Лабораторная работа №5

Команды сопроцессора

Цель работы: изучение команд сопроцессора для выполнения арифметических операций.

**Теоретические сведения**

Сопроцессор представляет собой отдельный функциональный модуль центрального процессора, предназначенный для выполнения операций над вещественными числами, а также для вычисления значений основных трансцендентных функций. Сопроцессор содержит следующие регистры:

1. Восемь регистров R0…R7, каждый из которых имеет размерность 80 битов, которые образуют стек сопроцессора.
2. Регистр состояния процессора SWR (Status Word Register) содержит информацию о текущем состоянии сопроцессора: какой регистр в данный момент является вершиной стека, какие возникли исключения и т.д.
3. Управляющий регистр сопроцессора CWR (Control Word Register) управляет режимами работы сопроцессора: точностью вычислений, погрешностью округлений, маскировкой исключений.
4. Регистр слова тегов (Tags Word Register) используется для контроля за состоянием каждого из регистров R0…R7.

Структура регистров сопроцессора больше похожа на кольцо, чем на стек. Номер текущей вершины этого кольца содержится в поле TOP регистра SWR. Текущая вершина стека имеет логический номер ST(0). Следующий регистр имеет номер ST(1) и так далее до ST(7). При загрузке числа в сопроцессор, оно попадает на вершину стека в регистр ST(0). Все числа в стеке при этом сдвигаются: число, которое до этого было на месте ST(i) перемещается на ST(i+1), значение поля TOP регистра SWR уменьшается на единицу.

R0

R1

R2

R3

R4

R5

R6

R7

ST(0)

ST(1)

ST(2)

ST(3)

ST(4)

ST(5)

ST(6)

ST(7)

Физические

номера регистров

Логические

номера регистров

Вершина

стека

TOP

SWR

TOP=000

R0

R1

R2

R3

R4

R5

R6

R7

ST(5)

ST(6)

ST(7)

ST(0)

ST(1)

ST(2)

ST(3)

ST(4)

Физические

номера регистров

Логические

номера регистров

Вершина

стека

TOP

SWR

TOP=011

Система команд сопроцессора включает около 80 команд. Для того, чтобы было легче ориентироваться в назначении команд необходимо запомнить следующие правила:

Название команд сопроцессора начинается с буквы F (Float);

Вторая буква обозначает тип операндов, с которыми работает команда: I – целые числа, B – двоично-десятичные числа (BCD-числа). Если вторая буква не является буквой B или I, то команда работает с вещественными числами;

Если команда заканчивается буквой P, это означает, что поле TOP увеличивается на единицу и содержимое ST(0) выталкивается из стека. Во многих командах сопроцессора вершина стека ST(0) адресуется неявно.

Сопроцессор поддерживает работу с вещественными числами следующего вида:

Команды передачи данных

|  |  |  |
| --- | --- | --- |
| Вещественные числа | | |
| FLD <операнд> | TOP=TOP-1; ST(0)=операнд | Загрузка вещественного числа размером 32/64/80 битов на вершину стека |
| FLD ST(i) | TOP=TOP-1; ST(0)=ST(i+1) | Загрузка копии числа из ячейки ST(i) на вершину стека ST(0) |
| FST <операнд> | операнд=ST(0) | Сохранение вещественного числа из вершины стека в память |
| FSTP <операнд> | операнд=ST(0); TOP=TOP+1 | Сохранение вещественного числа из вершины стека в память c выталкиванием его из стека |
| Целые числа | | |
| FILD <операнд> | TOP=TOP-1; ST(0)=операнд | Загрузка целого числа размером 16/32/64 битов на вершину стека |
| FIST <операнд> | операнд=ST(0) | Сохранение целого числа из вершины стека в память |
| FISTP <операнд> | операнд=ST(0); TOP=TOP+1 | Сохранение целого числа из вершины стека в память c выталкиванием его из стека |
| Десятичные числа | | |
| FBLD <операнд> | TOP=TOP-1; ST(0)=операнд | Загрузка десятичного числа в формате BCD размером 10 байт на вершину стека |
| FBSTP <операнд> | операнд=ST(0); TOP=TOP+1 | Сохранение десятичного числа из вершины стека в память c выталкиванием его из стека |

Следующий пример демонстрирует загрузку чисел разного типа в стек.

