# 7. Работа с математическим сопроцессором

## Программная модель сопроцессора

Программная модель арифметического сопроцессора состоит из 3 групп регистров.

* Регистры R0..R7 – предназначены для хранения вещественных операндов, которые используются в вычислениях. Каждый регистр содержит 80 бит (0-63 – мантисса, 64-78 – порядок, 79 – знак числа). Эти регистры составляют стек сопроцессора и оптимизированы на реализацию вычислений с использованием обратной польской записи.
* Три служебных регистра SWR, CWR и TWR длиной 16 бит каждый.

SWR – регистр состояния сопроцессора. Содержит информацию о текущем состоянии сопроцессора, указывает, какой из регистров R0..R7 является вершиной стека сопроцессора, какие исключения возникли после выполнения последней команды и каковы особенности ее выполнения. Этот регистр является аналогом регистра флагов центрального процессора.

CWR – управляющий регистр сопроцессора. С помощью его полей можно регулировать точность выполнения вычислений, управлять округлением, маскировать исключения.

TWR – регистр слова тегов. Используется для контроля за состоянием каждого из регистров R0..R7. Для этого каждому из регистров стека сопроцессора в регистре TWR отведено по 2 бита: 0, 1 – R0; 2, 3 – R1 и т.д.

* Два регистра указателей DPR и IPR. Используются при обработке исключительных ситуаций.

DPR – регистр указателя данных. Используется для хранения адреса операнда команды, вызвавшей исключение.

IPR – регистр указателя команды. Используется для хранения адреса вызвавшей исключение команды.

Стек регистров сопроцессора организован по принципу кольца. Вершина стека является плавающей и перемещается после записи операнда в вершину (например, если текущей вершиной является регистр R0, то после записи значения в стек текущей вершиной станет регистр R1). Команды сопроцессора не оперируют физическими номерами регистров R0..R7. Они используют логические номера регистров ST(0), ST(1)...ST(7) относительно текущей вершины стека.

Регистр состояния сопроцессора SWR содержит 16 бит, назначение которых описано в таблице 7.1.

Таблица 7.1.

|  |  |  |
| --- | --- | --- |
| Бит | Обозначение | Назначение |
| 0 | IE | Недействительная операция |
| 1 | DE | Денормализованный операнд |
| 2 | ZE | Ошибка деления на нуль |
| 3 | OE | Ошибка переполнения – выход порядка за максимально допустимый диапазон значений |
| 4 | UE | Ошибка антипереполнения (результат слишком маленький) |
| 5 | PE | Ошибка точности – округление числа при выходе за пределы разрядной сетки |
| 6 | SF | Ошибка работы стека сопроцессора. 1 – возникла одна из исключительных ситуаций PE, UE или IE, выполнена попытка записи в заполненный стек или чтения из пустого стека. После анализа этого бита его нужно обнулить вместе с битами PE, UE, IE. |
| 7 | ES | Суммарная ошибка работы сопроцессора. 1 – возникла любая из шести исключительных ситуаций (биты 0-5). |
| 8 | C0 | Код условия |
| 9 | C1 | Код условия |
| 10 | C2 | Код условия |
| 11-13 | TOP | Номер физического регистра R0..R7, который является текущей вершиной стека |
| 14 | C3 | Код условия |
| 15 | B | Бит занятости. 1 – сопроцессор выполняет команду или происходит прерывание от основного процессора. 0 – сопроцессор свободен |

Биты C0..C3 являются аналогом регистра флагов центрального процессора. В них команды сопроцессора записывают коды условий, интерпретация которых зависит от конкретной команды.

Регистр управления сопроцессором CWR содержит 16 бит, назначение которых описано в таблице 7.2.

