Лабораторная работа № 2. Вещественные числа и операции над ними

Цель работы

Получить представление о представлении чисел с плавающей запятой в памяти ЭВМ и практические навыки по использованию команд вещественной арифметики на ассемблере.

Теоретические сведения

Введение

Важной частью современного процессора архитектуры x86 (IA-32, Intel) является устройство обработки чисел в формате с плавающей точкой. Данное устройство впервые появилось в составе компьютера на базе микропроцессора i8086/88 и получило название *математический сопроцессор*. Выбор названия определялся тем, что устройство было предназначено для расширения возможностей основного процессора и было реализовано в виде отдельной микросхемы. Эта микросхема имела название i8087. Включение устройства в компьютер было не обязательным. С появлением новых моделей процессоров x86 совершенствовались и сопроцессоры, хотя их программная модель осталась практически неизменной. Для процессоров моделей i80286 и i80386 сопроцессоры также были реализованы в виде отдельных кристаллов и имели обозначения i80287 и i80387 соответственно. А для процессора i80486 уже существовало две реализации: в одной сопроцессор оставался отдельным кристаллом, а в другой – он исполнялся на одном кристалле с основным процессором и становился неотъемлемой частью компьютера. Начиная с процессора типа Pentium, сопроцессор встраивается в основной процессор и преобразуется в блок операций с плавающей точкой (Float Point Unit – FPU).

Форматы данных математического сопроцессора

Представление вещественных чисел в двоичной системе счисления

Одной из форм записи вещественных чисел является их представление в *экспоненциальном виде*, в котором отдельно записывают мантиссу числа и его порядок. Однако число при этом может быть записано многими способами, например 7.5 может быть записано как 7.5×10^0 или 0.75×10^1 или 0.075×10^2 и т.д. Поэтому обычно числа в экспоненциальном виде нормализуют. В математике при *нормализованной записи вещественного числа* порядок p выбирается таким, чтобы абсолютная величина мантиссы M была не меньше единицы, но менее десяти: $1 \leq M < 10$, например, 350 записывается как 3.5×10^2. Этот вид записи позволяет легко сравнивать два числа в экспоненциальном виде.

Для представления вещественных чисел в памяти компьютера используют двоичную систему счисления и, соответственно, степени двойки вместо степеней десяти.

Обычную дробь, например, 0.324, в десятичной системе можно представить в виде:

$3/10+2/100+4/1000$ или $3/10^1+2/10^2+4/10^3$, где знаменатели – увеличивающиеся степени числа 10.
В двоичной дроби в качестве знаменателя используются степени 2. Так, двоичную дробь 0.101 можно записать:
$1/2^1+0/2^2+1/2^3$ или $1/2+0/4+1/8$,
что в десятичной системе равно $0.5+0.0+0.125 = 0.625$.
Так же, как и в десятичной системе счисления, в двоичной системе счисления не все дроби можно представить точно. Точно записываются только дроби, знаменатели которых являются степенями 2, например $3/4 = 1/2^1+0/2^2 = 0.11_2$ или $7/16 = 1/2+1/16 = 0.1001_2$ (здесь и далее нижний индекс обозначает основание системы счисления, в которой представлено число).
Кроме того, для представления точного значения, даже если оно существует, может не хватить разрядной сетки формата. Поэтому вещественные числа в памяти компьютера, как правило, *представлены не точно*. Точность их представления определяется количеством бит, отведенных в формате для записи мантиссы. В то время как диапазон представляемых значений определяется количеством бит, отведенных для записи порядка.
При ручном переводе вещественного числа из десятичного представления в двоичное, отдельно переводятся целая и дробная части. Как показывает опыт, перевод целой части вещественного десятичного числа в двоичный формат не представляет труда. А вот преобразование дробной части достаточно сложно. Существует несколько способов преобразования десятичной дроби в двоичную.

Разложение десятичной дроби на сумму дробей вида 1/2+1/4+1/8 +….

Метод заключается в разбиении десятичной дроби на сумму дробей, знаменатель которых кратен степеням двойки.
Например:
$0.625_{10} = 625/1000_{10} = 5/8_{10} = 4/8 + 1/8 = 1/2 + 1/8 = 0.1_2 + 0.001_2 = 0.101_2$
или
$5/8_{10} = 4/8 + 1/8 = 1/2 + 1/8 = 1/2 + 0/4 + 1/8 = 1/2^1 + 0/2^2 + 1/2^3 = 0.101_2$.
Однако большинство десятичных дробей нельзя представить в виде суммы дробей, чьи знаменатели раскладываются по степеням двойки. Так 1/5 в виде суммы дробей со знаменателями – степенями двойки представить нельзя. В этом случае строится приближенное представление, для получения которого следует использовать другие приемы перевода.

