3. ЛАБОРАТОРНАЯ РАБОТА № 3 «ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА»

Цель работы: изучение принципов выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

1. Математический сопроцессор

Сопроцессор дополняет возможности основного процессора и предназначен для выполнения следующих функций:

– поддержки арифметики с плавающей точкой;

– поддержки численных алгоритмов вычисления значений тригонометрических функций, логарифмов и т. п.;

– обработки десятичных чисел с точностью до 18 разрядов, что позволяет сопроцессору выполнять арифметические операции без округления над целыми десятичными числами со значениями до 10^{18};

– обработки вещественных чисел из диапазона $3{,}37 \cdot 10^{-4932}$ … $1.18 \cdot 10^{+4932}$.

С точки зрения программиста, сопроцессор представляет собой совокупность регистров, каждый из которых имеет свое функциональное назначение (рисунок 1.1). В программной модели сопроцессора можно выделить три группы регистров.

1. Восемь регистров `r0`,…,`r7`, составляющих основу программной модели сопроцессора – *стек сопроцессора*. Размерность каждого регистра – 80 битов.

2. Три служебных регистра:

– *регистр состояния сопроцессора* `swr` (Status Word Register – регистр слова состояния) – отражает информацию о текущем состоянии сопроцессора. В регистре `swr` содержатся поля, позволяющие определить: какой регистр является текущей вершиной стека сопроцессора, какие исключения возникли после выполнения последней команды, каковы особенности выполнения последней команды (некий аналог регистра флагов основного процессора) и т. д.;

– *управляющий регистр сопроцессора* `cwr` (Control Word Register – регистр слова управления) – управляет режимами работы сопроцессора. С помощью полей в этом регистре можно регулировать точность выполнения численных вычислений, управлять округлением, маскировать исключения;

– *регистр слова тегов* `twr` (Tags Word Register – слово тегов) – используется для контроля за состоянием каждого из регистров `r0`,…,`r7`. Команды сопроцессора используют этот регистр, например, для того чтобы определить возможность записи значений в эти регистры.

3. Два регистра указателей – данных dpr (Data Point Register) и команд ipr (Instruction Point Register). Они предназначены для запоминания информации об адресе команды, вызвавшей исключительную ситуацию, и адресе ее операнда. Эти указатели используются при обработке исключительных ситуаций (но не для всех команд).

Все эти регистры являются программно доступными.

