Архитектура ЭВМ

Лекция 7. Математический сопроцессор

к.ф.-м.н. Филонов Павел Владимирович
filonovpv@gmail.com

Московский Государственный Технический Университет
Гражданской Авиации

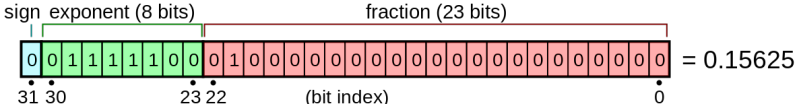
Содержание

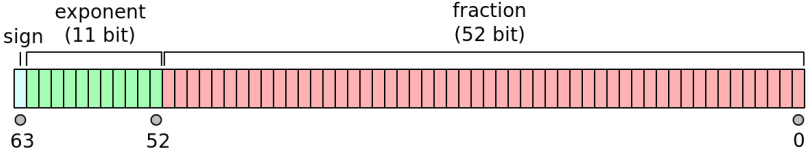
- Числа с плавающей точкой;
- Регистры сопроцессора;
- Операции:
  - Обменн данными;
  - Арифметические;
  - Вычисление базовых функций;
- Польская инверсная запись;
- Работа с флагами сопроцессора.

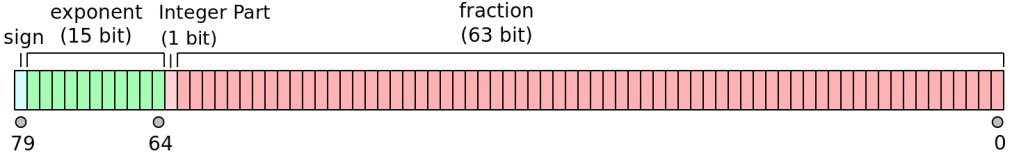
Числа с плвающей точкой

Стандарт IEEE 754 описывает 3 формата кодирования числе с плавающей точкой:

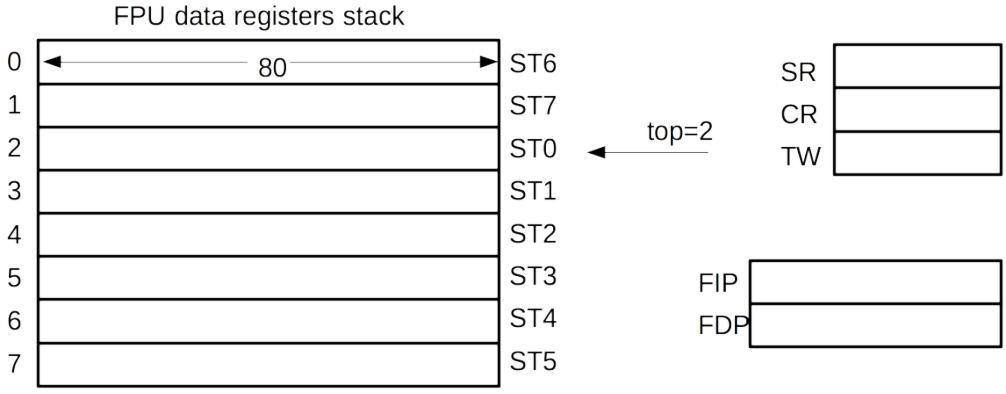
- одинарной точности — 4 байта;
- двойной точности — 8 байт;
- расширенной точности — 10 байт.