Таблица 7.2.

|  |  |  |
| --- | --- | --- |
| Бит | Обозначение | Назначение |
| 0 | I | Маски исключений. Предназначены для маскирования исключительных ситуаций, возникновение которых фиксируется битами 0-5 регистра SWR. 1 – соответст-вующее исключение обрабатывается самим сопроцессором. 0 – при возникновении исключения возбуждается прерывание 10h, обработчик которого должен быть написан программистом. |
| 1 | D |
| 2 | Z |
| 3 | O |
| 4 | U |
| 5 | P |
| 6 | Зарезервировано |  |
| 7 | IEM | Маска разрешения прерываний. 1 – даже при возникновении незамаскированного исключения (бит 0 -5 равен 0) прерывание не возбуждается. |
| 8-9 | PC | Поле управления точностью:  00 –мантисса занимает 24 бита,  10 – мантисса занимает 53 бита,  11 – мантисса занимает 64 бита. |
| 10-11 | RC | Поле управления округлением:  00 – округление по обычным правилам,  01 – округление в меньшую сторону,  10 – округление в большую сторону,  11 – отбрасывание дробной части результата (используется в операциях целочисленной арифметики). |
| 13-15 | Зарезервировано |  |

Регистр тегов TWR представляет совокупность двухразрядных полей, соответствующих определенному физическому регистру стека сопроцессора. Значение поля характеризует состояние соответствующего физического регистра:

* 00 – регистр занят допустимым ненулевым значением,
* 01 – регистр содержит ноль,
* 10 – регистр содержит одно из специальных значений, кроме нуля,
* 11 – регистр пуст и в него можно записать число.

## Форматы данных сопроцессора

Сопроцессор работает с числами в следующих форматах:

* двоичные целые числа длиной 16, 32 и 64 бита;
* упакованные целые десятичные числа длиной до 9 байт (могут содержать до 18 десятичных цифр);
* вещественные числа в коротком (32 бита), длинном (64 бита) и расширенном (80 бит) форматах.

Вещественные числа являются основным форматом, с которым работает сопроцессор. Они представляются в виде мантиссы и порядка, связанных формулой:



Здесь A – вещественное число, M – мантисса числа, p – порядок числа.

Мантисса должна быть представлена в нормализованном виде, т.е. удовлетворять соотношению: 1 ≤ M < 2. Другими словами, в мантиссе всегда должна быть единичная целая часть. Требование нормализованности позволяет избежать случаев неоднозначного представления одного и того же числа вещественными числами с различным значением порядка.

В сопроцессоре на аппаратном уровне принято соглашение, что порядок p всегда задается неотрицательным значением q, называемым характеристикой и связанным с p формулой:

q = p + фиксированное смещение.

Форматы вещественных чисел описаны в таблице 7.3.

Таблица 7.3

|  |  |  |  |
| --- | --- | --- | --- |
| Формат числа | Короткий | Длинный | Расширенный |
| Длина (бит) | 32 | 64 | 80 |
| Диапазон значений | 10-38..1038 | 10-308..10308 | 10-4392..104392 |
| Мантисса M | 23 бита  0-22 биты | 53 бита  0-51 биты | 64 бита  0-62 биты |
| Характеристика q | 8 бит  23-30 биты | 11 бит  52-61 биты | 15 бит  63-78 биты |
| Диапазон характеристик | 0..255 | 0..2047 | 0..32767 |
| Значение фиксированного смещения | +127 | +1023 | +16383 |
| Диапазон порядков | -126..+127 | -1022..  +1023 | -16382..  +16383 |
| Бит знака числа | 31 | 62 | 79 |
| Директива описания данных в программе | DD | DQ | DT |

Основным форматом для сопроцессора является расширенный формат вещественных чисел. Если используются другие вещественные или целочисленные форматы, сопроцессор выполняет их преобразование в расширенный формат.

Примеры описания вещественных чисел в программе на языке ассемблера:

; описание числа 51.25 в коротком формате

DD 51.25

DD 51.25E0

DD 0.5125E2

; описание числа 3 в длинном формате

DQ 3.0

DQ 0.3E1

; описание числа 0.005 в расширенном формате

DT 0.005

DT 5.0E-3

При работе с дампом вещественного числа нужно иметь ввиду следующие моменты:

* в коротком (32 бита) и длинном (64 бита) представлении единичный бит целой части мантиссы не хранится, а добавляется к записи числа на аппаратном уровне. Это позволяет немного увеличить разрядность мантиссы;
* в длинном (80 бит) представлении вещественного числа единичный бит целой части мантиссы присутствует в явном виде.

Кроме обычных чисел сопроцессор умеет работать с некоторыми специальными форматами, которые могут получиться в результате выполнения некоторых математических операций. Они называются специальными численными значениями.

Существуют следующие специальные численные значения.

