Арифметический сопроцессор FPU

(продолжение)

Вопросы

- Пересылка данных
- Арифметические операции
- Сравнение чисел

Загрузка данных

команда	src
FLD src	веществ. переменная (32, 64, 80 бит)
FILD src	целая переменная со знаком (16, 32, 64 бит)
FBLD src	BCD переменная 80 бит

- ld – от load
- Действия:

  STP := STP – 1

  ST(0) := src

Сохранение данных

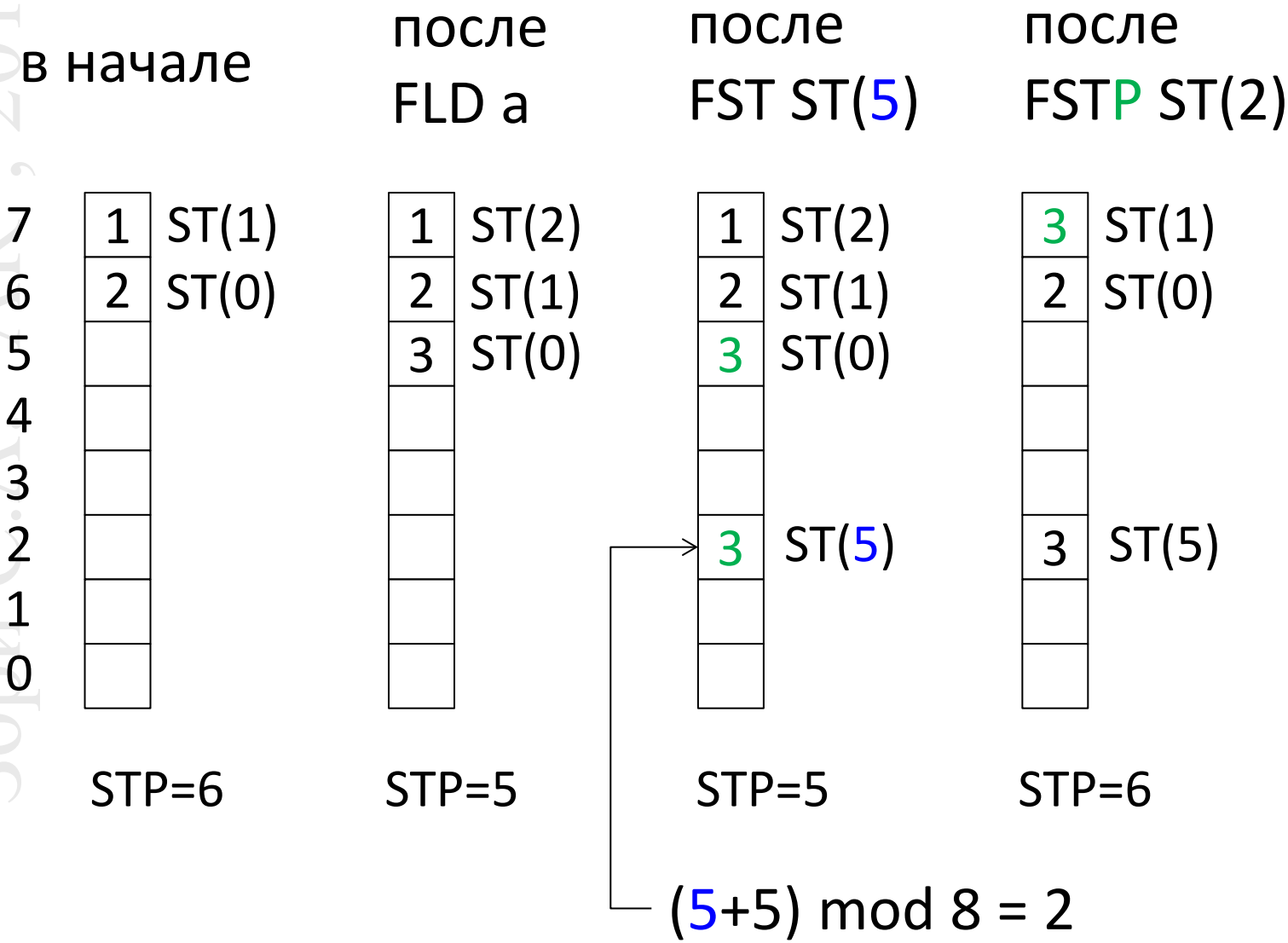
команда	dst
FST dst	dst := ST(0) вещественная переменная (32, 64 бит)
FSTP dst	то же + вещественное 80 бит
FIST dst	целая переменная (16, 32 бит)
FISTP dst	то же + целое 64 бит
FBSTP dst	BCD переменная 80 бит