Перевод умножением десятичной дроби на 2

Этот метод является стандартным методом перевода десятичной дроби в двоичное представление. Алгоритм перевода следующий:
а) последовательно выполняют умножение исходной десятичной дроби и получаемых дробных частей произведений на 2 до тех пор, пока не будет получена нулевая дробная часть или достигнута требуемая точность вычислений;
б) целые части произведения записывают в том порядке, в котором они получаются.
Например:
а) перевод дроби 0.75:
$0.75 \times 2 = 1.5 \rightarrow 1$
$0.5 \times 2 = 1 \rightarrow 1$
$0 \times 2 = 1.5$
Результат перевода: $0.75_{10}=0.11_2$
б) перевод десятичной дроби 0.2:
$0.2 \times 2 = 0.4 \rightarrow 0$
$0.4 \times 2 = 0.8 \rightarrow 0$

$0.8 \times 2 = 1.6 \rightarrow 1$
$0.6 \times 2 = 1.2 \rightarrow 1$
$0.2 \times 2 = 0.4 \rightarrow 0$
$0.4 \times 2 = 0.8 \rightarrow 0$
$0.8 \times 2 = 1.6 \rightarrow 1$
…
Результат перевода: $0.2_{10}=(0.0011001\ldots)_2$

Формат IEEE[1] представления двоичных чисел с плавающей точкой

В основе всех форматов, закрепленных стандартами IEEE, лежит представление числа в виде мантиссы со знаком и порядка, которое определяется формулой:
$A = (\pm M) \times N^{\pm p}$,
где M – мантисса числа;
N – основание системы счисления, представленное целым положительным числом;
p – порядок числа, показывающий истинное положение точки в разрядах мантиссы.
Архитектура процессора, как правило, накладывает ограничения на представление вещественных чисел. Для процессоров Intel эти ограничения сформулированы следующим образом:

- основание системы счисления $N = 2$;
- чтобы с максимальной точностью хранить в памяти двоичное число с плавающей точкой, его мантисса M должна быть представлена в нормализованном виде;
- чтобы не хранить знак порядка, число, характеризующее порядок числа, должно быть положительным.

Двоичное нормализованное представление числа определяется выполнением условия $1_{10} \leq M_{10} \leq 2_{10}$ или в двоичной системе счисления $1.0\ldots00_2 \leq M_2 \leq 10.0\ldots00_2$, то есть единственная цифра целой части мантиссы должна быть значащей (единицей), а порядок p, соответственно, таким, чтобы это условие выполнялось. Таким образом, *число считается нормализованным*, если слева от десятичной точки находится только один двоичный разряд, значение которого равно 1.
Для выполнения последнего ограничения процессора, касающегося строго положительного «порядка» числа, в памяти хранится не сам порядок, а его *характеристика*, которая получается добавлением к порядку некоторого фиксированного смещения. Фиксированное смещение для каждого из трех форматов вещественных чисел имеет свое значение, которое определяется количеством разрядов, отведенных в формате на порядок. В процессе вычислений по характеристике *аппаратно* определяется порядок.
Таким образом значение А вещественного числа определяется формулой:
$A = (-1)^s \times 2^{q-f} \times M$,
где s – значение знакового разряда: 0 – число больше нуля, 1 – число меньше нуля;
$q = p+f$ – характеристика порядка p, которая представляет из себя сумму порядка и некоторой константы f;
M – нормализованная мантисса числа.

---

[1] Институт инженеров по электротехнике и электронике — IEEE (англ. Institute of Electrical and Electronics Engineers) — международная некоммерческая ассоциация специалистов в области техники, мировой лидер в области разработки стандартов по радиоэлектронике, электротехнике и аппаратному обеспечению вычислительных систем и сетей.

В соответствии со стандартом данные вещественного типа могут представляться в одном из трех форматов: короткое – с одинарной точностью, длинное – с двойной точностью и расширенное (см. таблицу 1).

Таблица 1 – Форматы представления вещественных чисел

Формат	**Описание**
Короткое вещественное число[2]	32 бита: первый бит – знак мантиссы, 8 бит – для представления характеристики и 23 бита – для представления мантиссы. С помощью этого формата можно представить нормализованные числа в диапазоне приблизительно от 2^{-126} до 2^{+127}.
Длинное вещественное число[3]	64 бита: первый бит – знак мантиссы, 11 бит – характеристика и 52 бита – мантисса. С помощью этого формата можно представить нормализованные числа в диапазоне приблизительно от 2^{-1022} до 2^{+1023}.
Расширенное вещественное число[4]	80 бит: первый бит – знак мантиссы, 16 бит – характеристика и 63 бита – мантисса. С помощью этого формата можно представить нормализованные числа в диапазоне приблизительно от 2^{-16382} до 2^{+16383}.

Если значение знакового бита равно 1, то число считается отрицательным, если 0 – то положительным. Число нуль считается положительным.
Нормализованное двоичное число согласно правилу нормализации всегда имеет целую часть, равную 1. Поэтому при его представлении в памяти появляется возможность считать первый разряд единичным по умолчанию и учитывать его наличие только на аппаратном уровне. Это дает возможность увеличить диапазон представляемых чисел, так как появляется лишний разряд, который можно использовать для представления мантиссы числа. Но это справедливо только для короткого и длинного форматов вещественного числа. Расширенный формат, как внутренний формат представления числа любого типа в сопроцессоре, содержит целую единицу в явном виде.
В таблице 2 представлены основные характеристики каждого из указанных форматов представления вещественных чисел.