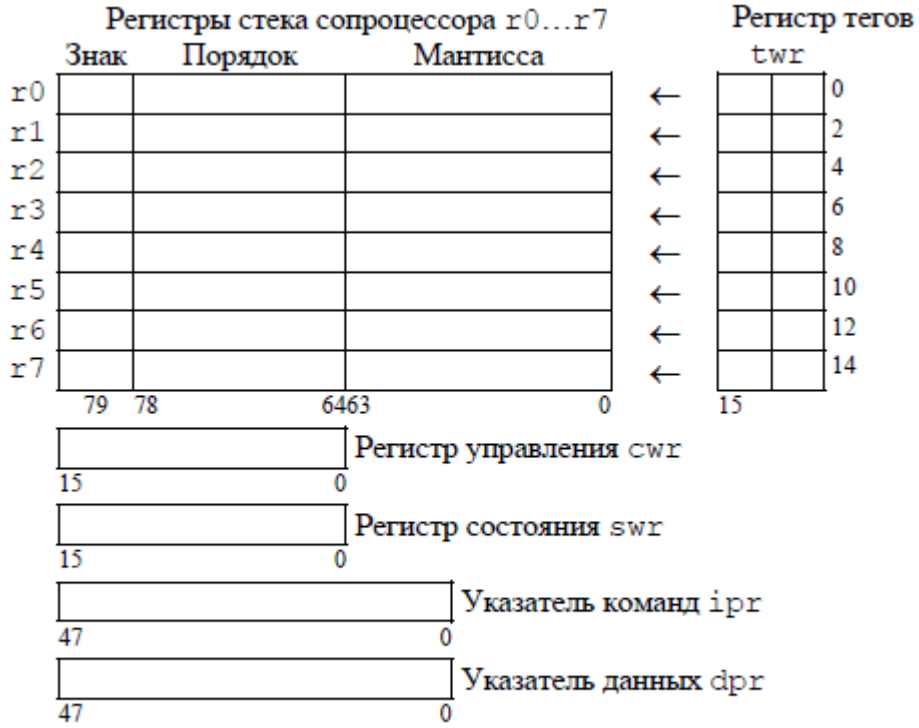

Рисунок 1.1 – Программная модель сопроцессора

Регистровый стек сопроцессора организован по принципу кольца. Это означает, что среди всех регистров, составляющих стек, нет такого, который является вершиной стека. Все регистры стека с функциональной точки зрения абсолютно одинаковы и равноправны. В стеке есть вершина, которая является плавающей. Контроль текущей вершины осуществляется аппаратно с помощью трехбитового поля top регистра swr (рисунок 1.2). В поле top фиксируется номер регистра стека 0...7 (r0,...,r7), являющегося в данный момент текущей вершиной стека.

b	c3	top			c2	c1	c0	es	sf	pe	ue	oe	ze	de	ie
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Рисунок 1.2 – Регистр состояния swr

Команды сопроцессора не оперируют физическими номерами регистров стека `r0...r7`. Вместо этого они используют логические номера этих регистров `st(0)...st(7)`. С помощью логических номеров реализуется относительная адресация регистров стека сопроцессора.

Сопроцессор расширяет номенклатуру форматов данных, с которыми работает основной процессор. Сопроцессор специально разрабатывался для вычислений с плавающей точкой. Но сопроцессор может работать и с целыми числами, хотя и менее эффективно. Форматы данных, с которыми работает сопроцессор:

– двоичные целые числа в трех форматах – 16, 32 и 64 бита;

– упакованные целые десятичные (BCD) числа – максимальная длина 18 упакованных десятичных цифр (9 байт);

– вещественные числа в трех форматах – коротком (32 бита), длинном (64 бита) и расширенном (80 бит).

Кроме этих основных форматов, сопроцессор поддерживает специальные численные значения, к которым относятся:

– денормализованные вещественные числа – это числа, меньшие минимального нормализованного числа для каждого вещественного формата, поддерживаемого сопроцессором;

– нуль;

– положительные и отрицательные значения бесконечность;

– нечисла;

– неопределенности и неподдерживаемые форматы.

В самом сопроцессоре числа в этих форматах имеют одинаковое внутреннее представление – в виде расширенного формата вещественного числа. Это один из форматов представления вещественных чисел, который точно соответствует формату регистров `r0...r7` стека сопроцессора.

Таким образом, даже если используются команды сопроцессора с целочисленными операндами, то после загрузки в сопроцессор операндов целого типа они автоматически преобразуются в формат расширенного вещественного числа.

Основной тип данных, с которыми работает сопроцессор, – вещественный. Данные этого типа описываются тремя форматами: коротким, длинным и расширенным (рисунок 1.6).

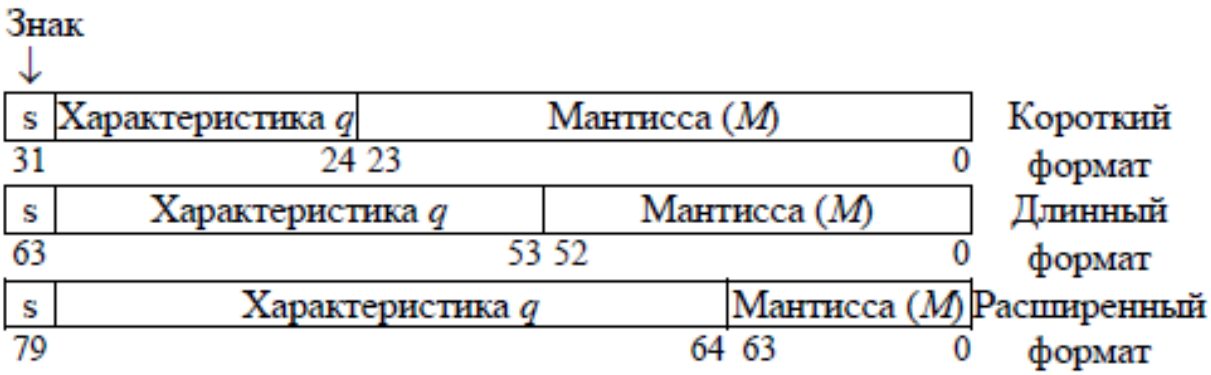

Рисунок 1.6 – Форматы вещественных чисел сопроцессора

Для представления вещественного числа используется формула:

