FPU. Конвенции вызова.

Иван Викторович Михайлов

ИТМО, КТ

imihajlow@gmail.com

11.02.2015

Еще инструкции базового набора.

CMOVcc

```
CMOVcc dst, src
```

Условная пересылка.

Пример:

```
cmp eax, ecx
cmovg eax, ecx ; eax = min(eax, ecx)
```

LEA

```
LEA regd, mem
```

Load Effective Address.

Примеры:

```
lea eax, [eax + eax*4] ; eax *= 5
lea eax, [ebx + ecx + 2] ; сложение двух регистров и константы
                         ; одной инструкцией
```

Инструкции с флагами

LAHF

AH $\leftarrow$ EFLAGS(SF:ZF:0:AF:0:PF:1:CF)

SAHF

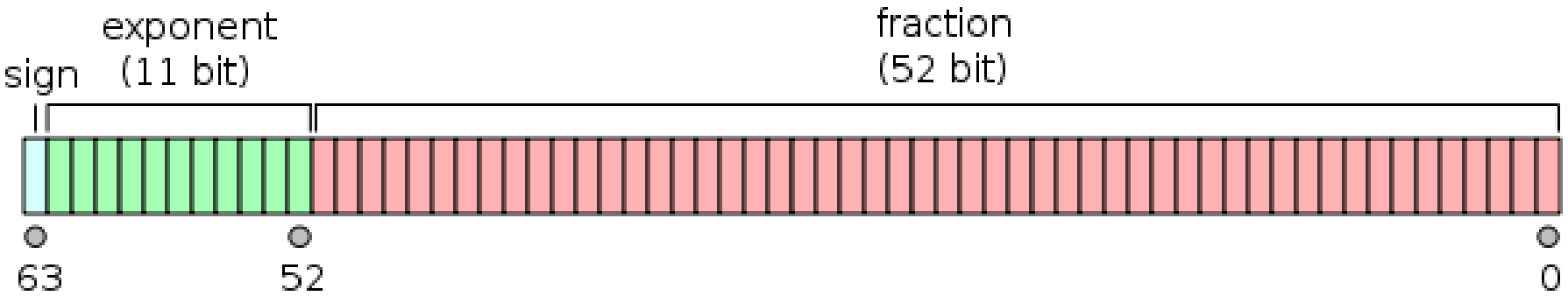
EFLAGS(SF:ZF:0:AF:0:PF:1:CF) $\leftarrow$ AH

STD/CLD

Установить и сбросить флаг направления (DF).

FPU. Типы данных.

Double



$$x = (-1)^{sign}(1 + \frac{fraction}{2^{52}}) \times 2^{exponent-1023}$$

Single

- 32 бита;
- Экспонента 8 бит;
- Мантисса 23 бита.

Extended 80-bit

- 80 бит;
- Экспонента 15 бит;
- Мантисса 63 бита;
- Бит 63 – целая часть (слева от точки).

Другие форматы

Целое

16, 32, 64 бита со знаком.

BCD

Двоично-десятичное число. 80 бит, 18 цифр по 4 бита (биты 0-71), знак (бит 79).

FPU. Специальные числа.

Ноль

- Целое – просто ноль.
- Дробное, все нули в экспоненте:
  - $+0$, 0 (просто ноль) – все биты 0;
  - -0 (отрицательный ноль) – бит знака 1, остальные 0;
  - Денормальные/субнормальные числа (denormal/subnormal) – экспонента ноль, мантисса не ноль, бит целого – 0.

Бесконечность

- Экспонента – все единицы;
- Бит целого (для 80-битного числа) – 1;
- Мантисса – все нули;
- Знак – знак бесконечности.

Неопределенности

Дробные:

- Экспонента – все единицы;
- Бит целого (для 80-битного числа) – 1;
- Мантисса – хотя бы один ненулевой бит:
  - 0X...XX – SNaN (signalling);
  - 1X...XX – QNaN (quiet).

Целое NaN – 10...00.

