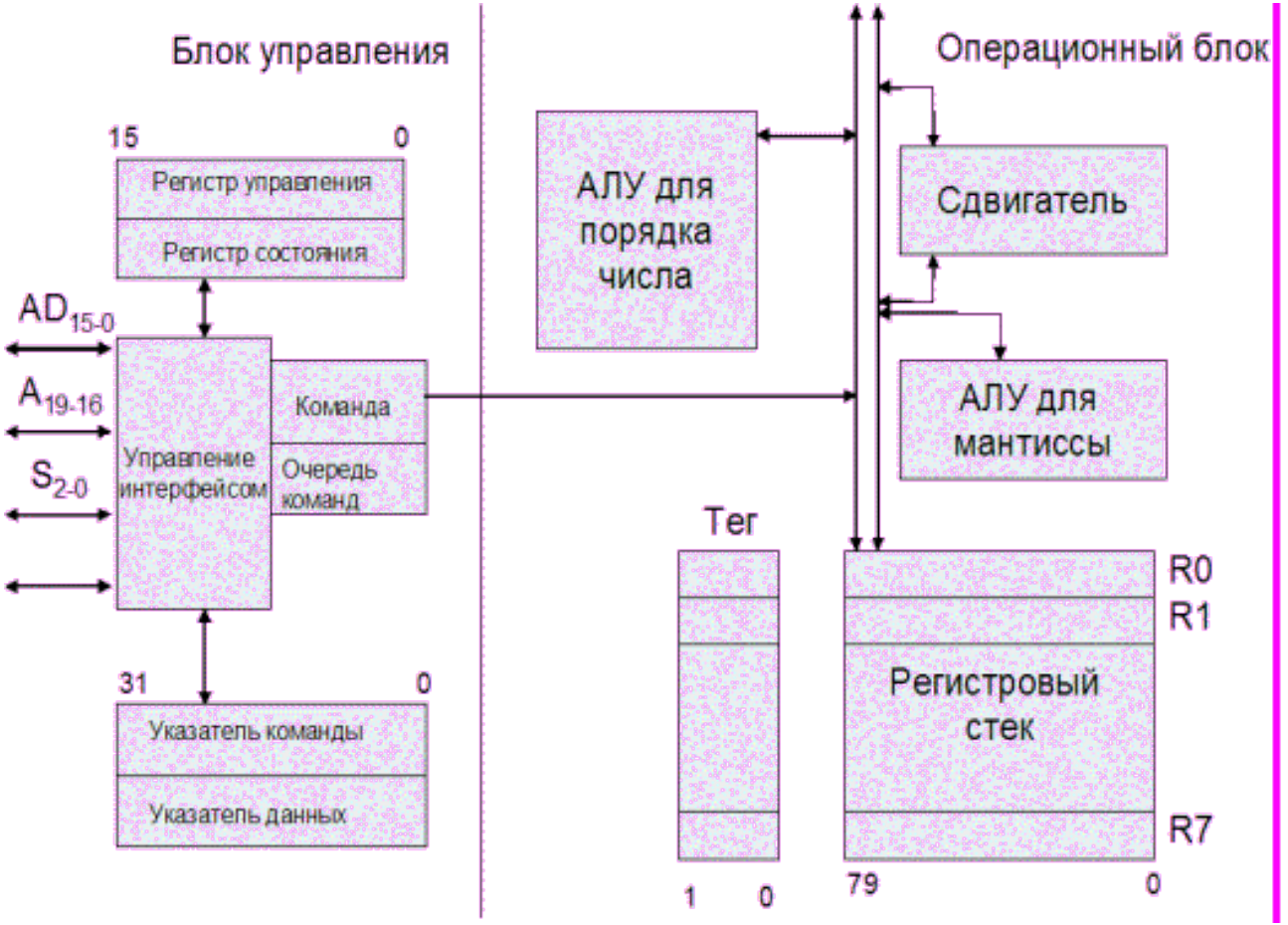
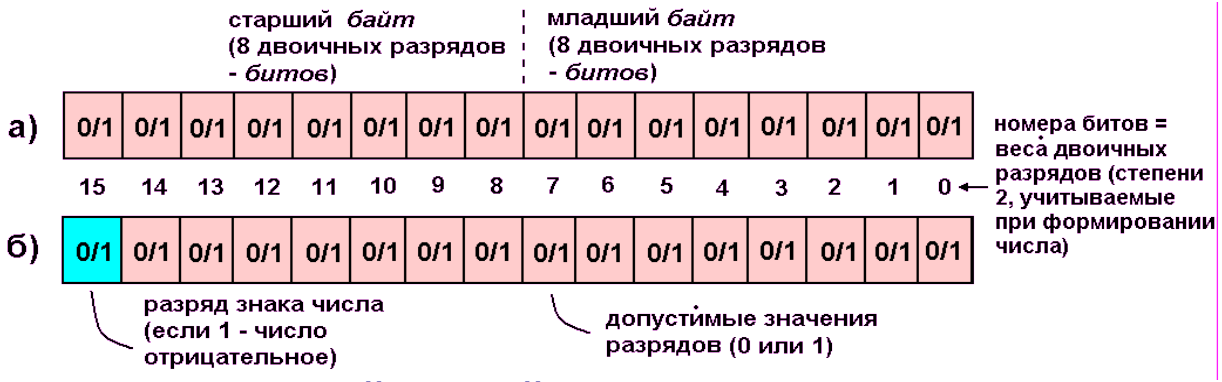
Арифметика с плавающей точкой в X86

