* **Работа с математическом сопроцессором:**
  + FPU — Floating Point Unit, т.е. устройство работы с вещественными числами
  + 8 регистров по 80 бит (ST(0) .. ST(7)) в стеке, где ST(0) — вершина стека, всегда хранящая последнее значение при загрузке данных, сдвигая предыдущие значения вниз по стеку
    - введённые в стек числа автоматически становятся вещественными
  + **типы даных:**
    - целочисленные:
      * byte (DB) — 1 байт
      * word (DW) — 2 байта
      * dword (DD) — 4 байта
      * qword (DQ) — 8 байт
      * ten bytes (DT) — 10 байт
    - дробные:
      * (чтобы объявить дробное значение, нужно в записи использовать точку)
      * real4 (DD) (float) — 4 байта
      * real8 (DD) (double) — 8 байт
      * real10 (DD) (FPU) — 10 байт
  + **команды:**
    - (зачастую вместо ST(i)может использоваться регистр или переменная)
    - инициализация:
      * finit — сброс FPU в состояние по умолчанию (округление к ближайшему целому)
      * fwait — команда ожидания завершения операции для синхронизации работы с основным процессороом
    - загрузка:
      * fild word ptr [var1] — загрузка целого числа (16-битная переменная var1 загружается в верхний регистра стека ST)
      * fld word ptr [var1] — загрузка дробного числа (загружается 16-битная переменная var1)
      * fld1 — загрузка единицы в ST(0)
      * fldz — загрузка нуля в ST(0)
      * fldpi — загрузка в ST(0)
    - арифметика:
      * fadd st(0), st(1) — сложение
      * fsub st(0), st(1) — вычитание
      * fmul st(0), st(1) — умножение
      * fdiv st(0), st(1) — деление
    - выгрузка:
      * fstp word ptr [res] — сохранение дробного значения ST0 в 16-битную переменную res с автоматическим удалением из стека
      * fist word ptr [res] — сохранение целочисленного значения ST0
      * fst word ptr [res] — то же, но без удаления из стека
      * fistp word ptr [res] — сохранение целочисленного значения ST0
    - работа со стеком:
      * fld st(i) — копирование ST(i) в вершину стека
      * fxch st(i) — обмен ST(i) с вершиной стека
      * ffree st(i) — освобождение регистра (маркировка как свободного)
      * fincstp — изменение указателя вершины стека
    - специальные математические функции:
      * fsin — синус от ST(0)
      * fcos — косинус от ST(0)
      * fsqrt — корень от ST(0)
      * fscale — умножает ST(0) на
    - операции сравнения:
      * fcom ST(i) — сравнение ST(0) и ST(i) без извлечения регистра из стека, с установкой флагов состояния FPU
        + флаги состояния FPU:

С0 = С2 = С3 = 0, если ST(0) больше

С0 = 1, С2 = С3 = 0, если ST(0) меньше

С0 = С2 = 0, С3 = 1, если ST(0) равно

С0 = С2 = С3 = 1, если ошибка сравнения

* + - * + после выполнения надо выгрузить регистр состояния FPU во флаги состояния основного процессора:

fstsw ax — выгрузка регистра состояния FPU в регистр ax

sahf — выгрузка регистра ah (ст.байт регистра ax) во флаги процессора

флаги переписываются:

C0 🡪 CF (флаг переноса)

C2 🡪 PF (флаг паритета)

C3 🡪 ZF (флаг нуля)

* + - * fcomp ST(i) — то же, но после удаляет ST(0)из стека
      * fcomi st(i) — то же, но устанавливает флаги основого процессора
      * fcompp — сравнивает ST(0) и ST(1), после чего их удаляет
      * ftst — сравнение ST(0) с нулём
      * fxam — возвращение типа числа ST(0) (нормальное, ноль, бесконечность, NaN, т.д.)
* работа с массивами (*БЕЗ ИСПОЛЬЗОВАНИЯ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА*):
  + **важно:** младший байт имеет младший адрес
  + loadsw — команда, которая загружает 2 байта того, что хранится по адресу   
    в sib или esi, в ax
    - после выполнения автоматически меняется значение SI на 2, причём:
      * если DF = 0 (для этого вызов CLD), то SI увеличится на 2
      * если DF = 1 (для этого вызов STD), то SI уменьшится на 2
    - loadsb, loadsd — для обработки байта или двойного слова