.data

x dd 5.5

y dq 15.7

z dt 4.0

i dd 23

n dq 15700

b dt 15201700h

.code

start:

fld x ; Загрузка вещественного числа типа float размером 4 байта

fld y ; Загрузка вещественного числа типа double размером 8 байт

fld z ; Загрузка вещественного числа типа long double размером 10 байт

fild i ; Загрузка 4-байтового целого числа типа int

fild n ; Загрузка 8-байтового целого числа типа long

fbld b ; Загрузка десятичного числа 15201700 размером 10 байт

|  |  |  |  |
| --- | --- | --- | --- |
| Вершина стека ---> | ST(0) | 15201700.000000000000 | b |
|  | ST(1) | 15700.000000000000000 | n |
|  | ST(2) | 23.000000000000000000 | i |
|  | ST(3) | 4.0000000000000000000 | z |
|  | ST(4) | 15.700000000000000000 | y |
|  | ST(5) | 5.5000000000000000000 | x |

Выгрузка чисел из сопроцессора в память осуществляется аналогично. Иногда возникает необходимость выгрузить число в регистр. Напрямую это сделать нельзя, но как промежуточный буфер можно использовать стек.

**sub** **ESP,** 4 ; Выделение памяти в стеке

**fstp** **dword** ptr **[ESP]** ; Вытолкнуть число типа float на вершину стека

**MOV** **EAX,** **[ESP]** ; Скопировать в регистр

**add** **ESP,** 4 ; Вернуть стек в исходное состояние

Чтобы избавиться от числа на вершине стека, можно использовать следующие две команды:

**ffree** **ST(**0**)**

**fincstp**

Команда ffree помечает регистр как пустой, а команда fincstp увеличивает указатель на вершину стека, тем самым выталкивая значение из стека.

Команды сравнения данных

Команды сравнения данных FCOM/FCOMP/FCOMPP, в зависимости от результата выполненной операции, изменяют флаги C3, C2, C1, C0 регистра SWR. Чтобы организовать в программе ветвления по результатам выполнения команд сравнения, необходимо переслать значения флагов C3…C0 регистра SWR в регистр флагов центрального процессора EFLAGS/FLAGS. Для этого существует специальная команда FSTSW, которая позволяет запомнить содержимое SWR в регистре AX или ячейке памяти. Далее, используя команду SAHF, нужно загрузить AH в младший байт регистра флагов. После этого бит C0 запишется на место флага CF, С2 – на место PF, С3 – на место ZF. Таким образом, результат выполнения команды сопроцессора окажется в регистре флагов EFLAGS/FLAGS и можно использовать команды условного перехода для создания разветвляющихся алгоритмов.

|  |  |  |  |
| --- | --- | --- | --- |
| Команды для работы с вещественными числами с модификацией C3…C0 | | | |
| FCOM | ST(0) | ST(1) | Сравнение содержимого регистров ST(0) и ST(1) |
| FCOM <операнд> | ST(0) | операнд | Сравнение содержимого регистра ST(0) с операндом в памяти |
| FCOMP  <операнд> | ST(0) | операнд | Сравнение содержимого регистра ST(0) с операндом в памяти и выталкивание значения из ST(0). TOP=TOP+1 |
| FCOMPP | ST(0) | операнд | Сравнение значений ST(0) и ST(1) и выталкивание значений обоих регистров из стека. TOP=TOP+2 |
| Команды для работы с вещественными числами с модификацией EFLAGS | | | |
| FCOMI  DB F0+i | ST(0) | ST(i) | Сравнение содержимого регистров ST(0) и ST(i) |
| FCOMIP  DF F0+i | ST(0) | ST(i) | Сравнение содержимого регистров ST(0) и ST(i) и выталкивание значения из ST(0). TOP=TOP+1 |
| Команды для работы с целыми числами | | | |
| FICOM <операнд> | ST(0) | операнд | Сравнение значения в ST(0) с целочисленным операндом размером 16/32 бита в памяти |
| FICOMP <операнд> | ST(0) | операнд | Сравнение значения в ST(0) с целочисленным операндом размером 16/32 бита в памяти и выталкивание ST(0) из стека. TOP=TOP+1 |
| FTST | ST(0) | 0 | Сравнение значения в ST(0) с нулём |
|  |  |  |  |

В следующей таблице показано, какие флаги устанавливают команды FCOM/FCOMP/FCOMPP в зависимости от результата сравнения операндов.

|  |  |  |  |
| --- | --- | --- | --- |
| **Условие** | **С3 (ZF)** | **С2 (PF)** | **C0 (CF)** |
| ST(0) > операнд | 0 | 0 | 0 |
| ST(0) < операнд | 0 | 0 | 1 |
| ST(0) = операнд | 1 | 0 | 0 |
| Операнды не сравнимы (#IA) | 1 | 1 | 1 |

Для сравнения вещественных чисел удобнее использовать команды FCOMI/FCOMIP, так как эти команды напрямую модифицируют EFLAGS. Но они поддерживаются не всеми архитектурами. Для того, чтобы использовать команду FCOMI ST(0), ST(i), нужно в тексте программы указывать её код – "db 0dbh, 0f0h+i".