Модуль операций с плавающей точкой (запятой) — это *стековый калькулятор*, работающий по *принципу постфиксной записи*. Аргументы помещаются в стек.
При выполнении операции требуемое количество аргументов снимается со стека. Результат операции помещается в стек. Поддерживается также прямая адресация аргументов в стеке относительно вершины.

В FPU числа хранятся в 80-битном формате с плавающей запятой, для записи или чтения из памяти могут использоваться:
- один из трёх форматов с плавающей точкой - 32, 64 и 80 бит
- целочисленные форматы - 16, 32 и 64 бита
- 80-битный BCD-формат.



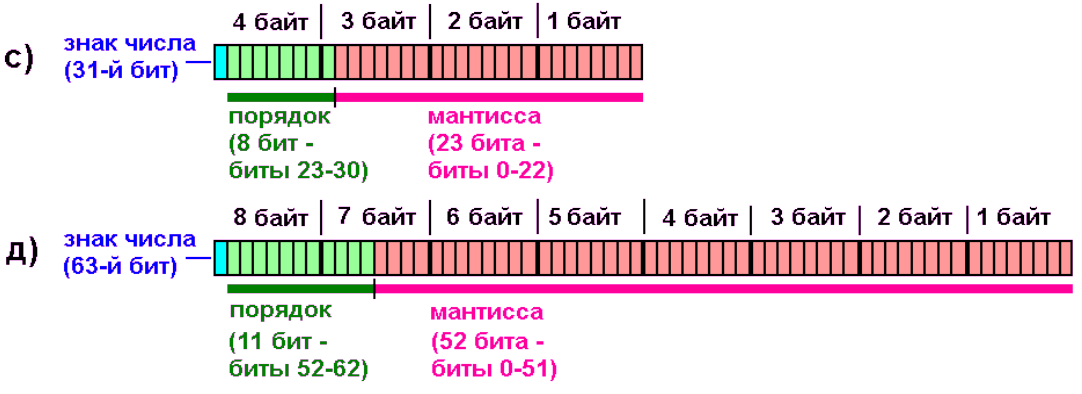
Представление чисел в процессорах фирмы Intel (стандарт IEEE 754, 1985)



а) - целое двухбайтовое число без знака; диапазон $(0 \div 2^{16}-1) = (0 \div 65'535_{10})$

б) - целое со знаком; диапазон от $-2^{16-1} = -32768_{10}$ до $2^{15}-1 = 32767_{10}$

Числа с плавающей запятой (тип *float*) во многих ЭВМ хранятся в формате IEEE* *Standard 754 Floating Points Numbers* (поддерживается в процессорах семейства Intel x86/87). Порядок представляется т.н. *смещённой формой (к истинному значению добавляется 127/1023, чтобы всегда иметь положительное значение);* целая часть мантиссы не сохраняется (всегда равна 1).