Таблица 2 – Характеристики форматов вещественных чисел

Характеристика	**Короткий**	**Длинный**	**Расширенный**
Длина числа, бит	32	64	80
Размерность мантиссы, бит	24	53	64
Диапазон представляемых значений	10^{-38}…10^{+38}	10^{-308}…10^{+308}	10^{-4932}…10^{+4932}
Размерность характеристики q, бит	8	11	16
Фиксированное смещение f	+127	+1023	+16383

[2] Соответствует типу float в языке С.
[3] Соответствует типу double в языке С.
[4] Соответствует типу long double в языке С.

Диапазон порядка *p*	-126…127	-1022…1023	-16382…16383
Диапазон характеристики *q*	0…255	0…2047	0…32767

Как следует из таблицы, все положительные порядки имеют в двоичном представлении характеристики старший бит, равный единице, а отрицательные нет. Например:
а) p = -1, q= -1+127=126, характеристика=01111110;
б) p=1, q=1+127=128, характеристика=10000000.
Таким образом, *в старшем бите характеристики скрыт знак порядка вещественного числа*.

Обобщенный алгоритм преобразования вещественного десятичного числа в двоичное число с плавающей точкой формата IEEE.

Этот алгоритм описывает последовательность действий, которые нужно предпринять, для преобразования вещественного числа в двоичное число формата IEEE.

1. Представить целую часть вещественного числа в двоичном виде и поставить после нее десятичную точку.
2. Преобразовать дробную часть вещественного числа в двоичный формат, используя один из приведенных выше методов для определения значений всех битов мантиссы, предусмотренных форматом.
3. Записать полученное значение дробной части после десятичной точки. Если значение мантиссы меньше выделенного под нее количества разрядов, то дополнить ее незначащими нулями справа до предусмотренного форматом размера.
4. Нормализовать полученное двоичное число, определив значение показателя степени.
5. К показателю степени прибавить фиксированное число в соответствии с форматом, и представить его в двоичном виде. В результате будет получено значение характеристики числа, используемой в выбранном формате.
6. Записать значение характеристики в соответствующие биты формата перед нормализованной мантиссой, отбросив единицу целой части мантиссы при использовании короткого и длинного форматов.
7. Если число положительное, то в самый старший разряд представления следует записать 0, если отрицательное – то 1.

Другие форматы данных, обрабатываемые сопроцессором

Кроме трех основных вещественных форматов, для обработки которых собственно и разрабатывался сопроцессор, последний может работать и с другими форматами, хотя и менее эффективно. Сопроцессор может обрабатывать:
а) Целые числа:
- двоичные целые числа в трех форматах – 16, 32 и 64 бита;
- упакованные целые десятичные (BCD) числа – максимальная длина 18 упакованных десятичных цифр (9 байт).
б) Специальные численные значения:
- денормализованные вещественные числа;
- нуль;
- положительные и отрицательные значения бесконечности;
- нечисла;
- неопределенности и неподдерживаемые форматы.

Сопроцессор может обрабатывать только один формат упакованных десятичных чисел. Во внутреннем представлении такое число занимает 80 бит или 10 байт. Старший 10-й байт в старшем бите содержит знак числа. Остальная часть этого байта игнорируется. Остальные 9 байт содержат по две десятичных цифры. Соответственно в регистры сопроцессора можно поместить только 18 десятичных цифр. Упакованные десятичные числа также представляются в сопроцессоре в расширенном формате.
Несмотря на большой диапазон вещественных чисел, представимых в формате сопроцессора, бесконечное количество чисел находится за рамками представимого диапазона. Для получения возможности реагировать на ситуации, в которых могут быть получены значения из непредставимого диапазона, в сопроцессоре предусмотрены специальные комбинации бит, называемые специальными численными значениями.
К специальным значениям относятся:
- денормализованные вещественные числа,
- отрицательная и положительная бесконечности,
- нечисла;
- неопределенности,
- значения в неподдерживаемых форматах,
- нуль.
Для представления специальных значений зарезервированы минимальная 000…00 и максимальная 111…11 характеристики.
Нуль. Значение нуля относится к специальным. Это делается из-за того, что нуль выделяется среди корректных вещественных значений, формируемых как результат некоторой операции. Кроме того, нуль может формироваться как реакция сопроцессора на определенную ситуацию.
Нуль представляется с нулевой характеристикой и нулевой мантиссой:
0 0000000 00000000 000000000 000000000.
Денормализованные вещественные числа. Денормализованные числа – числа, значение которых меньше минимально представимого в нормализованном виде в формате. По мере приближения к нулю значение мантиссы уменьшается и наступает момент, когда разрядной сетки, отведенной под характеристику, становится мало. В этот момент значение характеристики обращается в нуль. Число, при достижении которого это происходит на самом деле отлично от нуля, так как между истинным нулем и минимальным нормализованным находится еще бесконечное множество чисел. Именно эти числа и представляют собой денормализованные числа. Однако диапазон денормализованных чисел не безграничен и определяется количеством разрядов, отведенных под мантиссу.
Денормализованное число имеет нулевую характеристику и ненулевую мантиссу, например:
0 0000000 00000100 00000111 00000000
Бесконечности. Сопроцессор имеет средства для представления бесконечности. Среди причин, приводящих к формированию значения бесконечности, в первую очередь следует назвать переполнение и деление на нуль.
Бесконечности кодируются с максимальным значением характеристики 111…11 и мантиссой равной 1. Соответственно различают положительную и отрицательную бесконечности:
$+\infty$: 0 1111111 10000000 00000000 00000000
$-\infty$: 1 1111111 10000000 00000000 00000000
Нечисла. К нечислам относятся такие битовые последовательности, которые не совпадают ни с одним из рассмотренных выше форматов. Нечисла представляются с