Условия переходов командами FCOMI/FCOMIP:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Условие** | **ZF** | **PF** | **CF** | **Переход** |
| ST(0) > ST(i) | 0 | 0 | 0 | ja |
| ST(0) < ST(i) | 0 | 0 | 1 | jb |
| ST(0) = ST(i) | 1 | 0 | 0 | je |
| ST(0) ≥ ST(i) | \* | 0 | 0 | jae |
| ST(0) ≤ ST(i) | \* | 0 | \* | jbe |
| Операнды не сравнимы (#IA) | 1 | 1 | 1 |  |

|  |  |
| --- | --- |
| Пример использования разных способов сравнения чисел. | |
| fcom | fcomi |
| fld x  fld y  fcomp  fstsw AX  sahf  jc j1 ; y < x  jnc j2 ; y ≥ x  … | fld x  fld y  db 0dbh, 0f0h+1 ; Код команды fcomi  ST(0), ST(1)  ja j1 ; y > x  … |

Арифметические команды

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Команды для работы с целыми числами | | | | |
| FIADD  <операнд> | ST(0) | операнд | Сложение значения в ST(0) с 16/32-разрядным операндом в памяти. ST(0)=ST(0) + операнд |  |
| FISUB  <операнд> | ST(0) | операнд | Вычитание из ST(0) значения 16/32-разрядного операнда, находящегося в памяти.  ST(0)=ST(0)-операнд |  |
| FIMUL  <операнд> | ST(0) | операнд | Умножение значения на вершине стека ST(0) на значение 16/32-разрядного операнда, расположенного в памяти. ST(0)=ST(0)\*операнд |  |
| FIDIV  <операнд> | ST(0) | операнд | Деление содержимого ST(0) на значение целочисленного операнда размером 16/32 бита в памяти. ST(0)/ST(0)\*операнд |  |
| FISUBR <операнд> | ST(0) | операнд | Вычитание значения ST(0) из 16/32-разрядного операнда, расположенного в памяти. Результат вычитания помещается в ST(0).  ST(0) = операнд-ST(0) |  |
| FIDIVR <операнд> | ST(0) | операнд | Деление значения 16/32-разрядного операнда, расположенного в памяти, на содержимое ST(0). Результат деления помещается в ST(0).  ST(0) = операнд/ST(0) |  |
| Команды для работы с вещественными числами | | | | |
| FADD | ST(0) | ST(1) | Сложение двух чисел, расположенных на вершине стека. Результат помещается в ST(0).  ST(0)=ST(0)+ST(1) |  |
| FADD <операнд> | ST(0) | операнд | Сложение значения в ST(0) с операндом, расположенным в памяти.  ST(0)=ST(0)+операнд |  |
| FADD  ST(i), ST(0) | ST(i) | ST(0) | Сложение содержимого регистра ST(i) со значением, распложенным на вершине стека ST(0). Результат помещается в ST(i). ST(i)=ST(i)+ST(0) |  |
| FADDP  ST(i), ST(0) | ST(i) | ST(0) | Сложение ST(i) со значением, расположенным на вершине стека ST(0) и выталкивание из стека значения ST(0). Из-за выталкивания вершина стека смещается и результат попадает в ST(i-1).  ST(i)=ST(i)+ST(0). TOP=TOP+1 |  |
| FSUB | ST(0) | ST(1) | Вычитание из ST(0) значения, расположенного в ST(1). Результат помещается в ST(0).  ST(0)=ST(0) – ST(1) |  |
| FSUB  <операнд> | ST(0) | ST(1) | Вычитание из ST(0) значения, расположенного в памяти. Результат помещается в ST(0).  ST(0)=ST(0)-операнд |  |
| FSUB  ST(i), ST(0) | ST(i) | ST(0) | Вычитание из значения в ST(i) значения, расположенного в вершине стека сопроцессора. Результат помещается в ST(i).  ST(i)=ST(i)-ST(0) |  |
| FSUB  ST(0), ST(i) | ST(0) | ST(i) | Вычитание из значения на вершине стека ST(0) значения, расположенного в ST(i). Результат помещается в ST(0). ST(0)=ST(0)-ST(i) |  |
| FSUBP  ST(i), ST(0) | ST(i) | ST(0) | Вычитание из ST(i) значения, расположенного в вершине стека сопроцессора ST(0). Результат сначала помещается в ST(i), затем ST(0) выталкивается и результат перемещается в  ST(i-1). ST(i)=ST(i)-ST(0). TOP=TOP+1 |  |