с) – вещественное одинарной точности (*single*; диапазон $1{,}5\times10^{-45} \div 3{,}4\times10^{38}$; точность 7÷8 десятичных зна́чащих цифр)

д) – вещественное двойной точности (*double;* диапазон $5\times10^{-324} \div 1{,}7\times10^{308}$; точность 15÷16 десятичных зна́чащих цифр)

***) - 10-ти байтовое представле́ние (*extended*) даёт диапазон от $3{,}4\times10^{-4932}$ до $1{,}1\times10^{4932}$ и точность 19÷20 десятичных значащих цифр**

е) Двоично-десятичный формат (Binary Coded Decimal - BCD)

BCD Integer - один байт, содержит одну десятичную цифру

Packed BCD Integer - один байт, содержит две десятичные цифры

* ***IEEE*** (*Institute of Electrical and Electronics Engineers*) – институт инженеров по электротехнике и электронике; IEEE является разработчиком большого количества стандартов в соответствующих областях деятельности.

Возможности арифметики с плавающей точкой

Математические операции:

- арифметические операции, сравнение, деление по модулю, округление, смена знака, модуль,
- квадратный корень,
- тригонометрические функции (*sin*, *cos*, *tg* и др.), обратные тригонометрические функции (*arcsin*, *arctg* и др.), логарифмические функции (log_2x, log_{10}x, log_ex), показательные функции (x^y, 2^x, 10^x, e^x), гиперболические функции (*sh*, *ch*, *th* и др.), обратные гиперболические функции (*arsh*, *arch*, *arth* и др.),
- загрузка константы (0, 1, число π, $log_2 10$, $log_2 e$, $lg(2)$, $ln(2)$) и некоторые другие специфические операции.

Сопроцессор i87 вычисляет любую функцию от аргументов двойной точности, давая результат не менее чем двойной точности с ошибкой младшего разряда округления. Полное соответствие стандарту IEEE 754 достигнуто только начиная с модели i387 (позднее 1985).

Возможности арифметики с плавающей точкой

FPU может обрабатывать *погранúчные состояния* с помощью специальных значений, представúмых форматом с плавающей запятой:

- денормализованное число (число, близкое к переполнению; при дальнейшем возрастании модуля денормализованное число становится бесконечностью),
- бесконечность (положительная и отрицательная), возникает при делении на нуль ненулевого значения а также при переполнениях,
- нечúсла (англ. *Not-a-Number* - NaN). Нечúсла могут определять такие случаи, как:
- неопределённость (*IND*), возникает при комплексном результате (например, при вычислении квадратного корня из отрицательного числа) и в некоторых других случаях,
- недействительное значение (*qNaN*, *sNaN*) - может использоваться компилятором (для предотвращения использования неинициализированных переменных) или отладчиком,
- нуль - в формате с плавающей запятой нуль также считается специальным значением.
- В зависимости от флагов FPU, специальные значения могут инициировать обработку исключения операционной системой.

Регистры FPU, их названия, взаимосвязь и назначение

r0÷r7 – регистры кольцевого стека сопроцессора

Служебные регистры:

- TWR (*Tags Word Register*) - регистр слова тегов (необходим для контроля над регистрами r0-r7; напр., для определения возможности записи), размер - 16 бит
- CWR (*Control Word Register*) - управляющий регистр сопроцес-сора (служит для управления режимами работы сопроцес-сора)
- SWR (*Status Word Register*) - регистр состояния, содержит информацию о текущем состоянии сопроцессора
- IPR (*Instruction Point Register*) - указатель команд
- DPR (*Data Point Register*) - указатель данных

Архитектура и команды FPU

Три группы регистров:

I. Стек процессора: регистры r0-r7, размерность каждого регистра - 80 бит.

II. Служебные регистры:

- регистр состояния SWR (*Status Word Register*) содержит информацию о текущем состоянии сопроцессора, размерность - 16 бит,
- управляющий регистр сопроцессора CWR (*Control Word Register*) служит для управления режимами работы сопроцессора, размер - 16 бит,
- регистр слова тегов TWR (*Tags Word Register*) необходим для контроля над регистрами r0-r7 (напр., для определения возможности записи), размер - 16 бит.

III. Регистры указателей:

- указатель данных DPR (*Data Point Register*), размерность - 48 бит,
- указатель команд IPR (*Instruction Point Register*), размер - 48 бит.