* Денорамлизованные вещественные числа – числа, меньше минимального нормализованного числа. Это числа очень близкие к нулю. Биты порядка заполнены нулями;
* Нуль. Возможен положительный нуль (все биты числа равны нулю) и отрицательный (все биты равны нулю, а знаковый – единице);
* Бесконечность. Бывает положительная и отрицательная. В положительной нулевыми являются биты знака и мантиссы, а порядок заполнен единицами. В отрицательной бит знака равен единице.
* Не-числа. Существуют сигнальные и тихие не-числа. В сигнальных первый бит мантиссы равен 0 (или 10 для расширенного формата), все биты порядка установлены в 1. Сопроцессор не может формировать сигнальное не-число, а программист может загрузить его в стек сопроцессора преднамеренно для вызова исключительной ситуации. В тихом не-числе первый бит мантиссы равен 1 (11 для расширенного формата), все биты порядка установлены в 1. Сопроцессор может генерировать тихие не-числа в операциях, если один из операндов является тихим не-числом.
* Неопределенность. Знаковый бит равен 1, первый бит мантиссы – 1 (11 для расширенного формата), остальные биты мантиссы сброшены в 0, порядок заполнен единицами.
* Неподдерживаемое число. К таким числам относятся все остальные ситуации.

## Команды сопроцессора

Для команд сопроцессора используются следующие мнемонические соглашения.

* Все команды начинаются с буквы F.
* Вторая буква определяет тип операнда, с которым работает команда. I – целое двоичное число, B – целое число в упакованном формате, в остальных случаях – вещественное чисо.
* Последняя буква P – после выполнения команды операнд выталкивается из стека сопроцессора.
* Последняя или предпоследняя буква R – реверс операндов при выполнении команды.

Команды передачи данных приведены в таблице 7.4.

Таблица 7.4

|  |  |
| --- | --- |
| Команда | Действие |
| Передача вещественных чисел | |
| FLD источник | Загрузка числа из памяти в вершину стека сопроцессора |
| FST приемник | Сохранение числа из вершины стека сопроцессора в память |
| FSTP приемник | Сохранение числа из вершины стека сопроцессора в память с последующим выталкиванием его из вершины стека |
| Передача целых чисел | |
| FILD источник | Загрузка числа из памяти в вершину стека сопроцессора |
| FIST приемник | Сохранение числа из вершины стека сопроцессора в память |
| FISTP приемник | Сохранение числа из вершины стека сопроцессора в память с последующим выталкиванием его из вершины стека |
| Передача упакованных чисел | |
| FBLD источник | Загрузка числа из памяти в вершину стека сопроцессора |
| FBST приемник | Сохранение числа из вершины стека сопроцессора в память |
| FBSTP приемник | Сохранение числа из вершины стека сопроцессора в память с последующим выталкиванием его из вершины стека |
| Другие способы передачи | |
| FXCH ST(I) | Обмен вершины стека ST(0) с другим регистром стека ST(I) |

Команды загрузки констант в вершину стека сопроцессора перечислены в таблице 7.5.

Таблица 7.5

|  |  |
| --- | --- |
| Команда | Действие |
| FLDZ | Загрузить 0 |
| FLD1 | Загрузить 1 |
| FLDPI | Загрузить значение π |
| FLDL2T | Загрузить двоичный логарифм десяти |
| FLDL2E | Загрузить двоичный логарифм числа e |
| FLDLG2 | Загрузить десятичный логарифм двойки |
| FLDLN2 | Загрузить натуральный логарифм двойки |

Группа команд сравнения выполняют сравнение содержимого вершины стека ST(0) с источником и устанавливают соответствующие флаги сопроцессора или основного процессора (начиная с Pentium Pro). Источник может находиться или в регистре стека ST(i), не являющемся вершиной, или в памяти. Если источник в команде не указан, то сравнение выполняется с регистром ST(1). При сравнении целых чисел источник может находиться только в памяти.

Команды сравнения перечислены в таблице 7.6.