- st – от store

Сохранение данных

- Эти команды пересылают данные из верхушки стека в **область памяти (переменную)**.
- При этом содержимое указателя стека (поля ST) не изменяется.
- **FST / FSTP** могут записывать также **в любой регистр** (пустой или занятый) **FPU** (использовать ссылку на численный регистр **ST(i)**), поэтому можно использовать эту команду для копирования верхушки стека в любой другой численный регистр FPU.
- При этом пустой регистр помечается занятым

Пересылка данных



Загрузка констант

- Добавляют константу в вершину стека

команда	константа
FLD1	1
FLDZ	+0
FLDPI	π
FLDL2E	$log_2(e)$
FLDL2T	$log_2(10)$
FLDLN2	$ln(2)$
FLDLG2	$log_{10}(2)$

- эффективнее, чем загрузка из памяти

Обмен местами

- **FXCH** ST(n) – обменять ST и ST(n)
- **FXCH** – обменять ST и ST(1)

Управление стеком

команда	действие	
FINCSTP	STP := (STP + 1) по модулю 8 *(стек уменьшается)*	не меняют регистр **TW**
FDECSTP	STP := (STP – 1 + 8) по модулю 8 *(стек растет)*	
FFREE ST(n)	пометить ST(n) как пустой	

Управление стеком

- Удаление элемента из вершины стека:

```
FFREE st(0)
FINCSTP
```

- Пример без FFREE:

```
          ; в стеке a
FLD b     ; a, b
FINCSTP   ; a
          ; ST(7) - бывший ST, помечен занятым
FLD b     ; ошибка стека!
```

Арифметические операции

- **Базовые**:

команда	действие
FADD	+
FSUB	-
FMUL	*
FDIV	/

Арифметические операции

Пусть ● – код операции (ADD, …).

Она выполняет действие dst := dst ● src

Варианты команд:

команда	dst	src
F● src	ST	вещественная переменная 32/64 бит
F● ST, ST(n)	ST	ST(n)
F● ST(n), ST	ST(n)	ST
F●P ST(n), ST	ST(n)	ST
F●P, **F●**	ST(1)	ST
FI● src	ST	целая переменная 16/32 бит

Арифметические операции

Эти 3 команды эквивалентны:

F●

F●P

F●P st(1), st(0)

F● извлекает из стека, хотя без суффикса **P**!

Для наглядности ее лучше не использовать!

Арифметические операции

Команды

FSUB**R**, FSUB**R**P, FISUB**R**, FDIV**R**, FDIV**R**P, FIDIV**R**

аналогичны командам

FSUB, FSUBP, FISUB, FDIV, FDIVP, FIDIV

за исключением:

они выполняют действие dst := src ● dst

Пример

- Вычислить: b / (a * d + 1) + a
- ОПН: b a d * 1 + / a +

```
FLD   b    ; b
FLD   a    ; b, a
FLD   d    ; b, a, d
FMULP      ; b, a*d
FLD1       ; b, a*d, 1
FADDP      ; b, a*d+1
FDIVP      ; b/(a*d+1)
FLD   a    ; b/(a*d+1), a
FADDP      ; b/(a*d+1)+a
```

9 команд

Пример

- Используем аргументы в памяти:

```
FLD   b    ; b
FLD   a    ; b, a
FLD   d    ; b, a, d
FMUL       ; b, a*d
FLD1       ; b, a*d, 1
FADDP      ; b, a*d+1
FDIVP      ; b/(a*d+1)
FLD   a    ; b/(a*d+1), a
FADDP      ; b/(a*d+1)+a
```

7 команд

FMUL [**d**]

FADD [a]

Пример

- Поменяем местами: b a d * 1 + / a +
- ОПН: a d * 1 + b **FDIVR** a +

```
FLD   a    ; a
FMUL  d    ; a*d
FLD1       ; a*d, 1
FADDP      ; a*d+1
FDIVR b    ; b/(a*d+1)
FADD  a    ; b/(a*d+1)+a
```