Архитектура и команды FPU

Система команд (*инструкций*) сопроцессора включает более 80 команд (инструкций FPU). Их обычно классифицируют по 5 группам:

I. Команды передачи данных:

- вещественные данные,
- целочисленные данные,
- десятичные данные (BCD-арифметика),
- загрузка констант (0, 1, число π, $log_2 10$, $log_2 e$, $lg(2)$, $ln(2)$),
- обмен данными,
- условная пересы́лка (Pentium II/III).

II. Команды сравнения данных:

- вещественные данные
- целочисленные данные
- анализ (сравнение) данных
- сравнения с нулём
- условное сравнение (Pentium II/III)

Архитектура и команды FPU

Система команд (*инструкций*) сопроцессора включает более 80 команд (инструкций FPU). Их обычно классифицируют по 5 группам:

III. Арифметические команды:

- вещественные данные (сложение, вычитание, умножение, деление)
- целочисленные данные (сложение, вычитание, умножение, деление)
- вспомогательные арифметические команды (квадратный корень, модуль,
- изменение знака, выделение порядка и мантиссы)

IV. Команды вычисления элементарных трансцендентных функций:

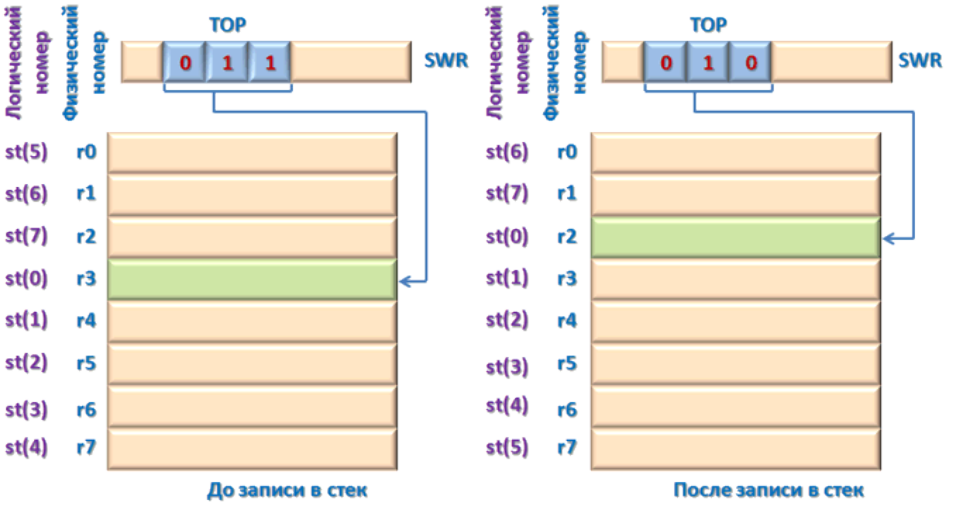
- тригонометрия (синус, косинус, тангенс, арктангенс)
- вычисление логарифмов и степеней

V. Команды управления:

- инициализация сопроцессора
- работа со средой
- работа со стеком
- переключение режимов

Выполнение операций с плавающей точкой

Регистровый стек сопроцессора организован по кольцево́му принципу (среди восьми регистров стека нет такого, который постоянно является вершиной стека). Все регистры стека с функциональной точки зрения абсолютно одинаковы и равноправны, вершина в кольцевом стеке сопроцессора является *плавающей*. Контроль текущей вершины осуществляется аппаратно с помощью 3-битового поля TOP регистра SWR.



- В поле TOP (*указа́тель на вершину стека*) задаётся номер регистра, являющегося в данный момент текущей вершиной стека (пример *косвенной адресации*).
- Команды сопроцессора оперируют не физическими номерами регистров r0÷r7, а их *логическими номерами* st(0)÷st(7). С их помощью реализуется относительная адресация регистров стека сопроцессора. Т.е., если текущей вершиной является регистр r3, то после записи в стек текущей вершиной становится регистр r2. По мере записи в стек, указатель его вершины движется по направлению к младшим номерам физических регистров (уменьшается на единицу). Если текущей вершиной является r0, то после записи очередного значения в стек текущей вершиной станет регистр r7. Логические номера регистров st(0)÷st(7) перемещаются вместе с изменением текущей вершины стека, логическая вершина всегда имеет имя st(0).
- Т.к. при составлении программы разработчик манипулирует не абсолютными, а относительными номерами регистров стека, могут возникнуть трудности при попытке интерпретации содержимого регистра тегов TWR с соответствующими физическими регистрами стека, для уточнения необходимо привлекать информацию из поля TOP регистра SWR.