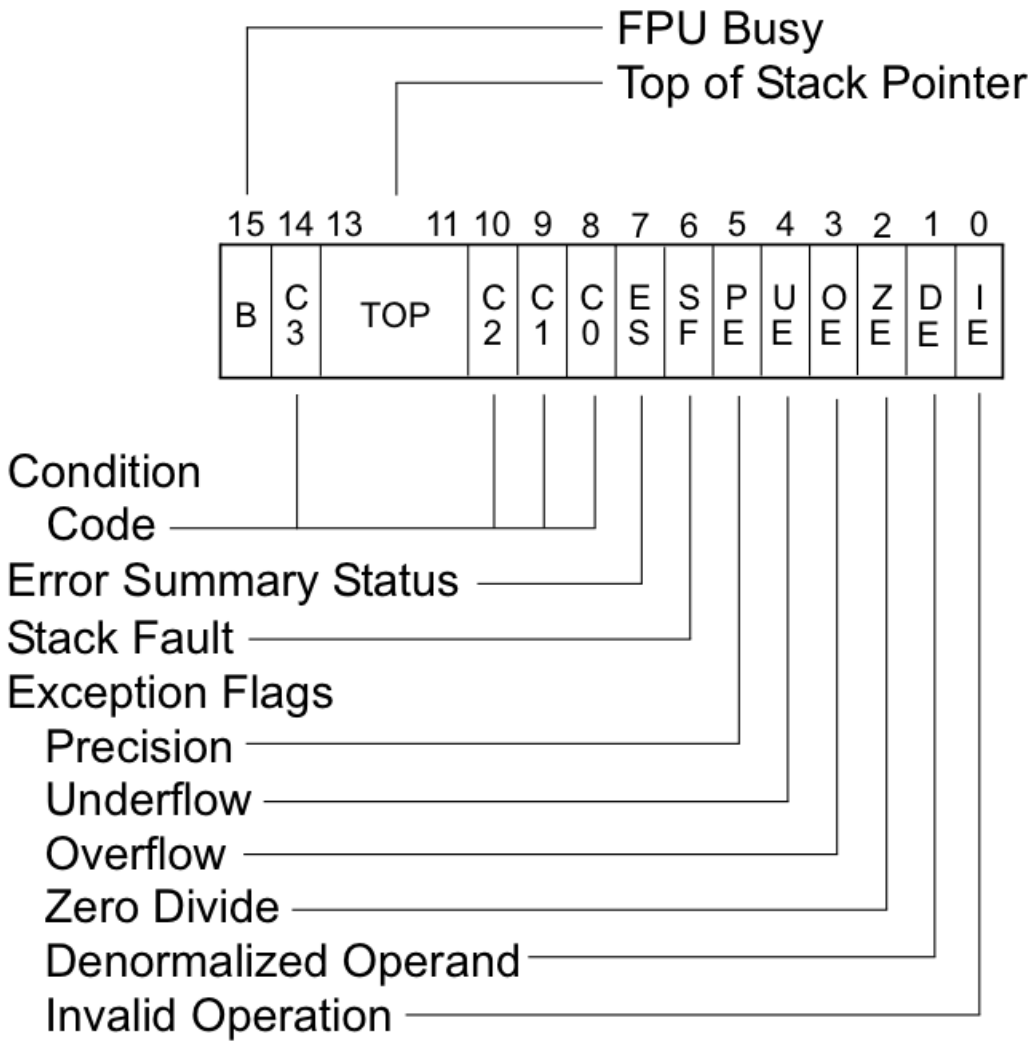
FPU. Регистры.