| FSUBR  ST(i), ST(0) | ST(i) | ST(0) | Вычитание из значения в ST(0) значения, расположенного в ST(i). Результат помещается в ST(i). ST(i)=ST(0)-ST(i) |  |
| FSUBR  ST(0), ST(i) | ST(0) | ST(i) | Вычитание из значения в ST(i) значения, расположенного в вершине стека сопроцессора. Результат помещается в ST(0).  ST(0)=ST(i)-ST(0) |  |
| FSUBRP  ST(i), ST(0) | ST(i) | ST(0) | Вычитание из значения в ST(0) значения, расположенного в ST(i). Результат помещается в ST(i) и ST(0) выталкивается из стека.  ST(i)=ST(0)-ST(i). TOP=TOP+1 |  |
| FMUL | ST(0) | ST(1) | Умножение содержимого ST(0) на ST(1). Результат помещается в ST(0). ST(0)=ST(0)\*ST(1) |  |
| FMUL <операнд> | ST(0) | операнд | Умножение содержимого ST(0) на значение 32/64-разрядного операнда, расположенного в памяти. Результат помещается в ST(0).  ST(0)=ST(0)\*операнд. |  |
| FMUL  ST(0), ST(i) | ST(0) | ST(i) | Умножение содержимого вершины стека ST(0) на значение ST(i). Результат помещается в ST(0).  ST(0)=ST(0)\*ST(i) |  |
| FMUL  ST(i), ST(0) | ST(i) | ST(0) | Умножение значения ST(i) на значение содержимого вершины стека ST(0). Результат помещается в ST(i). ST(i)=ST(i)\*ST(i) |  |
| FMULP  ST(i), ST(0) | ST(i) | ST(0) | Умножение значения ST(i) на значение содержимого вершины стека ST(0). Результат помещается в ST(i). ST(0) выталкивается и результат попадает в ST(i-1).  ST(i)=ST(i)\*ST(0). TOP=TOP+1 |  |
| FMULP | ST(0) | ST(1) | Умножение значения ST(0) на значение, содержащееся на ST(1). Результат сначала помещается в ST(1), затем ST(0) выталкивается и результат помещается на вершину стека. ST(1)=ST(1)\*ST(0). TOP=TOP+1 |  |
| FDIV | ST(0) | ST(1) | Деление содержимого вершины стека ST(0) на значение, содержащееся в ST(1). Результат помещается в ST(0). ST(0)=ST(0)/ST(1). |  |
| FDIV  <операнд> | ST(0) | операнд | Деление значения ST(0) на вещественное число (операнд) размером 32/64 бита, расположенное в памяти. ST(0)=ST(0)/операнд |  |
| FDIV  ST(0), ST(i) | ST(0) | ST(i) | Деление содержимого вершины стека ST(0) на ST(i). Результат помещается в ST(0).  ST(0)=ST(0)/ST(i) |  |
| FDIV  ST(i), ST(0) | ST(i) | ST(0) | Деление содержимого ST(i) на значение в вершине стека ST(0). ST(i)=ST(i)/ST(0) |  |
| FDIVP ST(i),ST(0) | ST(i) | ST(0) | Деление содержимого ST(i) на значение в ST(0) с выталкиванием ST(0) из стека. Результат попадает в ST(i-1).  ST(i) = ST(i)/ST(0). TOP=TOP+1 |  |
| FDIVR <операнд> | ST(0) | операнд | Деление вещественного значения, располо-женного в памяти на значение, содержащееся на вершине стека ST(0). ST(0) = операнд / ST(0) |  |
| FDIVR  ST(0), ST(i) | ST(0) | ST(i) | Деление значения, содержащегося в ST(i) на значение, расположенное в вершине стека ST(0).  ST(0)=ST(i)/ST(0) |  |
| FDIVR  ST(i), ST(0) | ST(i) | ST(0) | Деление значения, расположенного на вершине стека ST(0) на значение ST(i). Результат помещается в ST(i). ST(i)=ST(0)/ST(i) |  |
| FDIVRP | ST(0) | ST(1) | Деление содержимого вершины стека ST(0) на значение в ST(1) с выталкиванием ST(0) из стека. Результат после этого попадает в ST(0).  ST(1) = ST(0)/ST(1). TOP=TOP+1 |  |
| FDIVRP  ST(i), ST(0) | ST(i) | ST(0) | Деление значения, расположенного на вершине стека ST(0) на значение ST(i). ST(0) выталкивается из стека. Результат попадает в ST(i-1). ST(i)=ST(0)/ST(i). TOP=TOP+1 |  |