Выполнение операций с плавающей точкой

- Т.о. образом реализуется принцип к*ольцевого стека*, обладающего большой гибкостью (в частности при передаче параметров в процедуру). Ясно, что для повышения гибкости разработки и использования процедур *нежелательно привязывать их по передаваемым параметрам к аппаратным ресурсам* (физическим номерам регистров сопроцессора). Много удобнее задавать порядок следования передаваемых параметров в виде логических номеров регистров, такой способ передачи является однозначным и не требует от разработчика знания излишних подробностей об аппаратных реализациях сопроцессора. Логическая нумерация регистров сопроцессора поддерживается на уровне системы команд, *идеально реализует эту идею*. При этом не имеет значения, в какой физический регистр стека сопроцессора были помещены данные перед вызовом подпрограммы, важен только порядок сле́дования параметров в стеке. Вследствие этого подпрограмме важно знать *только порядок размещения передаваемых параметров в стеке*.
- Новизна архитектуры в использовании для промежуточных вычислений 80-битного вещественного числа с 64-битной мантиссой и 15-битной экспонентой, стековой организации сопроцессора с восемью 80-битными регистрами и набором инструкций, обеспечивающим вычисление большого числа математических функций. 80-битный формат решал ряд известных трудностей организации вычислений и создания программного обеспечения для числовой обработки - было значительно снижено влияние ошибок округления при работе с 64-битными вещественными операндами, а также обеспечена точность вычислений для 18-значных двоично-десятичных и целых 64-битных чисел.

Выполнение операций с плавающей точкой

- *Процесс вы́борки очередной команды всегда инициирует основной (центральтый) процессор (*CPU*).* После выборки команда попадает одновременно в оба процессора. Любая команда сопроцессора имеет код операции, первые пять бит, которого имеют значение 11011 (27_{10} - т.н. *Escape-при́знак; исключённая последовательность, экранированная последовательность, от англ. Escape Sequence).* Такая команда выполняется FPU, а не CPU. Собственно параметры FPU-команды занимают оставшиеся 11 бит из двух байтов (для 16-битных процессоров Intel), начинающихся с указанной битовой комбинации.
- После получения кода команды и необходимых данных сопроцессор начинает её выполнение. Пока сопроцессор выполняет команду, центральный процессор выполняет программу дальше, действуя параллельно с вычислениями сопроцессора. Если следующая команда также является командой сопроцессора, CPU останавливается и ожидает завершения выполнения сопроцессором предыдущей команды.

Использование FPU

Отдельная группа команд, название которых начинается с F (*Float*). Операндом команд является, как правило, один из регистров st() или ячейка памяти; другой операнд обычно подразумевается как st(0); можно записывать просто st.

При использовании команд набора MMX (***M****ulti****M****edia E****x****tensions,* набор *целочисленных* SIMD-команд фирмы Intel для ускорения процессов кодирования/декодирования потоковых аудио и видеоданных, 1997) применяются 64-х битовые регистры MMX0-MMX7; физически совмещённые со стеком регистров FPU.

При использовании команд набора SSE (***S****treaming* ***S****IMD* ***EX****tensions*, набор SIMD-инструкций, разработанный Intel как ответ на аналогичный набор инструкций 3DNow! от AMD, 1999) применяются 8 (16 для x64) 128-битных регистров XMM0-XMM7 (XMM15); каждый регистр может содержать четыре 32-битных значения с *плавающей точкой одинарной точности*.