6 команд

Пример 2

- Команды с аргументом st(n) удобны для доступа к ранее вычисленному выражению
- Вычислить: (a + b) * x + y / (a + b)
- ОПН: a b + x * y a b + / +

```
FLD    a                    ; a
FADD   b                    ; a+b
FLD    x                    ; a+b, x
FMUL   st(0), st(1)         ; a+b, (a+b)*x
FLD    y                    ; a+b, (a+b)*x, y
FDIVRP st(2), st(0);        y/(a+b), (a+b)*x
FADDP                       ; y/(a+b)+(a+b)*x
```

Дополнительные арифметические операции

команда	
FABS	$\mathrm{ST} := \lvert\mathrm{ST}\rvert$
FCHS	$\mathrm{ST} := -\mathrm{ST}$ от CHange Sign
FRNDINT	округлить ST до целого от RouND INTeger
FSQRT	$\mathrm{ST} := \sqrt{\mathrm{ST}}$

Сравнение чисел

- Сравнивают ST(0) с src

команда	src
FCOM src	ST(n), вещественная переменная 32/64 бит
FCOMP src	
FCOMPP	ST(1)
FTST	0

Сравнение чисел

- FUCOM, FUCOMP, FUCOMPP – аналогичны FCOM**, но:
  - src – только регистр
  - если один из операндов QNaN – не вызывают исключения