Команды загрузки констант

|  |  |  |
| --- | --- | --- |
| Команда | Действие | Описание |
| FLDZ | TOP = TOP - 1; ST(0)=0 | Загрузка нуля |
| FLD1 | TOP = TOP - 1; ST(0)=1 | Загрузка единицы |
| FLDPI | TOP = TOP - 1; ST(0)=3.1415926535 | Загрузка π |
| FLDL2T | TOP = TOP - 1; ST(0)=3.3219280948 | Загрузка log210 |
| FLDL2E | TOP = TOP - 1; ST(0)=1.4426950408 | Загрузка log2e |
| FLDLG2 | TOP = TOP - 1; ST(0)=0.3010299956 | Загрузка lg 2 |
| FLDLN2 | TOP = TOP - 1; ST(0)=0.6931471805 | Загрузка ln 2 |

Дополнительные арифметические команды

|  |  |  |
| --- | --- | --- |
| Команда | Опе-ран-ды | Действие |
| FSQRT | ST(0) | Вычисление квадратного корня числа, содержащегося на вершине стека ST(0). Результат помещается в ST(0). |
| FABS | ST(0) | Вычисление модуля числа, находящегося на вершине стека ST(0). Результат помещается в ST(0). |
| FCHS | ST(0) | Изменение знака значения, находящегося на вершине стека ST(0). Результат помещается в ST(0). |
| FRNDINT | ST(0) | Округление до целого значения, расположенного в ST(0). Режим округления определяет поле CWR.RC:  если CWR.RC = 00b, то округление до ближайшего целого;  если CWR.RC = 01b, то округление до ближайшего меньшего целого;  если CWR.RC = 10b, то округление до ближайшего большего целого;  если CWR.RC = 11b, то округление с отбрасыванием дробной части числа.  Результат записывается в ST(0). Для сохранения/изменения полей управляющего регистра CWR нужно использовать команды FSTCWR и FLDCWR. |
| FSCALE | ST(0)  ST(0) | Масштабирование по степеням двойки числа, содержащегося в ST(0). Команда изменяет порядок значения, находящегося в вершине стека, на величину, в ST(1). ST(1) при этом округляется до ближайшего меньшего целого. Результат помещается в ST(0). |
| FPREM | ST(0)  ST(1) | Вычисление частичного остатка от деления содержимого вершины стека ST(0) на значение в ST(1).  ST(0)=ST(0)-Q\*ST(1), где Q – целое частное от деления.  К примеру, если ST(0)=10.6, ST(1)=3.5, то в результате в ST(0) помещается 0.1=10.6-3.5\*3 (Q=3). |
| FXTRACT | ST(0) | Выделение порядка и мантиссы числа, находящегося на вершине стека ST(0). Исходное значение раскладывается, порядок (целое число) помещается в ST(1), мантисса в ST(0). Результат удовлетворяет условию:  .  Пример. Если загрузить в ST(0) число 0.0044, то в результате выполнения команды FXTRACT будет  ST(0) = 1.1264, ST(1) = -8, т.е. 0.0044 = 1.1264 \* 2-8 |

Команды трансцендентных функций

|  |  |  |  |
| --- | --- | --- | --- |
| FCOS | ST(0) | Вычисление косинуса угла, выраженного в радианах, находящегося на вершине стека. Результат помещается в ST(0). ST(0)=cos(ST(0)) |  |
| FSIN | ST(0) | Вычисление синуса угла, выраженного в радианах, находящегося на вершине стека ST(0). Результат помещается в ST(0).  ST(0)=sin(ST(0)) |  |
| FSINCOS | ST(0) | Вычисление синуса и косинуса угла (в радианах), находящегося на вершине стека ST(0). Результат помещается в ST(0) и ST(1).  ST(0)=sin(ST(0)), ST(1)=cos(ST(0)) |  |
| FPTAN | ST(0) | Вычисление частичного тангенса угла, выраженного в радианах, находящегося на вершине стека ST(0). Значение угла может находиться в пределах 0≤α≤π/4. Результат записывается в два верхних регистра и представляется следующим образом:  ST(1) = tg(ST(0)), ST(0) = 1.0 |  |
| FPATAN | ST(0)  ST(1) | Вычисление арктангенса угла, представленного отношением в двух верхних регистрах стека сопроцессора. Результат сначала записывается в ST(1), затем из стека выталкивается один элемент и результат попадает в ST(0).  ST(1) = arctg(ST(1)/ST(0)); TOP=TOP+1 |  |
| F2XM1 | ST(0) | Вычисление значения выражения 2x-1 при -1<x<1. Результат помещается в ST(0).  ST(0)=2ST(0)-1. |  |
| FYL2X | ST(0)  ST(1) | Вычисление значения функции *z* = *y* log2(*x*).  Перед выполнением операции *x* размещается на вершине стека ST(0), *y* – в ST(1). Команда выталкивает из стека *x* и *y*. Результат после этого попадает в ST(0). |  |