Использование FPU

Некоторые (часто используемые) команды FPU

*Команды передачи данных (аналог **MOV** для CPU):*

- **FLD** src ; загрузить вещественное число в st(0) (на вершину стека) из области памяти src (область памяти может быть 32, 64 или 80-битная)
- **FILD** src ; загрузить целое число в st(0) из памяти src (16, 32 или 64-битной)
- **FBLD** src ; загрузить двоично-десятичное число в st(0) из 80-битной области памяти
- **FLDZ** ; загрузить 0 в st(0)
- **FLD1** ; загрузить 1 в st(0)
- **FLDPI** ; загрузить число π в st(0)
- **FLDL2T** ; загрузить $log_2 10$ в st(0)
- **FLDTL2E** ; загрузить $log_2 e$ в st(0)
- **FLDLG2** ; загрузить $lg(2)$ в st(0)
- **FLDLN2** ; загрузить $ln(2)$ в st(0)

Использование FPU

Команды передачи данных (продолжение):

- **FST** dest ; запись вещественного числа из st(0) в память (область памяти 32, 64 или 80-бит)
- **FSTP** dest ; запись вещественного числа из st(0) в память (область памяти 32, 64 или 80-битная), при этом происходит выта́лкивание верши́ны из стека (*операция POP*)
- **FBST** dest ; запись двоично-десятичного числа в память (область памяти 80-бит)
- **FBSTP** dest ; запись двоично-десятичного числа в память (область памяти 80-бит, при этом происходит выталкивание вершины из стека)
- **FXCH** st(i) ; обме́н значениями вершины стека и стекового регистра i ($1 \le i \le 7$)

Использование FPU

Команды сравнения данных:

- **FCOM** ; сравнение st(0) и st(1), флаги устанавливаются, как при st(0)-st(1).
- **FCOM** src ; сравнение st(0) с операндом в памяти, операнд м.б. 32- или 64 бит
- **FCOMP** src ; сравнение вещественного числа в st(0) с операндом и с выталкиванием st(0) из стека, операнд м.б. регистром и областью памяти
- **FCOMPP** ; сравнение st(0) и st(1) с двойным выталкиванием из стека.
- **FICOM** src ; сравнение целых чисел в st(0) с операндом, операнд может быть 16- или 32-битным
- **FICOMP** src ; сравнение целых чисел в st(0) с операндом, операнд может быть 16- или 32-битной областью памяти или регистром; при выполнении операции происходит выталкивание st(0) из стека
- **FTST** ; проверка st(0) на нуль.
- **FUCOM** st(i) ; сравнение st(0) с st(i) без учёта порядков
- **FUCOMP** st(i) ; сравнение st(0) с st(i) без учета порядков, при выполнении операции происходит выталкивание из стека

Использование FPU

Команды сравнения данных (продолжение):

- **FUCOMPP** st(i) ; сравнение st(0) с st(i) без учёта порядков, при выполнении операции происходит двойное выталкивание из стека
- **FXAM** ; анализ содержимого вершины стека (результат помещается в биты c3-c2,c0; значения: 000b - неподде́рживаемый формат, 001b - не число, 010b - нормализованное число, 011b - бесконечность, 100b - нуль, 101b - пустой операнд, 110b - ненормализованное число)

Использование FPU

Арифметические команды:

- **FADD** src ; сложение вещественных чисел, st(0) ← st(0)+src, src суть 32- или 64-битное число **FADD** st(i),st ; то же самое st(i) ← st(i)+st(0)
- **FADDP** st(i),st ; сложение вещественных чисел, st(i) ← st(i)+st(0), при выполнении операции происходит выталкивание стека
- **FIADD** src ; сложение целых чисел, st(0) ← st(0)+src, src суть 16- или 32-битное число
- **FSUB** src ; вычитание вещественных чисел, st(0) ← st(0)-src, src суть 32- или 64-битное число
- **FSUB** st(i),st ; то же st(i) ← st(i)-st(0)
- **FSUBP** st(i),st ; вычитание вещественных чисел, st(i) ← st(i)-st(0), при выполнении операции происходит выталкивание стека
- **FSUBR** st(i),st ; обратное вычитание вещественных чисел, st(0) ← st(i)-st(0)
- **FSUBRP** st(i),st ; обратное вычитание вещественных чисел, st(0) ← st(i)-st(0), при выполнении операции происходит выталкивание стека

Использование FPU

Арифметические команды (продолжение):