любым знаком, максимальной характеристикой и любой мантиссой кроме 1, например:
0 1111111 10000100 00000000 00000000
Неопределенность. Неопределенностью называют результат недействительной операции:
0 1111111 10000000 00000000 00000000
Неподдерживаемые форматы. Существует достаточно много битовых наборов, которые можно представить в расширенном формате. Для большинства из этих форматов формируется исключение «недействительная операция».

Программная модель сопроцессора

Программная модель сопроцессора с точки зрения программиста представляет собой совокупность регистров, каждый из которых имеет свое функциональное назначение. В программной модели сопроцессора можно выделить три группы регистров: регистры стека, служебные регистры и регистры данных (см. рисунок 1).

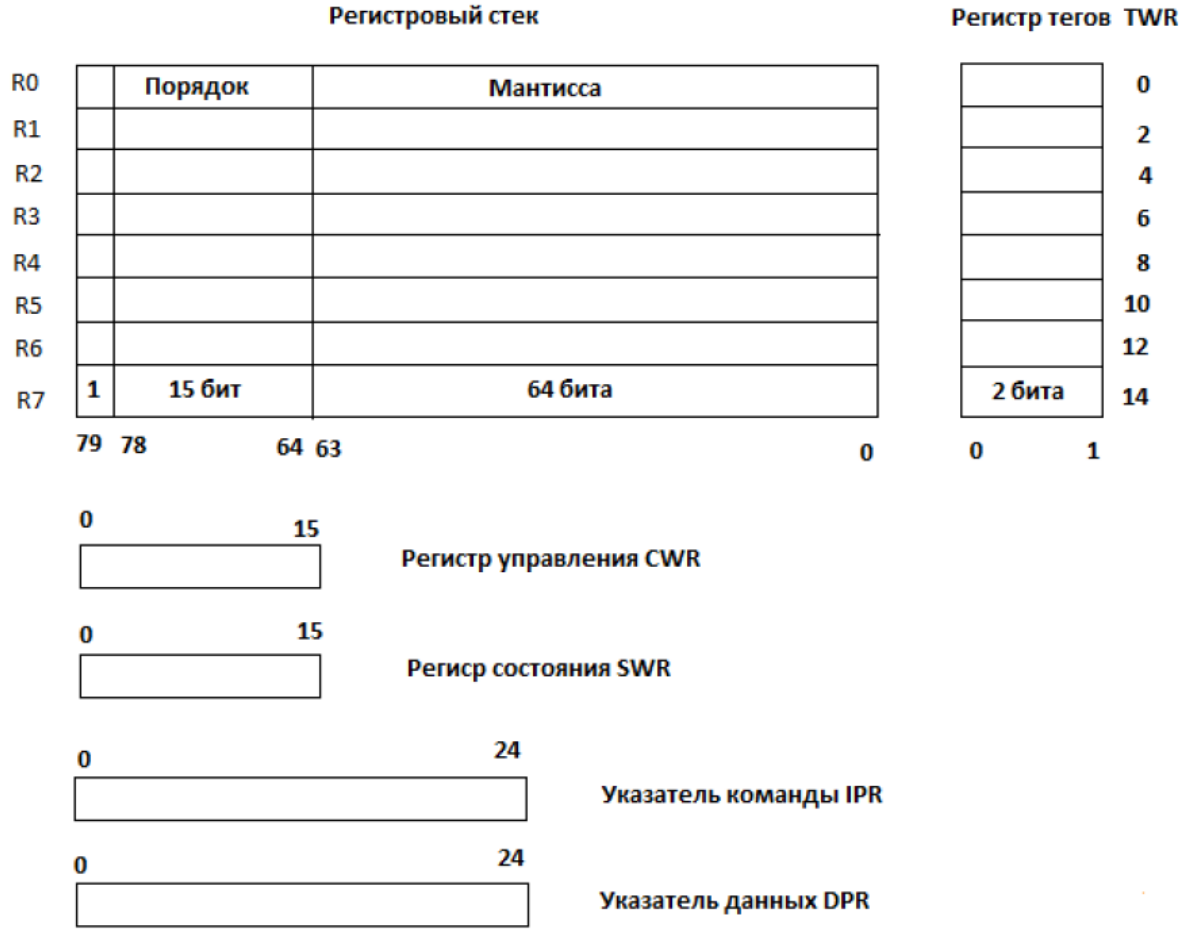

Рисунок 1 – Регистры сопроцессора

Регистры стека

Команды сопроцессора не оперируют физическими номерами регистров стека R0 – R7. Вместо этого они используют логические номера этих регистров St(0) – St(1). Именно с помощью логических номеров реализуется относительная адресация регистрового стека сопроцессора.
По мере записи в стек указатель его вершины движется по направлению к младшим номерам физических регистров. Так, если текущей вершиной стека был регистр R0,