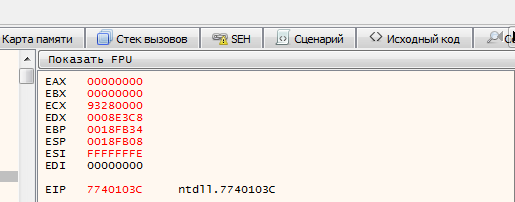
Для вычисления других тригонометрических функций нужно выражать их через имеющиеся в системе команд. Функции arcsin(*x*), arcos(*x*) выражаются через arctg(*x*):

Среди команд ассемблера также нет специальной команды для возведения произвольного числа *x* в степень *y*. Для *x* > 0 и *y* > 0 нужно использовать формулу:

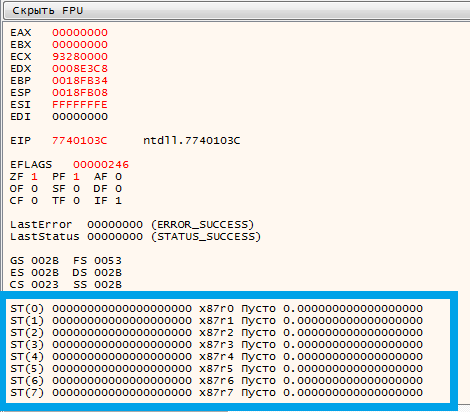
Для возведения числа 2 в произвольную степень также нет команды. Команда F2XM1 корректно считает только тогда, когда в ST(0) находиться число в интервале   
(-1, 1). Поэтому число *t* = *y*log2*x* нужно разбить на целую (*a*) и дробную части (*b*). Дробную часть можно получить в результате вычисления командой FPREM частичного остатка от деления *t* на единицу. Есть другой способ. Если сначала получить целую часть *a* с помощью округления до ближайшего меньшего целого, то *b* = *t* - *a*. Тогда

2*a* нужно вычислять помощью команды FSCALE. Целую часть можно специально не вычислять, так как команда FSCALE сама округляет число в ST(1) до ближайшего меньшего целого. 2*b* теперь можно вычислить командой F2XM1, но после выполнения этой команды к результату нужно прибавить единицу. Если *x* < 0, то нужно просто вынести знак: (-*x*)*y* = -*xy*. Если *y* < 0, то *x*-*y*= 1 / *xy*.

Для того, чтобы просматривать содержимое регистров сопроцессора ST(0)-ST(7) во время отладки программы в x32dgb, нужно над окном состояния регистров нажать на кнопку «Показать FPU»:



После этого можно будет увидеть значения этих регистров:



Печать вещественных чисел на консоль осуществляется функцией crt\_printf со спецификатором "%f" или "%lf".

|  |
| --- |
| Вывод числа типа double |
| x dq 7.5  double\_fmt db "%f"**,** 0  **push** **dword** ptr x**[**4**]**  **push** **dword** ptr x**[**0**]**  **push** offset double\_fmt  **call** crt\_printf  **add** **esp,** 12 |

При написании собственных подпрограмм будем придерживаться стандартного соглашения о вызовах. Если функция возвращает вещественное значение, то принято оставлять его на вершине стека сопроцессора в ST(0). Вещественные аргументы передаются через стек в обратном порядке также, как и целочисленные.

Если функция задействует определённое количество регистров сопроцессора и при этом эти регистры уже заняты в вызывающей подпрограмме, то можно временно сохранить их в стек, а потом после использования восстановить:

**sub** **ESP,** 2**\***10 ; Выделяем память для двух регистров

; Сохраняем два верхних регистра сопроцессора в стеке

**fstp** **TBYTE** PTR **[ESP]**

**fstp** **TBYTE** PTR **[ESP+**10**]**

;... Выполняем какие-то операции

; Восстанавливаем регистры сопроцессора в обратном порядке

**fld** **TBYTE** PTR **[ESP+**10**]**

**fld** **TBYTE** PTR **[ESP]**

**add** **esp,** 2**\***10 ; Освобождаем занимаемую память

Этот же приём можно использовать, если в нашей программе недостаточно регистров сопроцессора.