Таблица 7.6

|  |  |
| --- | --- |
| Команда | Действие |
| FCOM источник | Сравнение вещественных чисел |
| FCOMP источник | Сравнение вещественных чисел и выталкивание операнда из вершины стека |
| FCOMPP | Сравнение ST(0) и ST(1) и выталкивание их из стека |
| FUCOM источник | Сравнение вещественных данных без учета порядков |
| FUCOMP источник | Сравнение вещественных данных без учета порядков и выталкивание операнда из вершины стека |
| FUCOMPP | Сравнение ST(0) и ST(1) без учета порядков и выталкивание их из стека |
| FICOM источник | Сравнение целочисленных данных |
| FICOMP источник | Сравнение целочисленных данных и выталкивание операнда из вершины стека |
| FCOMI ST(0), ST(i)  (Pentium Pro) | Сравнение вещественных чисел с установкой флагов основного процессора |
| FCOMIP ST(0), ST(i)  (Pentium Pro) | Сравнение вещественных чисел с установкой флагов основного процессора и выталкивание операнда из вершины стека |
| FUCOMI ST(0), ST(i)  (Pentium Pro) | Сравнение вещественных чисел без учета порядков с установкой флагов основного процессора |
| FUCOMIP ST(0), ST(i)  (Pentium Pro) | Сравнение вещественных чисел без учета порядков с установкой флагов основного процессора и выталкивание операнда из вершины стека |
| FTST | Сравнение содержимого ST(0) с нулем |
| FXAM | Анализ содержимого ST(0) и определение типа находящегося в нем числа |

После выполнения команд сравнения устанавливаются по результатам сравнения устанавливаются флаги в регистре SWR в соответствии с таблицей 7.7.

Таблица 7.7

|  |  |  |  |
| --- | --- | --- | --- |
| Условие | Флаг С3 | Флаг C2 | Флаг C0 |
| ST(0) > источник | 0 | 0 | 0 |
| ST(0) < источник | 0 | 0 | 1 |
| ST(0) = источник | 1 | 0 | 0 |
| Операнды несравнимы | 1 | 1 | 1 |

Для реализации условных переходов по значениям этих флагов их нужно поместить в регистр флагов основного процессора. Это делается последовательным выполнением двух команд:

FSTSW ; загрузить регистр SWR в AX.

SAHF ; загрузить AH в младший байт регистра флагов.

После выполнения этих команд значение C0 находится в CF, C2 – PF, C3 – ZF.

Команды сравнения для процессора Pentium Pro позволяют сразу выполнить такую загрузку флагов в процессе выполнении самой команды сравнения.

Результатами действия команды FXAM является установка флагов C3, C2, C0. Возможные комбинации значений приведены в таблице 7.8.

Таблица 7.8

|  |  |  |  |
| --- | --- | --- | --- |
| Тип числа в ST(0) | Флаг С3 | Флаг C2 | Флаг C0 |
| Не поддерживаемое | 0 | 0 | 0 |
| Не-число | 0 | 0 | 1 |
| Нормальное конечное число | 0 | 1 | 0 |
| Бесконечность | 0 | 1 | 1 |
| Нуль | 1 | 0 | 0 |
| Регистр пуст | 1 | 0 | 1 |
| Денормализованное число | 1 | 1 | 0 |

Группа команд арифметических операций предназначена для реализации основных арифметических действий над вещественными и целыми числами. В командах обработки вещественных чисел в качестве источника может выступать либо регистр из стека сопроцессора, либо операнд в памяти, имеющий короткий или длинный формат. В командах обработки целых чисел источник может быть только операндом в памяти.

Описание базовых арифметических команд приведено в табл. 7.9.

Таблица 7.9

|  |  |
| --- | --- |
| Команда | Действие |
| FADD | Сложение ST(0) и ST(1). Результат помещается в ST(0) |
| FADD источник | Сложение ST(0) и источника. Результат помещается в ST(0) |
| FADD ST(i), ST | Сложение ST(i) и ST(0). Результат помещается в ST(i) |
| FADDP ST(i), ST | Сложение ST(i) и ST(0) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |
| FSUB | Вычитание значения ST(1) из ST(0). Результат помещается в ST(0) |
| FSUB источник | Вычитание значения источника из ST(0). Результат помещается в ST(0) |
| FSUB ST(i), ST | Вычитание значения ST(0) из ST(i). Результат помещается в ST(i) |
| FSUBP ST(i), ST | Вычитание значения ST(0) из ST(i) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |
| FSUBR ST(i), ST | Вычитание значения ST(i) из ST(0). Результат помещается в ST(i) |
| FSUBRP ST(i), ST | Вычитание значения ST(i) из ST(0) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |
| FMUL | Умножает значение ST(0) на ST(1). Результат помещается в ST(0) |
| FMUL ST(i) | Умножает значение ST(0) на ST(i). Результат помещается в ST(0) |
| FMUL ST(i) , ST | Умножает значение ST(0) на ST(i). Результат помещается в ST(i) |
| FMULP ST(i) , ST | Умножает значение ST(0) на ST(i) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |
| FDIV | Делит значение ST(0) на ST(1). Результат помещается в ST(0) |
| FDIV ST(i) | Делит значение ST(0) на ST(i). Результат помещается в ST(0) |
| FDIV ST(i), ST | Делит значение ST(0) на ST(i). Результат помещается в ST(i) |
| FDIVP ST(i), ST | Делит значение ST(0) на ST(i) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |
| FDIVR ST(i), ST | Делит значение ST(i) на ST(0). Результат помещается в ST(i) |
| FDIVRP ST(i), ST | Делит значение ST(i) на ST(0) с выталкиванием значения из ST(0). Результат помещается в ST(i-1) |