- **FISUB** src ; вычитание целых чисел, st(0) ← st(0)-src; src суть 16- или 32-битное число
- **FISUBR** src ; вычитание целых чисел, st(0) ← st(0)-src, src суть 16- или 32-битное число, при выполнении операции происходит выталкивание из стека
- **FMUL** ; **FMUL** st(i) ; **FMUL** st(i),st ; умножение двух операндов, в первом случае st(0) ← st(0)*st(1), во втором случае st(0) ← st(i)*st(0), в третьем случае st(i) ← st(i)*st(0)
- **FMULP** st(i),st(0) ; умножение и выталкивание из стека: st(i) ← st(i)*st(0)
- **FIMUL** src ; умножение st(0) на целое число; st(0) ← st(0)□src, операнд может быть 16- и 32-битным числом
- **FDIV** ; **FDIV** st(i) ; **FDIV** st(i),st ; st(0) ← st(0) / st(1) ; st(0) ← st(0) / st(i) ; st(i) ← st(0) / st(i)
- **FDIVP** st(i),st ; деление с выта́лкиванием из стека: st(i) ← st(0) / st(i)
- **FIDIV** src ; деление целых чисел: st(0) ← st(i) / src, делитель src может быть 16- и 32-битным числом

Использование FPU

Арифметические команды (продолжение):

- **FDIVR** st(i),st ; обратное деление вещественных чисел: st(0) ← st(i) / st(0)
- **FDIVRP** st(i),st ; обратное деление вещественных чисел и выталкивание из стека: st(0) ← st(i) / st(0)
- **FIDIVR** src ; обратное деление целых чисел: st(0) ← src / st(0)
- **FSQRT** ; извлечь корень из st(0) и поместить на то же место
- **FSCALE** ; масштабирование: st(0) ← st(0)*2^st(1)
- **FEXTRACT** ; выделение мантиссы и порядка из числа st(0) - в st(0) помещается порядок, в st(1) - мантисса.
- **FPREM** ; нахождение остатка от деления: st(0) ← st(0) % st(1)
- **FRNDINT** ; округление до ближайшего целого числа, находящегося в st(0): st(0) ← INT(st(0))
- **FABS** ; нахождение абсолютного значения: st(0) ← ABS(st(0))
- **FCSH** ; изменение знака: st(0) ← **-** st(0)

Использование FPU

Трансцендентные функции:

- **FCOS** ; вычисление косинуса: st(0) ← *cos*(st(0)), содержимое st(0) интерпретируется как *угол в радианах*
- **FPTAN** ; частичный тангенс: st(1) ← tg(st(0)), st(0)=1.0 (реализовано для совместимости с более ранними моделями сопроцессоров)
- **FPATAN** ; вычисление арктангенса, вычисляется функция *arctg*(st(1) / st(0)). После вычисления функции происходит выталкивание из стека, после чего значение функции помещается в óного вершину st(0)
- **FSIN** ; вычисление синуса: st(0) ← *sin*(st(0)).
- **FSINCOS** ; вычисление синуса и косинуса: st(0) ← *sin*(st(0)) и st(1) ← *cos*(st(0))
- **F2XM1** ; вычисление 2^X-1: st(0) ← 2^st(0)-1
- **FYL2X** ; вычисление Y*log_2X: st(0) ← Y, st(1) ← X. Происходит выталкивание из стека и только потом в вершину стека st(0) помещается результат вычисления
- **FYL2XP1** : вычисление Y*log_2X. st(0) ← Y, st(1) ← X. Происходит выталкивание из стека и только потом в вершину стека st(0) помещается результат вычисления

Использование FPU

Команды управления блоком FPU:

- **FINIT** ; инициализация сопроцессора.
- **FSTSW** ax ; запись слова состояния FPU в регистр **ax** CPU (*удобный вариант выполнения условных переходов по сравнению чисел с плавающей точкой*)
- **FSTSW** dest ; запись слова состояния в dest
- **FLDCW** src ; загру́зка управляющего слова (16 бит) из dest
- **FSTCW** dest ; сохранение управляющего слова в dest
- **FCLEX** ; сброс исключений
- **FLDENV** src ; загрузка состояния оборудования из памяти
- **FSAVE** dest ; сохранение состояния оборудования и файла регистров в памяти
- **FRSTOR** src ; загрузка состояния оборудования и файла регистров в памяти
- **FINCSTP** ; инкремент указателя стека (изменяет поле TOP)
- **FDECSTP** ; декремент указателя стека (изменяет поле TOP)
- **FFREE** st(i) ; освобождение регистра (поме́тка регистра st(i) как свободного)
- **FNOP** ; холостая операция сопроцессора
- **WAIT/FWAIT** ; ожидание процессором завершение операции сопроцессором