$$A = (\pm M) \cdot N^{\pm(p)}, \qquad (1.1)$$

где M – мантисса числа A. Мантисса должна удовлетворять условию $|M| < 1$;

N – основание системы счисления, представленное целым положительным числом;

p – порядок числа, показывающий истинное положение точки в разрядах мантиссы (по этой причине вещественные числа имеют еще название чисел с плавающей точкой, так как ее положение в разрядах мантиссы зависит от значения порядка).

Для удобства обработки в компьютере чисел с плавающей точкой, архитектурой компьютера на компоненты формулы (1.1) накладываются следующие ограничения:

– основание системы счисления $N = 2$;

– мантисса M должна быть представлена в *нормализованном* виде. Для архитектуры микропроцессора Intel нормализованным является число вида:

$$A = (-1)^s \cdot N^q \cdot M, \qquad (1.2)$$

где s – значение знакового разряда (0 – число больше нуля; 1 – число меньше нуля);

p – порядок числа. Его значение аналогично значению порядка p в формуле (1.1).

В этой формуле знак имеют и порядок вещественного числа, и его мантисса. На рисунке 1.6 видно, что формат хранения вещественного числа в памяти имеет только поле для знака мантиссы. А где же хранится знак порядка? В сопроцессоре Intel на аппаратном уровне принято соглашение, что порядок p определяется в формате вещественного числа особым значением, называемым *характеристикой* q. Величина q связана с порядком p посредством формулы (1.3) и представляет собой

некоторую константу. Условно назовем ее *фиксированным смещением*:

$$q = p + \text{фиксированное смещение}. \qquad (1.3)$$

Для каждого из трех возможных форматов вещественных чисел смещение q имеет разное, но фиксированное для конкретного формата значение, которое зависит от количества разрядов, отводимых под характеристику (таблица 1.1).

Таблица 1.1 – Формат вещественных чисел

Формат	Короткий	Длинный	Расширенный
Длина числа (биты)	32	64	80
Размерность мантиссы M	24	53	64
Диапазон значений	$10^{-38}...10^{38}$	$10^{-308}...10^{308}$	$10^{-4932}...10^{4932}$
Размерность характеристики q	8	11	15
Значение фиксированного смещения	+127 (7F) 0111 1111	+1023 (3FF) 0011 1111 1111	+16383 (3FFF) 0011 1111 1111 1111
Диапазон характеристик q	0...255	0...2047	0...32767
Диапазон порядков p	–126...+127	–1022...+1023	–16382...+16383

В таблице 1.1 показаны диапазоны значений характеристик q и соответствующих им истинных порядков p вещественных чисел. Нулевому порядку вещественного числа в коротком формате соответствует значение характеристики равное 127, которому в двоичном представлении соответствует значение 0111 1111. Отрицательному порядку p, например, –1, будет соответствовать характеристика $q = -1 + 127 = 126$.

В двоичном виде ей соответствует значение 0111 1110. Положительному порядку p, например, +1, будет соответствовать характеристика $q = 1 + 127 = 128$, в двоичном виде ей соответствует значение 1000 0000. То есть, все положительные порядки имеют в двоичном представлении характеристики старший бит равный единице, а отрицательные порядки – нет. Знак порядка спрятан в старшем бите характеристики.

