              t2 2
5          fcvt.d.w        ft2 t2                  # ft2 = 2.0
6          li              t1 1
7          fcvt.d.w        ft1 t1                  # ft1 = 1.0
8          fld             ft0 x t0                # ft0 = x
9          fnmsub.d        ft3 ft2 ft0 ft1         # ft3 = -(2 * x) + 1
10         fmadd.d         fa0 ft0 ft0 ft3         # fa0 = x * x + ft3
11         li              a7 3                    # Вывод числа двойной точности
12         ecall
```

В псевдоинструкции **fcvt** присутствует таинственный параметр **dyn** — это константа, означающая, что используется **стратегия округления** по умолчанию

Перемещение между f-регистрами

fmv f1 f2 — в действительности псевдоинструкция