то после записи очередного значения в стек сопроцессора текущей вершиной стека станет физический регистр R7. Логические же номера «плавают» вместе с текущей вершиной стека. Логический регистр St(0) всегда ассоциирован с вершиной стека. Таким образом реализуется принцип кольца.
Стековая организация регистров сопроцессора позволяет не привязываться к аппаратным ресурсам – физическим номерам регистров сопроцессора, а оперировать логической нумерацией, поддерживаемой на уровне системы команд. При этом не имеет значения номер физического регистра, в который помещаются данные, так как определяющим является порядок следования данных в стеке.

Служебные регистры

К служебным относятся три регистра – регистр состояния сопроцессора, управляющий регистр сопроцессора и регистр слова тегов.
Регистр состояния сопроцессора swr (Status Word Register) отражает текущее состояние сопроцессора после выполнения команды. В регистре содержатся поля, позволяющие определить: какой регистр является вершиной стека, какие исключения возникли после выполнения команды, какие особенности выполнения команды. Регистр состояния включает следующие биты:
а) *6 флагов исключительных ситуаций* – все возможные исключения сведены к шести типам, каждому из которых соответствует один бит регистра состояния (флаг).
Биты 0-5 устанавливаются при наступлении особых случаев, если они не маскированы:

- бит 0 – недействительная операция (IE);
- бит 1 – ненормализованный операнд (DE);
- бит 2 – деление на ноль (ZE);
- бит 3 – переполнение (OE);
- бит 4 – антипереполнение (UE);
- бит 5 – потеря точности (PE);

б) *бит sf (Stack Fault)* – ошибка работы стека сопроцессора, бит устанавливается в единицу, если возникла одна из трех исключительных ситуаций. В частности его установка информирует о попытки записи в заполненный стек или чтения из пустого стека.
После анализа этого бита его следует установить в 0.
SF=1, C1=1 – переполнение стека;
SF=1, C1=0 – чтение из пустого стека;
в) *бит es (Error Summary)* – суммарная ошибка сопроцессора – устанавливается в единицу, если возникает любая из шести перечисленных выше исключительных ситуаций;
г) *4 бита C0 –C3 – коды условия (Condition Code)* – назначение этих битов аналогично флагам в регистре флагов основного процессора – отразить результат выполнения операции.
Интерпретация битов:
бит 8 – C0 (CF); бит 9 – C1; бит 10 – C2 (PF); бит 14 – C3 (ZF);
д) *биты 11-13 – трехбитовое поле top* – содержит указатель регистра текущей вершины стека;
е) *бит 15 – бит занятости сопроцессора (B).*
Регистр управления cwr (Control Word Register) определяет особенности обработки численных данных. Он состоит из:
а) шести масок исключений – маски предназначены для маскирования исключительных ситуаций, возникновение которых фиксируется с помощью шести

флагов регистра состояния. Если какая то из шести масок установлена в 1, то соответствующее исключение будет обрабатываться самим сопроцессором. Если маски установлены в 0, то при возникновении соответствующего исключения будет возбуждено прерывание. При этом операционная система должна содержать соответствующий обработчик. Биты 0..5 – маска особых случаев – используется для запрета прерывания в случае, если:

- бит 0 – обнаружена недействительная операция (IM=0);
- бит 1 – обнаружен ненормализованный операнд (DM=1);
- бит 2 – зафиксировано деление ненулевого числа на ноль (ZM=1);
- бит 3 – обнаружено переполнение (OM=1);
- бит 4 – обнаружено антипереполнение (исчезновение порядка UM=1);
- бит 5 – обнаружена потеря точности (PM=1);

б) поле управления точностью (PC) – предназначено для выбора длины мантиссы. Биты 8-9 – 00 – 24 бита; 10 – 53 бита; 11 – 64 бита. По умолчанию значение поля устанавливается pc=11;
в) поля управления округлением (RC) – позволяет управлять процессом округления чисел во время работы сопроцессора. Необходимость округления может возникнуть, если в процессе выполнения команды получается непредставимый результат, например, периодическая дробь. Установка поля округления позволит выполнить округление в нужную сторону. Биты 10-11 задают режим округления:
00 – к ближайшему целому;
01 – округление в меньшую сторону;
10 – округление в большую сторону;
11 – отбрасывание дробной части (используется для приведения числа к форме, которая используется в операциях целочисленной арифметики).
г) бит 12 – интерпретации бесконечности (IC=1): 0 – беззнаковая бесконечность; 1 – бесконечность со знаком.
Регистр тегов twr (Tag Word Register) – используется для контроля состояния каждого из регистров R0 – R7. Команда сопроцессора использует этот регистр для определения, например, возможности записи в эти регистры.
Регистр отводит 2 бита на каждый регистр стека:
00 – в регистре не нулевое действительное число;
01 – в регистре истинный ноль;
10 – в регистре не число или бесконечность;
11 – регистр пуст.