Так как нормализованное вещественное число всегда имеет целую единичную часть (исключая представление перечисленных выше специальных численных значений), то при его представлении в памяти появляется возможность считать первый разряд вещественного числа

единичным по умолчанию и учитывать его наличие только па аппаратном уровне. Это дает возможность увеличить диапазон представимых чисел, так как появляется лишний разряд, который можно использовать для представления мантиссы числа. Но это справедливо только для короткого и длинного форматов вещественных чисел. Расширенный формат, как внутренний формат представления числа любого типа в сопроцессоре, содержит целую единичную часть вещественного в явном виде.

Короткое вещественное число длиной в 32 разряда определяется директивой `dd`. При этом обязательным в записи числа является наличие десятичной точки, даже если оно не имеет дробной части. Для транслятора десятичная точка является указанием, что число нужно представить в виде числа с плавающей точкой в коротком формате. Это же касается длинного и расширенного форматов представления вещественных чисел, определяемых директивами `dq` и `dt`.

Другой способ задания вещественного числа директивами `dd`, `dq` и `dt` – экспоненциальная форма с использованием символа «e».

Пример. Определим в программе вещественное число 45,56 в коротком формате:

```
dd 45.56
```

или

```
dd 45.56e0
```

или

```
dd 0.4556e2
```

В памяти это число будет выглядеть так:

```
71 3d 36 42
```

Так как в архитектуре Intel принят перевернутый порядок следования байт в памяти в соответствии с принципом «младший байт по младшему адресу», истинное представление числа 45,56 будет следующим: `42 36 3d 71`. В двоичном представлении в памяти число будет иметь вид, представленный на рисунке 1.7. На рисунке 1.7 видно, что старшая единица мантиссы при представлении в памяти отсутствует.

Переведем десятичную дробь 45,56 в эквивалентное двоичное представление:

$45{,}56_{10} = 101101{,}10001111010111000 1\ldots_2$.

Нормализуем число. Для этого переносим точку влево, до тех пор, пока в целой части числа не останется одна двоичная единица. Число переносов влево (или вправо, если десятичное число было меньше единицы) будет являться порядком числа. После перемещения точки получаем значение порядка $p = 5$. Соответственно, характеристика бу-

дет выглядеть так: $q = p + 127 = 5 + 127 = 132_{10} = 1000\ 0100_2$.

Сформированный результат в виде вещественного числа в коротком формате состоит из трех компонент:

– знака – 0;

– характеристики 1000 0100;

– мантиссы 0110 1100 0111 1010 1110 001…, содержащей целую чисть числа 45 без первой значащей единицы (01101) и дробную часть числа (100 0111 1010 1110 001…).

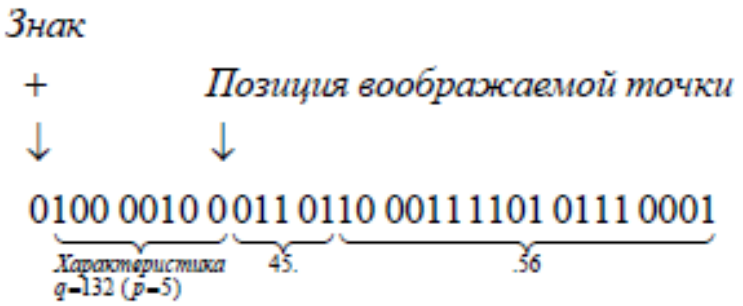

Рисунок 1.7 – Двоичное представление вещественного числа в коротком формате

Пример. Определим в программе вещественное число 45,56 в длинном формате:

```
dq 45.56
```

или

```
dq 45.56e0
```

В памяти это число будет выглядеть так:

```
47 e1 7a 14 ae c7 46 40
```

Перевернув его, получим истинное значение:

```
40 46 c7 ae 14 7a e1 47
```

Для представления числа 45,56 в регистрах сопроцессора переведем это число в двоичную систему счисления:

$45{,}56_{10} = 101101{,}1000111101011100001\ldots_2$.

Порядок числа $p = 5$.

Характеристика числа

$$q = p + 1023 = 5 + 1023 = 1028_{10} = 100\ 0000\ 0100_2.$$

Сформированный результат в виде вещественного числа в длинном формате состоит из трех компонент:

– знака – 0;

– характеристики 100 0000 0100;

– мантиссы 0110 1100 0111 1010 1110 001…, содержащей целую чисть числа 45 без первой значащей единицы (01101) и дробную часть числа (100 0111 1010 1110 001…).