Дополнительные арифметические команды предназначены для вычисления различных функций. Их перечень приведен в таблице 7.10.

Таблица 7.10

|  |  |
| --- | --- |
| Команда | Действие |
| FSQRT | Вычисляет квадратный корень из значения ST(0). Результат помещается в ST(0). |
| FABS | Вычисляет модуль значения ST(0). Результат помещается в ST(0). |
| FCHS | Изменение знака ST(0) на противоположный |
| FXTRACT | Выделение мантиссы и порядка из значения ST(0). Мантисса помещается в ST(0) и представляется вещественным числом с нулевым порядком. Порядок заносится в ST(1) и представляется вещественным числом со знаком, имеющим значение истинного порядка числа (без учета фиксированного смещения) |
| FSCALE | Команда, обратная FXTRACT. ST(0) – мантисса числа, ST(1) – порядок числа.  ST(0) = ST(0) \* 2 ST(1) |
| FRNDINT | Округляет до целого значение в ST(0) в соответствии с порядком округления, заданным регистром CWR |
| FPREM | Нахождение частичного остатка от деления путем последовательного вычитания не более 64 раз содержимого ST(1) из ST(0) |
| FSIN | Вычисление sin(x), x расположен в ST(0). Результат помещается в ST(0).  -263<=x<=263 |
| FCOS | Вычисление cos(x), x расположен в ST(0). Результат помещается в ST(0).  -263<=x<=263 |
| FSINCOS | Вычисление sin(x) и cos(x). ST(1)=sin(ST(0)).  ST(0)=cos(ST(0)). -263<=x<=263 |
| FTAN | Вычисление частичного тангенса tg(a)=x/y.  ST(0)=x, ST(1)=y. -263<=a<=263 |
| FPATAN | Вычисление арктангенса a=arctg(x/y). ST(0)=x, ST(1)=y. ST(0)=x, ST(1)=y. Результат заносится в ST(0). 0 < |y| < |x| < ∞. |
| F2XM1 | Вычисление 2x-1. -1 < x < 1. |
| FYL2X | Вычисление y\*Log2(x). x – ST(0), y – ST(1).  0< x <∞, -∞ < y < +∞. |
| FYL2XP1 | Вычисление y\*Log2(x+1). x – ST(0), y – ST(1).  0<= |x| <1-20.5/2 , -∞ < y < +∞. |

Группа команд управления сопроцессором предназначена для выгрузки/загрузки управляющих регистров, анализа и установки значений флагов сопроцессора. Эти команды в качестве операнда всегда имеют участок памяти определенной длины. Команды управления приведены в таблице 7.11.