Регистры указателей

Программная модель сопроцессора оперирует двумя регистрами указателей: указатель команды и указатель данных.
Регистр указатель команды ipr (Instraction Point Register) содержит адрес команды, вызвавшей особый случай, и 11 бит команды.
Регистр указатель данных dpr (data Point Register) содержит адрес операнда (если использовался) команды, вызвавшей особый случай.
Как было отмечено ранее, все регистры программной модели сопроцессора программно доступны. Однако, к одним из них доступ возможен с помощью специальных команд, предусмотренных в системе команд сопроцессора. К другим же регистрам доступ возможен только при выполнении дополнительных действий, так как специальных команд для такого доступа не предусмотрено.

Система команд сопроцессора

Система команд сопроцессора включает довольно много машинных команд. Она отличается большой гибкостью в выборе вариантов задания команд, реализующих определенную операцию, и их операндов. Минимальная длина команды – 2 байта. Все команды условно можно разбить на пять групп:

- передачи данных;
- арифметические;
- сравнения;
- трансцендентных операций;
- управления.

Мнемоническое обозначение команд сопроцессора характеризует особенности их работы. В связи с этим в мнемонике приняты следующие соглашения:
- первая буква всегда F;
- вторая буква определяет тип операнда в памяти, с которым работает команда:
I – обозначает операцию с целым числом из памяти;
B – операцию с десятичным числом из памяти;
при отсутствии этих букв – операция выполняется с вещественными числами;
- предпоследняя или последняя букв R (reversed) указывает обратный порядок операндов (для вычитания и деления, так как для этих команд очень важен порядок следования операндов);
- последняя буква P идентифицирует команду, последним действием которой является извлечение из стека операнда.

Команды передачи данных

Эта группа команд предназначена для организации обмена между регистрами стека, вершиной стека и ячейками памяти. С помощью этих команд осуществляются все перемещения значений операндов в сопроцессор и из него. Для каждого их трех типов данных, с которыми может работать сопроцессор, определена своя группа команд. Главная функция всех команд загрузки данных в сопроцессор является их преобразование в единое представление в виде расширенного вещественного числа. Все операции сохранения данных в память выполняют обратное преобразование.

1. Команды загрузки чисел в ST(0):

```
FLD <источник> ; вещественных чисел из памяти или стека
FILD <источник> ; целых чисел из памяти
FBLD <источник> ; десятичных чисел из памяти
```

Примеры:

```
FLD ST(0) ; копирует вершину стека
FLD QWORD PTR [EBX] ; загружает длинное вещественное
FILD WORD PTR [EDI] ; загружает целое число
```

2. Команды сохранения без извлечения из стека

```
- FST <приемник> ; копирует вещественное число из ST(0) в память
- FIST <приемник> ; копирует целое число из ST(0) в память
```

Примеры:

```
FST ST(5) ; копирует число из ST(0) в ST(5)
FST QWORD PTR [EBX] ; копирует вещ. число в память
FIST WORD PTR [EDI] ; копирует целое число в память
```

3. Команды записи в память с извлечением из стека

```
- FSTP <приемник>; перемещает значение ST(0) в память
(вещественное)
- FISTP <приемник>; перемещает значение ST(0) в память (целое)
- FBSTP <приемник>; перемещает значение ST(0) в память
(десятичное)
```

Примеры:
```
FSTP ST(5) ; перемещение в регистр стека (вещественное)
FISTP QWORD PTR [EBX] ; перемещение из стека в память (целое)
FBSTP P1 ; перемещение из стека в память (десятичное)
```
4. Команда обмена
```
- FXCH [<приемник>] ; меняет местами ST(0) и приемник, если
                    ; приемник не указан, то меняются местами
                    ; ST(0) и ST(1)
```
Пример:
```
FXCH ST(5) ; меняет местами ST(0) и ST(5)
FSQRT ; извлечение квадратного корня из ST(0)
FXCH ST(5) ; меняет местами ST(0) и ST(5)
```
5. Команды загрузки констант
```
FLDZ ; загрузка нуля
FLD1 ; загрузка единицы
FLDPI ; загрузка pi
FLDL2T ; загрузка двоичного логарифма десяти
FLDLG2 ; загрузка десятичного логарифма двух
FLDLN2 ; загрузка натурального логарифма двух
```

Арифметические команды

Данная группа команд реализует четыре основные арифметические операции – сложение, вычитание, умножение и деление. С точки зрения типов операндов, арифметические команды можно разделить на работающие с целыми и вещественными операндами.