Напишем подпрограмму length(x, y), которая возвращает длину вектора по формуле . Функция length имеет следующий заголовок на языке Си:

double length(double x, double y)

Перед вызовом функции аргументы должны быть переданы через стек в обратном порядке.

|  |  |
| --- | --- |
| Вызов функции | Тело функции |
| .data  x dq 3.0  y dq 4.0  .code  **push** **dword** ptr y**[**4**]**  **push** **dword** ptr y**[**0**]**  **push** **dword** ptr x**[**4**]**  **push** **dword** ptr x**[**0**]**  **call** length2 | length2 proc  **fld** **qword** ptr **[ESP** **+** 4**]** ; ST(0) = x  **fld** **ST(**0**)** ; ST(0) = ST(1)  **fmulp** **ST(**1**),** **ST(**0**)** ; ST(0) = x\*x  **fld** **qword** ptr **[ESP** **+** 12**]**; ST(0) = y  **fld** **ST(**0**)** ; ST(0) = ST(1)  **fmulp** **ST(**1**),** **ST(**0**)** ; ST(0) = y\*y  **faddp** **ST(**1**),** **ST(**0**)** ; ST(0) = x\*x + y\*y  **fsqrt** ; ST(0) = sqrt(x\*x + y\*y)  **ret** 2**\***8 ; Удаление аргументов из стека и возврат  length2 endp |

Задания для выполнения к работе

1. Написать функцию pow (*x*, *y*) для возведения числа *x* в степень *y*. Числа *x*, *y* могут быть произвольными, в том числе отрицательными. Рассмотреть случаи, когда  
    *x* = 0 и/или *y* = 0. Аргументы передавать подпрограмме через стек. Если алгоритм требует выгрузки чисел из сопроцессора в память или регистры, использовать для этого стек. Подобрать набор тестовых данных для проверки работы функции pow (не менее 10). Убедиться в том, что результаты работы написанной функции pow и стандартной функции pow библиотеки math.h языка C. В отчёт включить текст программы, блок-схему алгоритма функции pow и набор тестовых данных.

Функция pow должна удовлетворять соглашениям о вызовах. Аргументы должны передаваться ей через стек. К примеру, если *x*, *y* находятся в ST(0), ST(1), то они должны быть переданы так:

**sub** **ESP,** 8\*2 ; Выделение в стеке памяти для двух чисел типа double

**fstp** **QWORD** PTR **[ESP]** ; Выгрузка x

**fstp** **QWORD** PTR **[ESP + 8]** ; Выгрузка y

1. Составить таблицу состояния регистров для всех вещественных операций: для начальных операций перед циклом, для первой и последней итераций, для завершающих операций после цикла. Пример составления таблицы приведён ниже.
2. Численно исследовать на сходимость ряд. Аргументы тригонометрических функций считать в радианах. Для возведения чисел в степень использовать написанную функцию pow. В отчёт включить текст программы и значения суммы ряда при *n* от 1 до 50. Вывести результат на экран в виде:

n = 1; S = …

n = 2; S = …

…

Убедиться, что результаты вычислений совпадают с аналогичной программой на языке Python.

**Пример выполнения работы**

Вычислим численно сумму ряда:

Определим следующий порядок следования переменных в регистрах сопроцессора. Сумму будем накапливать в самом нижнем регистре сопроцессора ST(i) (т.е. на дне кольца сопроцессора). Знак текущего слагаемого (sign, он же числитель) будем хранить в регистре ST(i-2), число – в ST(i-1).