Таблица 7.11

|  |  |
| --- | --- |
| Команда | Действие |
| FINIT | Инициализация сопроцессора. Эквивалентна последовательности команд WAIT FNINIT. CWR=037h, SWR=0, TWR=0FFh, DPR=0, IPR=0 |
| FNINIT | Инициализация сопроцессора без ожидания обработки исключительных ситуаций |
| FLDCW источник | Загрузка CWR из слова в памяти |
| FSTCW приемник | Запись CWR в слово памяти. Эквивалентна WAIT FNSTCW |
| FNSTCW приемник | Запись CWR в слово памяти без ожидания обработки исключительных ситуаций |
| FSTSW приемник | Запись SWR в слово памяти. Эквивалентна WAIT FNSTSW |
| FNSTSW приемник | Запись SWR в слово памяти без ожидания обработки исключительных ситуаций |
| FSAVE приемник | Сохранение полного состояния сопроцессора в участок памяти 94 или 108 байт. Эквивалентна WAIT FNSAVE |
| FNSAVE приемник | Сохранение полного состояния сопроцессора без ожидания обработки исключительных ситуаций |
| FXSAVE приемник  (Pentium II) | Быстрое сохранение полного состояния сопроцессора в участке памяти размером 512 байт. |
| FRSTOR источник | Восстановление полного состояния сопроцессора, сохраненного командой FSAVE |
| FXSTOR источник  (Pentium II) | Восстановление полного состояния сопроцессора, сохраненного командой FXSAVE |
| FSTENV приемник | Сохранение пяти вспомогательных регистров в приемнике (CWR, SWR, TWR, DPR, IPR). Эквивалентна WAIT FNSTENV |
| FNSTENV приемник | Сохранение пяти вспомогательных регистров в приемнике (CWR, SWR, TWR, DPR, IPR) без ожидания обработки исключительных ситуаций |
| FLDENV источник | Загрузка из памяти (14 или 28 байт) пяти вспомогательных регистров сопроцессора (CWR, SWR, TWR, DPR, IPR) |
| FWAIT  WAIT | Ожидание готовности сопроцессора |
| FCLEX | Обнуление флагов исключений в регистре SWR (PE, UE, OE, ZE, DE, IE, ES, SE, B). Эквивалентна WAIT FNCLEX |
| FNCLEX | Обнуление флагов исключений в регистре SWR (PE, UE, OE, ZE, DE, IE, ES, SE, B) без ожидания обработки исключительных ситуаций |
| FINCSTP | Поле TOP в CSW увеличивается на 1 |
| FDECSTP | Поле TOP в CSW уменьшается на 1 |
| FFREE ST(i) | Освобождение регистра данных ST(i) |

## Обработка исключительных ситуаций

Исключение – внутреннее прерывание процессора, возникающее в ходе вычислительного процесса. В сопроцессоре может возникать 6 типов исключений. При возникновении какого-либо исключения устанавливается соответствующий бит в регистре состояния SWR.

Обработка исключения может выполняться двумя способами:

* обработка самим сопроцессором (маскированная обработка);
* программная обработка посредством реализации обработки возникающих прерываний (не замаскированная обработка).

Способ обработки зависит от значений соответствующих битов регистра управления сопроцессором CWR. Для каждого вида исключения в нем есть соответствующий бит. Если этот бит установлен в 1 (исключение маскировано), исключение обрабатывается самим сопроцессором. В противном случае (исключение не замаскировано) обработка должна быть выполнена программным путем.

Порядок маскированной обработки по видам исключений приведен в таблице 7.12.

Таблица 7.12

|  |  |
| --- | --- |
| Исключение | Действие |
| Недействительная операция | Возникает при работе со стеком или арифметическими операциями. Причины:   * загрузка операнда в непустой регистр стека; * попытка извлечь операнд из пустого регистра; * использование операнда с недопустимым значением.   В SWR устанавливается флаг IF. Маскируется битом IM в CWR. Если в SWR флаг SF=1, возникло при работе со стеком, SF=0 – неверный операнд.  Маскированная обработка – в регистр стека записывается тихое не-число. |
| Деление на ноль | Возникает в командах деления. В SWR устанавливается флаг ZE. Маскируется битом ZM в CWR.  Маскированная реакция – формирование результата в виде знаковой бесконечности. |
| Денормализованный операнд | Возникает при попытке выполнить операцию с денормализованным операндом. В SWR устанавливается флаг DE. Маскируется битом DM в CWR.  Маскированная реакция – только установка флага DE. |
| Переполнение | Возникает если порядок числа слишком велик для формата приемника. В SWR устанавливается флаг OE. Маскируется битом OM в CWR.  Маскированная реакция – формирование максимального представимого значения или знаковой бесконечности. |
| Антипереполнение | Возникает если порядок числа слишком мал для формата приемника. В SWR устанавливается флаг UE. Маскируется битом UM в CWR.  Маскированная реакция – формирование минимального представимого. |
| Неточный результат | Возникает когда результат невозможно точно представить в формате приемника и его нужно округлять. В SWR устанавливается флаг PE. Маскируется битом PM в CWR. Характер выполненного округления показывает бит C1:   * C1=0 – результат усечен; * C1=1 – результат округлен в большую сторону.   Маскированная реакция – округление числа. |

При немаскированной обработке появление ошибки в сопроцессоре вызывает прерывания 10h и 75h. Программист должен написать обработчик одного из этих прерываний, выполняющий коррекцию возникшей ошибки.