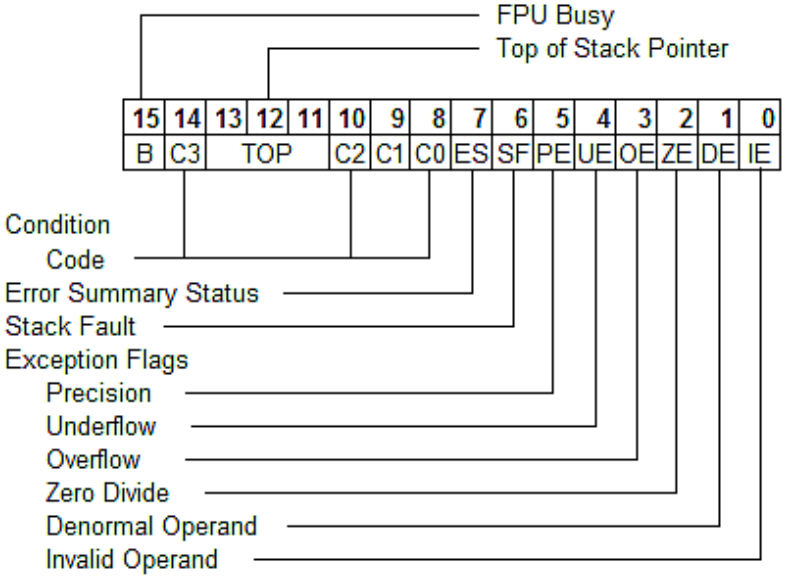


Регистры сопроцессора



- State Register (SR) — регистр состояния
- Control Register (CR) — регистр управления
- TW — регистр тегов
- FIP — хранит адрес последней выполненной сопроцессором команды
- FDP — хранит аргумент последней выполненной сопроцессором команды

Регистр состояния сопроцессора



Обмен данными с сопроцессором

- `fld mem32/mem64/mem80/stx` — загрузить значение на вершину стека;
- `fst mem32/mem64/stx` — сохранить значение с вершины стека в память или другой st регистр;
- `fstp mem32/mem64/mem80/stx` — тоже, что и `fst`, плюс выталкивает вершину стека;
- `fild mem16/mem32/mem64` — загрузить в стек целое число с конвертацией в число с плавающей точкой;
- `fist mem16/mem32` — сохранить число с вершины стека в целочисленном формате;
- `fistp mem16/mem32/mem64` — тоже, что и `fist`, плюс выталкиваем вершину стека;
- `fxch` — поменять местами `st0` и `st1`;
- `fxch stx` — поменять местами `st0` и `stx`.

Загрузка математических констант

Все команды загружают значение константы в регистр `st0`.

- `fld1` — 1.0;
- `fldz` — $+0.0$;
- `fldpi` — π;
- `fld2e` — $\log_2 e$;
- `fld2t` — $\log_2 10$;
- `fldln2` — $\ln 2$;
- `fldlg2` — $\lg 2$.

Пример: число π

```
1  print_pi:
2      push ebp
3      mov ebp, esp
4      sub esp, 4
5
6      fldpi
7      fstp dword [ebp - 4]
8      push dword [ebp - 4]
9      call print_float
10     add esp, 4
11
12     mov esp, ebp
13     pop ebp
14     ret
```

Арифметические операции

- `fadd mem32/mem64/stx` — прибавляет операнд к `st0`
- `fsub mem32/mem64/stx` — вычитает операнд из `st0`
- `fsubr mem32/mem64/stx` — вычитает `st0` из операнда и сохраняет в `st0`
- `fmul mem32/mem64/stx` — умножает операнд на `st0`
- `fdiv mem32/mem64/stx` — делит `st0` на свой операнд
- `fdivr mem32/mem64/stx` — делит свой операнд на `st0` и сохраняет в `st0`

Варианты над регистрами: `fxxx st0, stx` или `fxxx stx, st0`.
Выталкиваюшие варианты: `faddp, fsubp, fsubrp, fmulp, fdivp, fdivrp`.
Работают только над регистрами `fxxxp stx`
Операнд — целое число: `fiadd, disub, fisubr, fimul, fidiv, fidivr`

Польская инверсная запись

Любому арифметическому выражению можно поставить в соответсвие инверсную запись, которая не будет содержать скобки.
Примеры:

- $4 + 2 \rightarrow 4\ 2+$
- $3 * (4 + 2) \rightarrow 4\ 2 + 3*$
- $(1 - 2) * (4 + 5) \rightarrow 1\ 2 - 4\ 5 + *$