Форматы основных арифметических команд:
```
Fop [ST(1),ST] ; стековая (всегда с извлечением P)
Fop ST(i),ST или Fop ST,ST(i) ; регистровая
FopP ST(i),ST ; регистровая с извлечением
Fop [ST,]<операнд2> ; вещественный операнд в памяти
Fop [ST,]<операнд2> ; целочисленный операнд в памяти
```
где op = ADD <операнд1>=<операнд1>+<операнд2>
```
SUB <операнд1>=<операнд1>-<операнд2>
SUBR <операнд1>=<операнд2>-<операнд1>
MUL <операнд1>=<операнд1>*<операнд2>
DIV <операнд1>=<операнд1>/<операнд2>
DIVR <операнд1>=<операнд2>/<операнд1>
```
Примеры:
```
FADD ; ST <- ST+ST(1) – в стеке только результат
FADDP ST(5),ST ; ST(5) <- ST(5)+ST и извлечение
```
Дополнительные арифметические команды
```
FSQRT ; извлечение квадратного корня
FSCALE ; масштабирование ST(0) <- ST(0)*2ST(1), ST(1) – целое
число
FPREM ; вычисляет частичный остаток ST(0) <- ST(0) - q*ST(1),
```
где q – целая часть результата ST(0)/ST(1)
```
FPREM1 ; вычисляет частичный остаток ST(0)¬ ST(0) - q*ST(1),
```
где q – ближайшее к ST(0)/ST(1) целое число
```
FPNDINT ; округление до целого
FXTRACT ; расцепляет число на порядок, который заменяет число в
ST(1) и мантиссу, которая помещается в ST(0)
FABS ; получение модуля числа
```

```
FCHS ; изменение знака числа
```

Команды сравнения

Команды этой группы выполняют сравнение значений числа в вершине стека и операнда, указанного в команде.

```
FCOM [<операнд>] ; сравнение чисел
FCOMP [<операнд>] ; сравнение чисел и одно извлечение
FCOMPP [<операнд>] ; сравнение чисел и два извлечения из стека
FICOM <операнд> ; сравнение с целым числом
FICOMP <операнд> ; сравнение с целым и извлечение из стека
FUCOM <регистр> ; сравнение с регистром
FUCOMP <регистр> ; сравнение с регистром и извлечение
FUCOMPP <регистр> ; сравнение с регистром и два извлечения
FTST ; сравнение с нулем и замена источника на нуль
FXAM ; анализ операнда
```

В результате работы команд сравнения в регистре состояния устанавливаются значения битов кода условия (см. таблицу 3).

Таблица 3 – Установка кодов условий при сравнении операндов

Условие	**C3**	**C2**	**C1**	**C0**
ST(0) > <операнд2> или ST(1)	0	0	X	0
ST(0) < <операнд2> или ST(1)	0	0	X	1
ST(0) = <операнд2> или ST(1)	1	0	X	0
Не сравнимы	1	1	X	1

Чтобы передать результаты сравнения из сопроцессора основному процессору для обработки полученных кодов командами условного перехода основного процессора, нужно записать биты условия в регистр процессора `eflags`.

Для этого:

1. Сразу после сравнения код необходимо сохранить в регистре AX или в памяти. Для этого в системе команд сопроцессора есть специальная команда

```
FSTSW AX ; сохранить слово состояния в регистре AX
```

2. Затем значения нужных бит извлекаются и анализируются основным процессором.

Это можно сделать двумя способами:

а) загрузить коды во флажковый регистр:

```
SAHF ; команда копирует биты: C3 в ZF, C2 в PF, а C0 в CF
```

Бит C1 выпадает из общего правила, так как в регистре флагов на месте соответствующего ему бита находится единица. Анализ этого бита нужно проводить с помощью логических команд основного процессора. Далее для перехода можно использовать команды JE, JNE, JA, JB, JBE, а для проверки операндов нечисел – команду JP;

б) можно использовать команду `TEST AX,<константа кода>`

Константы, используемые для анализа комбинации кодов, приведены в таблице 4.

Таблица 4 – Константы для анализа комбинации кодов

Результат сравнения	**Константа**	**Переход**	**B=TOP?0< XXXXXXXX**
ST > операнд	4500h	JZ	01000101 00000000
ST < операнд	0100h	JNZ	00000001 00000000
ST = операнд	4000h	JNZ	01000000 00000000

Не сравнимы	0400h	JNE	00000100 00000000

Команды трансцендентных функций

Сопроцессор имеет ряд команд, предназначенных для вычисления значений тригонометрических, а также логических и показательных функций. Это значительно облегчает работу программиста. Однако следует помнить, что значение всех операндов команд, вычисляющих тригонометрические функции, задаются в радианах.

1. Команды тригонометрические (угол задается в радианах):

```
FPTAN ; ST(1) = tg(ST), где -263 <= ST <= 264 и в ST – единица
FPATAN ; ST = arctg(ST(1)/ST(0)), операнды из стека извлекает
FSIN ; ST = sin(ST), где -263 <= ST <= 264
FCOS ; ST = cos(ST), где -263 <= ST <= 264
FSINCOS ; ST(1) = sin(ST), где -263 <= ST <= 264 и ST= cos(ST)
```

2. Команды логарифмические и показательные (по основанию 2)

```
F2XM1 ; ST =2ST-1, где -1 <= ST <= 1 – обеспечивает более точные
значения вблизи 1
FYL2X ; ST =ST(1)*log2ST , где ST > 0
FYL2XP1 ; ST =ST(1)*log2(ST+1) , где –(1-2½/2) <= ST <= 1-2½/2 –
обеспечивает более точные значения вблизи 1.
```

Задание: узнать, почему в качестве допустимых значений ST используются именно такие диапазоны?