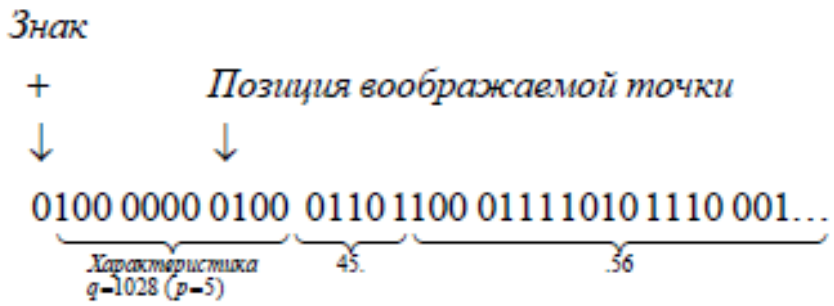

Рисунок 1.8 – Двоичное представление вещественного числа в длинном формате

Пример. Определим в программе вещественное число 45,56 в расширенном формате:

```
dt 45.56
```

В памяти это число будет выглядеть так:

```
71 3d 0a d7 a3 70 3d b6 04 40
```

Перевернув его, получим истинное значение в памяти:

```
40 04 b6 3d 70 a3 d7 0a 3d 71
```

Для представления числа 45,56 в регистрах сопроцессора переведем это число в двоичную систему счисления:

$45{,}56_{10} = 101101{,}1000111101011100001\ldots_2$.

Характеристика числа

$q = p + 16383 = 5 + 16383 = 16388_{10} = 100\ 0000\ 000\ 0100_2$.

Сформированный результат в виде вещественного числа в длинном формате состоит из трех компонент:

– знака – 0;

– характеристики 100 0000 0000 0100;

– мантиссы 1011 0110 0011 1101 0111 0001…, содержащей целую часть числа 45 с первой значащей единицей (101101) и дробную часть числа (100 0111 1010 1110 001…).

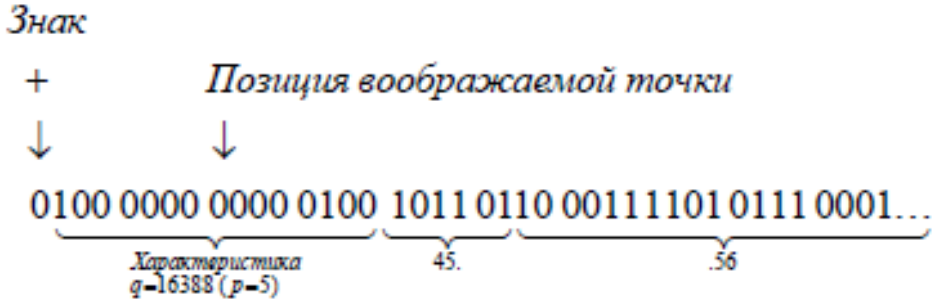

Рисунок 1.9 – Двоичное представление вещественного числа в расширенном формате

В мантиссе явно присутствует старшая единица, чего не было в коротком и длинном форматах представления вещественного числа.

Команды математического сопроцессора можно посмотреть в Приложении 1 или [1].

Мнемоническое обозначение команд сопроцессора характеризует особенности их работы:

1. Все мнемонические обозначения начинаются с символа `f` (float).

2. Вторая буква мнемонического обозначения определяет тип операнда в памяти, с которым работает команда:

`i` – целое двоичное число;

`b` – целое десятичное число;

отсутствие буквы – вещественное число.

3. Последняя буква `p` в мнемоническом обозначении команды означает, что последним действием команды обязательно является извлечение операнда из стека.

4. Последняя или предпоследняя буква `r` (reversed) в мнемоническом обозначении команды означает реверсивное следование операндов при выполнении команд вычитания и деления, так как для них важен порядок следования операндов.

Система команд сопроцессора отличается большой гибкостью в выборе вариантов задания команд, реализующих определенную операцию, и их операндов.