Составим таблицу состояния регистров сопроцессора для первых двух итераций алгоритма. Колонки таблицы соответствуют регистрам, строки – командам. В ячейках записаны значения регистров после выполнения текущей команды, обозначенной во втором столбце. Номер текущей команды (№) – условный номер строки в программе, в которой она выполняется. Стрелкой (←) обозначены команды, которые выталкивают значения из стека. Оранжевым цветом показаны регистры, с которыми работает текущая команда, голубыми – регистры, которые не изменились текущей командой.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Команда** |  | **ST(0)** | **ST(1)** | **ST(2)** | **ST(3)** | **ST(4)** | **ST(5)** | **ST(6)** | **ST(7)** |
| Инициализация регистров: | | | | | | | | | | |
| 1 | fldz |  | 0 (S) |  |  |  |  |  |  |  |
| 2 | fld1 |  | 1 (n) | 0 (S) |  |  |  |  |  |  |
| 3 | fld1 |  | 1 (sign) | 1 (n) | 0 (S) |  |  |  |  |  |
| Первая итерация: | | | | | | | | | | |
| 4 | fld ST(0) |  | 1 (sign) | 1 (sign) | 1 (n) | 0 (S) |  |  |  |  |
| 5 | fdiv ST(0), ST(2) |  | 1 (1/n) | 1 (sign) | 1 (n) | 0 (S) |  |  |  |  |
| 6 | faddp ST(3), ST(0) | ← | 1 (sign) | 1 (n) | 1 (S) |  |  |  |  |  |
| 7 | fld1 |  | 1 | 1 (sign) | 1 (n) | 1 (S) |  |  |  |  |
| 8 | faddp ST(2), ST(0) | ← | 1 (sign) | 2 (n) | 1 (S) |  |  |  |  |  |
| 9 | fchs |  | -1 (sign) | 2 (n) | 1 (S) |  |  |  |  |  |
| Вторая итерация: | | | | | | | | | | |
| 4 | fld ST(0) |  | -1 (sign) | -1 (sign) | 2 (n) | 1 (S) |  |  |  |  |
| 5 | fdiv ST(0), ST(2) |  | -1/2 (1/n) | -1 (sign) | 2 (n) | 1 (S) |  |  |  |  |
| 6 | faddp ST(3), ST(0) | ← | -1 (sign) | 2 (n) | 1/2 (S) |  |  |  |  |  |
| 7 | fld1 |  | 1 | -1 (sign) | 2 (n) | 1/2 (S) |  |  |  |  |
| 8 | faddp ST(2), ST(0) | ← | -1 (sign) | 3 (n) | 1/2 (S) |  |  |  |  |  |
| 9 | fchs |  | 1 (sign) | 3 (n) | 1/2 (S) |  |  |  |  |  |
|  | | | | | | | | | | |
| Последняя итерация (*n*=1000) | | | | | | | | | | |
| 4 | fld ST(0) |  | -1 (sign) | -1 (sign) | 1000 (n) | 0.693647 (S) |  |  |  |  |
| 5 | fdiv ST(0), ST(2) |  | -1/1000 (1/n) | -1 (sign) | 1000 (n) | 0.693647 (S) |  |  |  |  |
| 6 | faddp ST(3), ST(0) | ← | -1 (sign) | 1000 (n) | 0.692647 (S) |  |  |  |  |  |
| 7 | fld1 |  | 1 | -1 (sign) | 1000 (n) | 0.692647 (S) |  |  |  |  |
| 8 | faddp ST(2), ST(0) | ← | -1 (sign) | 1001 (n) | 0.692647 (S) |  |  |  |  |  |
| 9 | fchs |  | 1 (sign) | 1001 (n) | 0.692647 (S) |  |  |  |  |  |
| Заключительные операции: | | | | | | | | | | |
| 10 | ffree ST(0) |  |  | 1001 (n) | 0.692647 (S) |  |  |  |  |  |
| 11 | fincstp | ← | 1001 (n) | 0.692647 (S) |  |  |  |  |  |  |
| 12 | ffree ST(0) |  |  | 0.692647 (S) |  |  |  |  |  |  |
| 13 | fincstp | ← | 0.692647 (S) |  |  |  |  |  |  |  |
| 14 | fstp QWORD PTR [ESP] | ← |  |  |  |  |  |  |  |  |

Текст программы для вычисления суммы ряда:

.386 ; Тип процессора

.model **flat,** **stdcall** ; Модель памяти и стиль вызова подпрограмм

option **casemap:** **none** ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include windows.inc

include kernel32.inc

include user32.inc

include msvcrt.inc

; --- Подключаемые библиотеки ---

includelib user32.lib

includelib kernel32.lib

includelib msvcrt.lib

; --- Сегмент данных ---

.data

fmt db "S = %lf"**,** 0

; --- Сегмент кода ---

.code

start**:**

**finit** ; Инициализация сопроцессора

**fldz** ;(1); S = 0

**fld1** ;(2); n = 1

**fld1** ;(3); sign = 1

**mov** **ECX,** 1000 ; ECX = n

j\_cycle**:**

**fld** **ST(**0**)** ;(4); ST(0) = sign

**fdiv** **ST(**0**),** **ST(**2**)** ;(5); ST(0) = sign / n

**faddp** **ST(**3**),** **ST(**0**)** ;(6); S = S + sign / n

**fld1** ;(7); ST(0) = 1

**faddp** **ST(**2**),** **ST(**0**)** ;(8); n = n + 1

**fchs** ;(9); sign = -sign

**LOOP** j\_cycle ; Переход на начало цикла

**ffree** **ST(**0**)** ;(10); Освобождение регистра, в котором хранился знак

**fincstp** ;(11); Выталкивание ненужного значения sign из стека

**ffree** **ST(**0**)** ;(12); Освобождение регистра, в котором хранилось число n

**fincstp** ;(13); Выталкивание ненужного значения n из стека

**sub** **ESP,** 8 ; Выделение в стеке памяти для числа типа double

**fstp** **QWORD** PTR **[ESP]** ;(14); Выталкивание числа из ST(0) в стек

**push** offset fmt ; Строка формата

**call** crt\_printf ; crt\_printf("S = %lf", ST0)

**add** **ESP,** 12 ; Удаление аргументов из стека

**push** NULL

**call** ExitProcess ; Выход из программы

end start

**Варианты заданий**

|  |  |
| --- | --- |
| Вариант | Выражение |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 24 |  |
| 25 |  |