Алгоритм вычисления выражения в инверсной записи

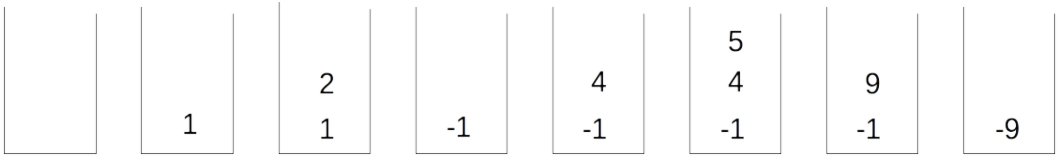
```
1  for c ← каждого элемента выражения
2        do if c — число
3              then push c
4              else  if c — операция
5                        then pop a
6                             pop b
7                             push применить c к a и b
8  pop result
9  return result
```

Пример: арифметика на стеке

- Переведем выражение в польскую инверсную запись

$$(1 - 2) * (4 + 5) \rightarrow 1\ 2 - 4\ 5 + *$$

- Используем для вычисления выражения стек



Арифметика на стеке

$$1\ 2 - 4\ 5 + *$$

```
1       fild dword [one]
2       fild dword [two]
3       fsubp st1
4       fld dword [four]
5       fld dword [five]
6       faddp st1
7       fmulp st1
8       fstp dword [result]
9       push dword [result]
10      call printfloat
11      add esp, 4
```

Сравнение и обработка результатов

Для сравнения числе с плавающей точкой испоьзуются следующие команды:

- `fcom mem32/mem64/stn` — сравнивает `st0` с операндом;
- `fcomp mem32/mem64/stn` — тоже, что и `fcmo`, плюс выталкивает `st0`;
- `fcompp` — сравнивает `st0` с `st1` и выталкивает оба регистра из стека.

В результате работы любой из этих операций устанавливаются флаги регистра `SR`. Проанализировать значение этих флагов можно скопировав их сначала в регистр `ax`, а затем в `FLAGS`.

```
1       fstsw ax      ; сохранить SW в AX
2       sahf          ; скопировать некоторые флаги из AH в FLAGS
```

После этой процедуры в регистр `FLAGS` будут скопированы флаги `ZF` и `CF`. Следовательно необходимо использовать команды услвных переходов для беззнаковых чисел (`ja, jb, jae, jbe, je, jne`).

Пример: max

Напишем подпрограмму, которая возвращает максимальное из двух чисел с плавающей точкой.

```
1  max:
2      push ebp
3      mov ebp, esp
4      sub esp, 116
5      pushfd
6      fsave [ebp - 116]
7      finit
8
9      fld dword [ebp + 8]
10     fld dword [ebp + 12]
11     fcompp
12     fstsw ax
13     sahf
14     ja .b_gt_a
15     mov eax, [ebp + 8]
16     jmp .end
17
18 .b_gt_a:
19     mov eax, [ebp + 12]
20
21 .end:
22     frstor [ebp - 116]
23     popfd
24     mov esp, ebp
25     pop ebp
26     ret
```

Управление сопроцессором

Для управления сопроцессором используется регистр CR.

- `fstcw mem16` — сохранить содержимое CR в память
- `fldcw mem16` — загрузить значения для CR из памяти

Пример: выставить режим округления «в сторону нуля».

```
1   sub esp, 2          ; выделить 2 байта на стеке
2   fstcw [esp]         ; сохранить в эту область памяти значение CR
3   or word [esp], 0000110000000000b
4                       ; выставить биты 11 и 10 (режим округления)
5   fldcw [esp]         ; загрузить новое значение CR
6   add esp, 2          ; освобождаем память
```

Резюме

- форматы кодирования числе с плавающей точкой
  - одинарной точности
  - двойной
  - расширенной
- регистры сопроцессора организованы в виде стека
- фрифметика на стеке
- операции сравнения чисел с плавающей точкой
- управление сопроцессором осуществляется через регистр `CR`