- FICOM, FICOMP – аналогичны FCOM*, но src – целая переменная 16/32 бит

Сравнение чисел

- Устанавливает флаги **C0**, **C2**, **C3** в регистре **SW**

условие	C3	C2	C0
ST > src	0	0	0
ST < src	0	0	1
ST = src	1	0	0
несравнимы	1	1	1

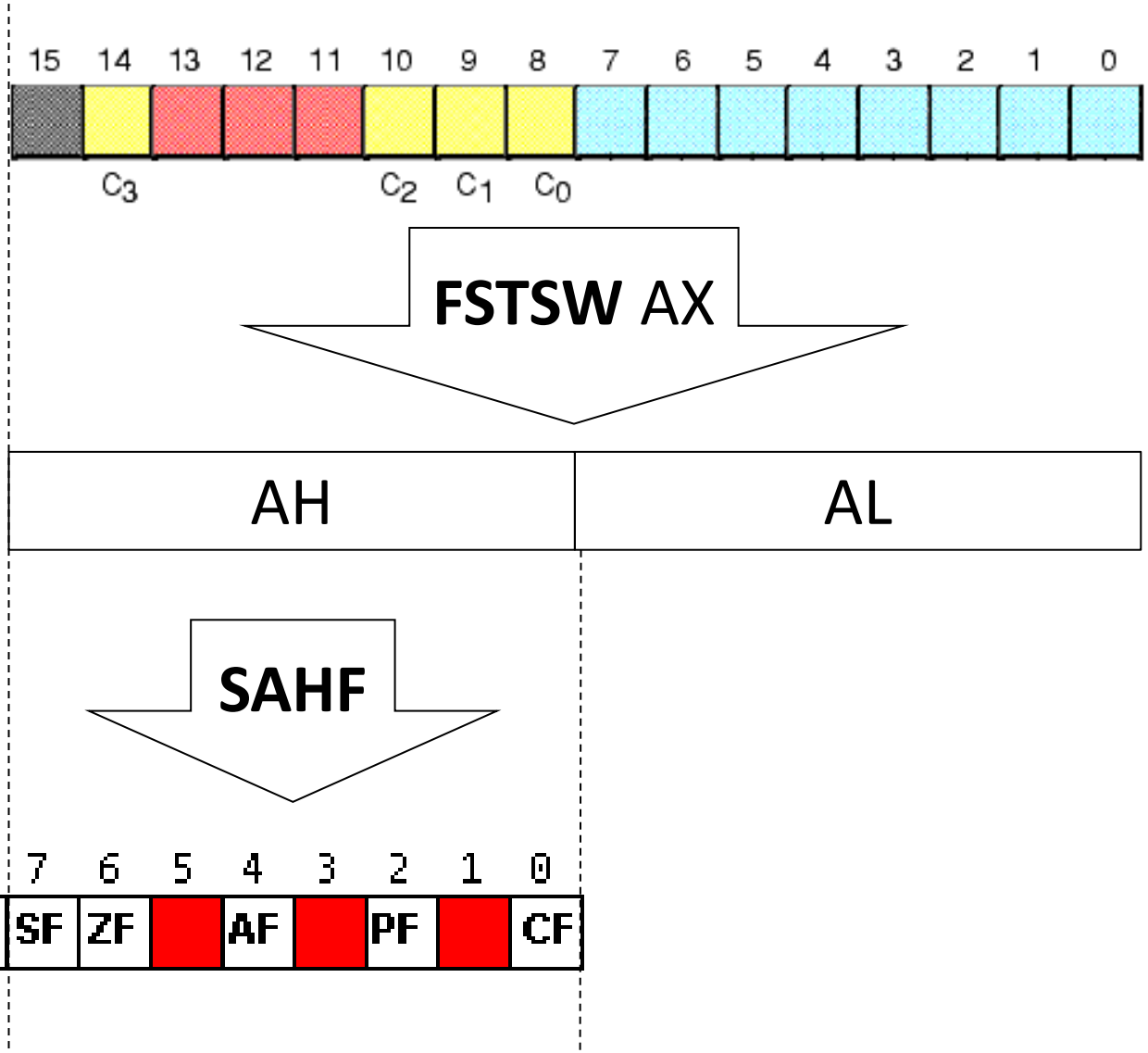
Сравнение чисел

- Флаги можно анализировать только на CPU
  => их нужно перенести в регистры CPU!

Стандартный способ:

```
FSTSW ax   ; AX := SW
SAHF       ; FLAGS := AH
j*         ; переход
```

Сравнение чисел



Сравнение чисел

- SAHF копирует:

флаг FPU	C3	C2	C1	C0
во флаг CPU	ZF	PF	-	CF

- После этого
  - результат сравнения (**ZF** и **CF**) проверяется так же, как для беззнаковых целых чисел (JB, JE, JA, JNE, …)
  - несравнимость проверяется по PF

Пример

Проверить a перед вычислением $\sqrt{a}$

```
    FLD   a     ; a
    FTST        ; сравнить a с 0
    FSTSW ax
    SAHF
    JB error    ; если a<0
                ; или JNAE error
    FSQRT
    …
error:
```

Сравнение чисел

- Команды сравнения, начиная с 80686:

FCOMI	аргументы	**FCOM**	аргумент
FCOMIP		**FCOMP**	
FUCOMI	st(0), st(n)	**FUCOM**	st(n)
FUCOMIP		**FUCOMP**	

Но сразу устанавливают ZF, PF и CF (не меняя AX), что короче и быстрее. Используйте их!

Пример

- Задача – та же

```
  FLD   a                 ; a
  FLDZ                    ; a, 0
    ; сравнить 0 с a
  FCOMIP st, st(0) ; a
  JA error                ; если 0>a
    ; или JNBE error
  FSQRT
error:
```

Пример

- Сравнить a и b

```
FLD   b          ; b
FLD   a          ; b, a
FCOMI st(0), st(1)
JB a_below_b ; a < b
JA a_above_b ; a > b
JE a_equal_b ; a = b
```

Анализ содержимого регистра

- **FXAM** (*от eXAMine*) – устанавливает флаги в зависимости от содержимого ST(0)

ST(0)	C3	C2	C0
неподдерживаемое	0	0	0
NaN	0	0	1
нормальное число	0	1	0
∞	0	1	1
0	1	0	0
пусто	1	0	1
денормализованное	1	1	0

Анализ типа числа

- Пример: проверить результат (a / b)

```
FLD   a    ; a
FDIV  b    ; a / b
FXAM
FSTSW ax
SAHF
JC error   ; если не-число, ∞ или пусто
```

Условная пересылка

- Начиная с 80686
- **FCMOV**cc ST, ST(n) – копирует ST(n) в ST если выполняется условие **cc** (флаги во **FLAGS)**
- Допустимые коды условий:
  - E, NE, B, BE, NB, NBE – смысл тот же, что для беззнаковых целых;
  - U – если несравнимы
  - NU – если сравнимы
  - прочие коды сравнения для беззнаковых целых (A, AE, NA, NAE) не поддерживаются

Пример

```
FLD a     ; a
FLD b     ; a, b
FCOMI st(0), st(1) ; b сравнили с a
```

- Найти максимум

```
     ; если b<a то st(0):=a
FCMOVB st(0), st(1); a, max(a,b)
```

- Или найти минимум

```
     ; если b>a то st(0):=a
FCMOVNBE st(0), st(1); a, min(a,b)
```