Регистры данных

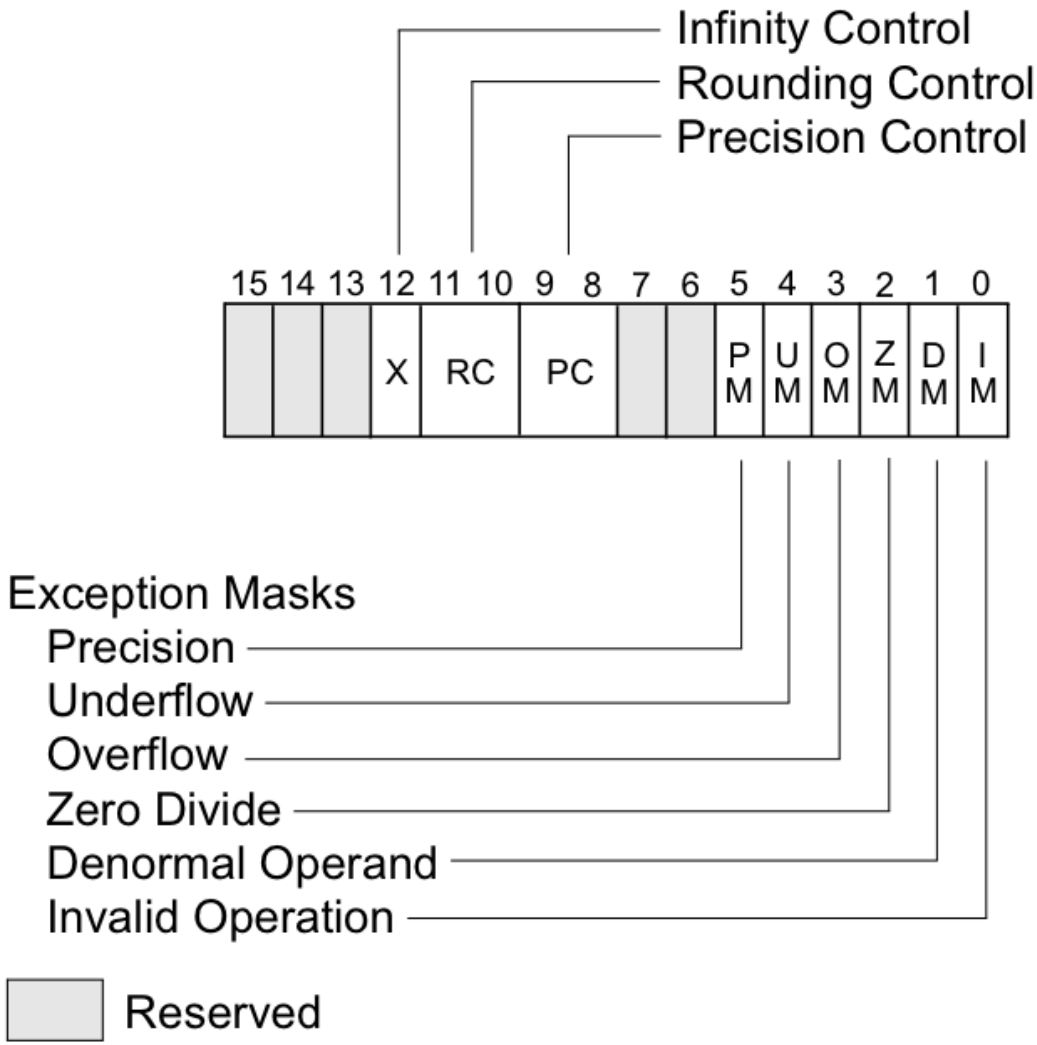
- 80 бит,
- 8 штук,
- организованы в стек.

`ST0` – вершина стека,
`ST1` – следующий после вершины,
...

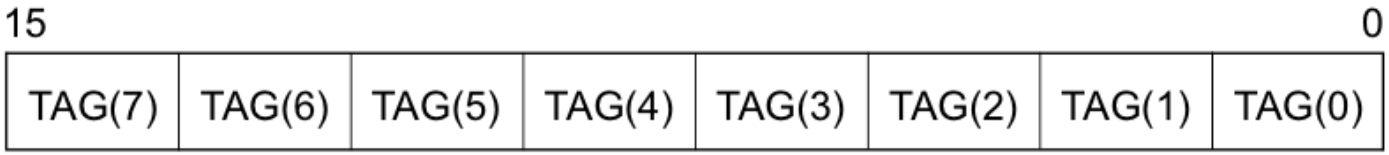
Регистр статуса



Регистр управления



Тэг



TAG Values

00 — Valid
01 — Zero
10 — Special: invalid (NaN, unsupported), infinity, or denormal
11 — Empty

FPU. Инструкции.

Передача данных

FLD/FILD/FBLD

`FLD mem/st`i – загрузить дробное число на стек,

`FILD mem` – загрузить целое,

`FBLD mem` – загрузить двоично-десятичное.

FST/FSTP/FIST/FISTP/FBSTP

`F*ST mem` – сохранить st0 в память,

`F*STP mem` – сохранить st0 в память и вынуть из стека.

Передача данных

FXCH

`FXCH` – поменять st0 и st1,

`FXCH st`i – поменять st0 и st$i$.

FCMOVcc

`FCMOVcc st0, st`i

cc = B (below), E (equal), U (unordered), BE, NB, NE, NU, NBE.