## Пример программы с использованием сопроцессора

Программа, которая вводит вещественное число в формате числа с фиксированной точкой, преобразует его во внутренний формат сопроцессора, и выполняет обратное преобразование из внутреннего формата сопроцессора в строку символов.

model small

.486

stack 100h

.data

ent db 0Ah, 0Dh, 'Enter: $'

rsl db 0Ah, 0Dh, 'Result: $'

opre db 0Ah, 0Dh, 'Operation: $'

opr db ?

inf db 'Inf$'

nan db 'NaN$'

n1 db 40

len1 db ?

n1d db 40 dup(?)

n2 db 40

len2 db ?

n2d db 40 dup(?)

result1 dq ?

result2 dq ?

result dq ?

resstr db 40 dup(?)

ext db 0Ah, 0Dh, 'Exit - Esc, continue - any other key...$'

.code

startupcode

finit

lea dx, ent + 2

jmp \_wrtk

\_begin:

lea dx, ent

\_wrtk: mov ah, 09h

int 21h

lea dx, n1

mov ah, 0ah

int 21h

lea dx, opre

mov ah, 09h

int 21h

mov ah, 01h

int 21h

mov opr, al

lea dx, ent

mov ah, 09h

int 21h

lea dx, n2

mov ah, 0ah

int 21h

push offset result1

push offset n1d

call StrToDouble

add sp, 2

push offset result2

push offset n2d

call StrToDouble

add sp, 2

lea dx, rsl

mov ah, 09h

int 21h

cmp opr, '+'

je \_add

cmp opr, '-'

je \_sub

cmp opr, '\*'

je \_mul

cmp opr, '/'

je \_div

jmp \_end

\_add:

fld result1

fld result2

faddp st(1), st(0) ; fadd и вытолкнуть после выполнения st(0)

fstp result

jmp \_write

\_sub:

fld result1

fld result2

fsubp st(1), st(0)

fstp result

jmp \_write

\_mul:

fld result1

fld result2

fmulp st(1), st(0)

fstp result

jmp \_write

\_div:

fld result1

fld result2

fdivp st(1), st(0)

fstsw ax

test ax, 100b

je \_nr

lea dx, inf

jmp \_wrtn

\_nr: test ax, 1

je \_nk

lea dx, nan

jmp \_wrtn

\_nk: fstp result

\_write:

push offset resstr

push offset result

call DoubleToStr

add sp, 2

lea dx, resstr

\_wrtn: mov ah, 09h

int 21h

\_end:

lea dx, ext

mov ah, 09h

int 21h

mov ah, 08h

int 21h

cmp al, 27

jnz \_begin

exitcode 0

DoubleToStr proc near

push bp

mov bp, sp

sub sp, 4 ; выделяем 4 байта в стеке

push ax bx dx cx di

pushf

fnstcw [bp - 4] ; сохраним значение регистра управления

fnstcw [bp - 2]

and word ptr [bp - 2], 1111001111111111b; биты 11-10 управление округлением, 11 - к нулю

or word ptr [bp - 2], 0000110000000000b

fldcw [bp - 2] ; Запись нового значения регистра управления

mov bx, [bp + 4]

fld qword ptr[bx] ; заталкиваем в стек сопроцессора число

ftst

fstsw ax

and ah, 1

cmp ah, 1

jne @@NBE

mov bx, [bp + 6]

mov byte ptr[bx], '-'

inc word ptr[bp + 6]

@@NBE: fabs

fst st(1)

fst st(2)

frndint

fsub st(2), st(0)

mov word ptr[bp - 2], 10

fild word ptr[bp - 2]

fxch st(1)

xor cx, cx

@@BG: fprem

fist word ptr [bp - 2]

push word ptr [bp - 2]

fxch st(2)

fdiv st(0), st(1)

frndint

fst st(2)

inc cx

ftst ; сравнить st(0) c 0

fstsw ax ; SR -> AX

sahf ; AH в флаги

jnz @@BG ; если 14 бит SR == 0 (6 бит AH) (если zf == 0 прыжок)

mov ax, cx

mov bx, [bp + 6]