Минимальная длина команды сопроцессора – 2 байта.

Методика написания программ для сопроцессора имеет свои особенности. Главная причина – в стековой организации сопроцессора.

При разработке программ необходимо учитывать:

– ограниченность глубины стека сопроцессора;

– несовпадение форматов операндов;

– отсутствие поддержки на уровне команд сопроцессора некоторых операций, таких как возведение в степень, вычисление тригонометрических функций.

2. Индивидуальное задание

Создайте новый проект в Visual Studio (См. мет. указания к лабораторной работе №1). Для просмотра содержимого регистров сопроцессора в окне «Регистры» дополнительно установите галочку «С плавающей запятой».

Напишите программу на ассемблере, реализующую решение задачи в соответствие с вариантом. В вычислениях использовать команды математического сопроцессора.

Вариант	Задача
1	В массиве 16-разрядных чисел со знаком заданы координаты точек треугольника. Проверить являются ли каждые три точки координатами треугольника.
2	Разработать алгоритм и написать программу вычисления площади треугольников по формуле Герона. Треугольники заданы в виде массива тремя сторонами как 32-х разрядные числа.
3	Вычислить и сохранить в массиве значения функции f(x) (100 элементов) на интервале от A до B, если функция задается как: x^2, если x>0; x, если -1=<x<=0; $2\sqrt{(x-6)}$, если x<-1
4	Область на координатной плоскости задается уравнением эллипса. Для массива точек определить принадлежат ли они к внутренней или внешней области. Результат поместить в массив: 1 - вне фигуры, 0 - внутри.
5	Отсортировать массив (N элементов) 16-ти разрядных чисел и найти сумму чисел, которые делятся на 11.
6	Для двумерного массива (N M) определить сумму чисел в нечетных столбцах и нечетных строках. Числа в массиве целые без знака 32-разрядные.
7	Определить является ли строка символов строкой представления действительного числа со знаком (например - 123.67). Знак числа может отсутствовать, знак точки обязателен, количество цифр неограниченно.
8	Построить массив из 70 16-ти разрядных чисел со знаком, если первые два числа равны 1, а следующие являются суммой двух предшествующих и, для каждого третьего знак отрицательный.

9	В массиве 16 разрядных чисел без знака найти сумму чисел с кодовой двоичной комбинацией 1101. Число элементов массива задается в программе.
10	В массиве заданы коэффициенты квадратных уравнений - A, B, C. Определить какое из уравнений имеет два корня, один корень и не имеет корней.
11	Прямые заданы уравнением Y=kX+b. Определить есть ли перпендикулярные прямые, и их общее количество.
12	Для массива N x N 32 разрядных чисел. Составить новую матрицу синусов и косинусов.
13	Для матрицы N x M найти номер строки, сумма элементов которой является минимальной. Элементы массива 16-разрядные числа со знаком.
14	В массиве чисел (N x N) переставить элементы диагоналей, если сумма элементов главной диагонали делится на 2 или на 5. Числа в массиве 32-разрядные без знака.
15	На плоскости задано три точки, каждая из которых задана парой координат (x, y). Определить задают ли эти точки прямоугольный треугольник. Координаты точек - действительные числа.
16	Линии на плоскости задаются двумя точками T1, T2. Определить какие из них пересекаются. Координаты точек 64-разрядные вещественные числа.
17	Отсортировать массив (N) прямоугольников заданных сторонами A, B по площади. Стороны - действительные числа.
18	Заданы: начальная скорость и ускорение автомобиля, получить пройденный путь автомобилем через заданные промежутки времени (промежутки заданы в виде массива).

3. Содержание отчета:

1. Текст программы с комментариями
2. Верификация программы: описание и пример решения задачи, скриншоты, показывающие содержимое регистров и значения переменных после каждого действия программы, входные данные и скриншоты регистров и переменных в ключевых точках программы для проверки работы программы при разных входных данных.

4. Литература

1. Система команд FPU / club155. – url: http://www.club155.ru/x86cmdfpu

Приложение 1. Команды математического сопроцессора