Константы

Загрузка на стек констант:

`FLD1`	$+1.0$
`FLDL2T`	$log_2 10$
`FLDL2E`	$log_2 e$
`FLDPI`	π
`FLDLG2`	$log_{10} 2$
`FLDLN2`	$ln 2$
`FLDZ`	$+0.0$

Арифметика

`Fop src`	st0 = st0 op src
`Fop st`i`, st0`	st$i$ = st$i$ op st0
`FopR src`	st0 = src op st0
`FopR st`i`, st0`	st$i$ = st0 op st$i$
`FIop mem`	операция с целым
`FIopR mem`	операция с целым, обратный порядок

op = ADD, SUB, MUL, DIV

Арифметика

`FABS`	$st0 = \lvert st0 \rvert$
`FCHS`	$st0 = -1 \times st0$
`FSQRT`	$st0 = \sqrt{st0}$
`FRNDINT`	$st0 = [st0]$
`FPREM, FPREM1`	остаток от деления
`FXTRACT`	выделить экспоненту и мантиссу

Сравнение

`F(U)COM(I)(P|PP) (src)`

- U – разрешает QNaN;
- I – устанавливает EFLAGS (обычно флаги FPU);
- P – вынуть из стека одно число;
- PP – вынуть из стека два числа.

	C3/ZF	C2/PF	C0/CF
$ST0 > src$	0	0	0
$ST0 < src$	0	0	1
$ST0 = src$	1	0	0
Unordered	1	1	1

FTST – сравнение st0 с нулем.
FXAM – классификация st0.

Управляющие инструкции

- `FWAIT` – ожидание завершения операции;
- `FINIT/FNINIT` – сброс FPU;
- `FSTSW/FNSTSW AX/mem16` – сохранение статуса FPU;
- ...

Классическое сравнение

```
fcom              ; Сравнить st0 и st1
fstsw ax          ; Сохранить состояние в AX
fwait             ; Подождать завершения операции
sahf              ; Помесить AH в флаги
jpe unordered     ; pf = 1
ja  st0_greater   ; cf = 0 && zf = 0
jb  st0_lower     ; cf = 1
jz  both_equal    ; zf = 1
```

Тригонометрия

- `FSIN`;
- `FCOS`;
- `FSINCOS`;
- `FPTAN`;
- `FPATAN`.

$st0 = cos, st1 = sin$

Логарифмы и экспоненты

- `FYL2X` – $st1 \cdot log_2 st0$;
- `FYL2XP1` – $st1 \cdot log_2(st0 + 1), -\delta < st0 < \delta, \delta = 1 - \sqrt{2}/2$;
- `FSCALE` – $st0 \leftarrow st0 \cdot 2^{\lfloor st1 \rfloor}$;
- `F2XM1` – $st0 \leftarrow 2^{st0}, -1 < st0 < 1$;

$\lfloor st1 \rfloor$ – округление в сторону нуля.

Конвенции вызова.

Вызов функций

- Где передавать параметры?
- Как возвращать результат?
- Какие регистры сохранять, а какие нет?

Стек

Параметров может быть много – надо использовать стек.

- В каком порядке класть параметры?
- Кто очищает стек?

Конвенции вызовов

- cdecl
- stdcall
- fastcall
- thiscall
- pascal

Порядок параметров

Справа-налево:

```
int foo(int a, int b, int c)
```

```
push c
push b
push a
call foo
```

pascal – наоборот.

Сохраняемые регистры

Вызываемый сохраняет:
EBX, ESI, EDI, EBP

Возвращаемое значение

EAX – 32 бита,
EDX:EAX – 64 бита (double, long long)

Очистка стека

- cdecl – вызывающий;
- остальные – вызываемый.

`ret` n

Параметры в регистрах

- GNU fastcall, Microsoft fastcall – ECX, EDX;
- Borland fastcall – EAX, ECX, EDX;
- Microsoft thiscall – this в ECX.

Пример

```
int __attribute__((fastcall))
foo(double a, int b, double c, int d, int e)

push dword [e]
mov edx, [d]
push qword [c]
mov ecx, [b]
push qword [a]
call foo
```

Практика.

Практика

https://github.com/itmoasm2015/practice

```
git clone https://github.com/itmoasm2015/practice.git
```

Что почитать

- Intel® 64 and IA-32 Architectures Software Developer's Manual, Vol. 1, Chapter 8
- Intel® 64 and IA-32 Architectures Software Developer's Manual, Vol. 2
- http://www.agner.org/optimize/calling_conventions.pdf

Конец.