@@BFG: pop dx

add dx, '0'

mov byte ptr[bx], dl

inc bx

loop @@BFG

fxch st(3)

fst st(2)

ftst

fstsw ax

sahf

jz @@CNE

mov byte ptr[bx], '.'

mov cx, 16

@@BFR: fmul st(0), st(1)

fst st(2)

frndint

fsub st(2), st(0)

fist word ptr [bp - 2]

fxch st(2)

mov ax, [bp - 2]

add ax, '0'

inc bx

mov byte ptr[bx], al

loop @@BFR

@@NIL: cmp byte ptr[bx], '0'

jne @@CNR

dec bx

jmp @@NIL

@@CNR: inc bx

@@CNE: mov byte ptr[bx], '$'

fstp st(0)

fstp st(0)

fstp st(0)

fstp st(0)

fldcw [bp - 4] ; восстановим настройки сопроцессора

popf

pop di cx dx bx ax

add sp, 4

pop bp

ret

DoubleToStr endp

StrToDouble proc near

push bp

mov bp, sp

sub sp, 2 ; выделяем 2 байта в стеке

push ax bx dx cx di

pushf

mov word ptr[bp - 2], 10 ; помещаем в выделенные 2 байта 10

fild word ptr[bp - 2] ; заталкиваем в стек сопроцессора 10

fldz ; заталкиваем в стек сопроцессора 0

mov di, 0

mov bx, [bp + 4] ; помещаем в bx адрес из стека

cmp byte ptr[bx], '-'

jne @@BPN

inc bx

mov di, 1

@@BPN: movsx ax, byte ptr [bx]

cmp ax, '.'

je @@PNT1

cmp ax, 0dh

jne @@CNT

fxch st(1)

fstp st(0)

jmp @@REN

@@CNT: sub ax, '0'

mov word ptr[bp - 2], ax

fmul st(0), st(1) ; умножаем число на вершине стека на 10

fiadd word ptr[bp - 2] ; добавляем к числу на вершине стека то что было в ax

inc bx

jmp @@BPN

@@PNT1:

xor cx, cx

@@BEG: inc bx

movsx ax, byte ptr [bx]

cmp ax, 0dh

je @@END

loop @@BEG

@@END: dec bx

fxch st(1)

fldz

@@APN: movsx ax, [bx]

cmp ax, '.'

je @@PNT2

sub ax, '0'

mov word ptr[bp - 2], ax

fiadd word ptr[bp - 2]

fdiv st(0), st(1)

dec bx

jmp @@APN

@@PNT2:

;

fxch st(1) ; меняем число 10 и остаток местами

fstp st(0) ; выталкиваем 10

faddp st(1) ; складываем целую и дробную части

@@REN:

cmp di, 1

jne @@CYK

fchs

@@CYK: mov bx, [bp + 6] ; помещаем в bx адрес из стека

fstp qword ptr [bx] ; помещаем по адресу из стека число

popf

pop di cx dx bx ax

add sp, 2

pop bp

ret

StrToDouble endp

end

## Задания для самостоятельного выполнения

1. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3+5x2-4x-20 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать короткий формат вещественных чисел.

2. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-7x2-9x+49 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать двойной формат вещественных чисел.

3. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3+7x2-16x-112 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать расширенный формат вещественных чисел.

4. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3+8x2+x-42 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать короткий формат вещественных чисел.

5. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-8x2+x+42 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать двойной формат вещественных чисел.

6. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3+x2-12x меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать расширенный формат вещественных чисел.

7. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-16x меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать короткий формат вещественных чисел.

8. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-6x2-16x+96 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать двойной формат вещественных чисел.

9. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-x2-25x+25 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать расширенный формат вещественных чисел.

10. Ввести вещественные значения a, b (a<b) и e. Если на отрезке [a; b] функция f(x)=x3-7x2+4x+12 меняет знак на противоположный, найти ее корень на этом отрезке методом деления отрезка пополам с точностью e. В противном случае вывести сообщение. Использовать короткий формат вещественных чисел.