Команды управления сопроцессором

Группа команд управления предназначена для управления работой сопроцессора. Мнемоники управляющих команд могут начинаться с FN (без ожидания и без проверки особых случаев) или F (с ожиданием). Команды с буквой N в мнемокоде выполняются немедленно, что позволяет сэкономить несколько тактов.

```
FNSTCW (FSTCW) ; записать содержимое CW в опер. память
FLDCW ; загрузить CW из оперативной памяти
FNSTSW (FSTSW) ; записать SW в оперативную память
FNSTSW AX (FSTSW AX) ; записать CW в AX
FNCLEX (FCLEX) ; сбросить флаги особых случаев в SW и биты ES, B
FNINIT (FINIT) ; инициализировать сопроцессор:
```

- управляющий регистр – бесконечность со знаком, округление к ближайшему, расширенная точность, все особые случаи замаскированы
- регистр состояния – B=0 (бит занятости сброшен), код условия не определен, ST=ES=0, флаги особых случаев установлены в нуль
- регистр тегов – все поля регистра тегов содержат значение 11 (пустой регистр))

```
FNSTENV (FSTENV) ; записать в память среду (содержимое всех
регистров, кроме численных, в предопределенном формате)
FLDENV ; загрузить среду
FNSAVE (FSAVE) ; записать полное состояние (дополнительно
сохраняет содержимое численных регистров)
FRSTOR ; восстановить полное состояние
FINCSTP ; увеличить указатель стека TOP на 1
FDECSTP ; уменьшить указатель стека TOP на 1
FFREE ; освободить регистр
FNOP ; нет операции (не производит никаких действий)
FSETPM ; установить защищенный режим работы (переводит
сопроцессор в защищенный режим работы)
```

Задания

1. Дан угол в радианах. Вывести его в градусах, минутах, секундах, т.е. перевести угол из радианной меры в градусную. Тестирование. Рекомендуется проверить работоспособность программы для углов, больших развернутого, а также для отрицательных.
2. Дана длина отрезка в футах и дюймах. Перевести в метры, сантиметры, миллиметры. 1 дюйм = 2,54 см. 1 фут = 3 hands = 12 inches (дюймам) = 30.48 сантиметрам, 1 hand = 4 дюйма = 10,16 см.
3. Дана длина отрезка в метрах, сантиметрах, миллиметрах. Перевести в футы и дюймы. 1 дюйм = 2,54 см. 1 фут = 3 hands = 12 inches (дюймам) = 30.48 сантиметрам, 1 hand = 4 дюйма = 10,16 см.
4. Заданы начало и конец промежутка в пределах одних суток в часах, минутах, секундах. Найти длину промежутка в тех же единицах измерения.
5. В такси одновременно сели три пассажира. Когда вышел первый пассажир на счетчике было P1 рублей, а когда вышел второй – P2 рублей. Сколько должен платить третий, если по окончанию поездки на счетчике P3 рублей? Плата за посадку – P0 рублей. Тестирование. Сумма оплаты всеми тремя должна быть равна показанию счетчика по окончании поездки. Если выходят все одновременно, то каждый платит (P0+P3)/3 рублей. Если же 1-й и 2=-й передумали ехать, то они платят по P0/3 , а оставшуюся сумму платит 3-й пассажир.
6. Функция sin(x) на отрезке [0; Pi/4] удовлетворительно аппроксимируется рядом Тейлора $y=x-x^3/6+x^5/120 -x^7/(120 \times 6 \times 7)+...$ Для заданного x вычислить y по этой формуле взяв 4-6 члена ряда Тейлора. Сравнить полученное значение с точным, вычисленным через стандартную функцию. Сколько точных знаков после запятой дает это формула ? Какова ее максимальная погрешность?
7. Функция Арктангенс относительно просто вычисляется, а через нее можно вычислить все остальные обратные тригонометрические функции. Ряд Тейлора для арктангенса $y= x- x^3/3 + x^5/5 – x^7/7 + x^9/9….$ Этот ряд достаточно быстро сходится при abs(x) <=1/2 Для заданного x вычислить y по этой формуле взяв 5-7 членов ряда Тейлора. Сравнить полученное значение с точным , вычисленным через стандартную функцию. Сколько точных знаков после запятой дает это формула ? Какова ее максимальная погрешность?
8. Для числа Pi известна формула Pi/4 == 4×Arctan(1/5) – arctan(1/239) Ряд Тейлора для арктангенса $y= x- x^3/3 + x^5/5 – x^7/7 + x^9/9….$ Этот ряд достаточно быстро сходится при abs(x) <=1/2 Сколько надо взять членов в ряде, чтобы вычислить число Пи хотя бы с 5-7 знаками после запятой? (Это умели еще в средние века). Сравнить полученное значение с точным.
9. Один из быстрых методов подсчета числа Пи основывается на непрерывной дроби Пи = 3+1/(7+1/(15+1/(1+1/(292+1/(1+1/(1+1/(1+1/(2+1/(1+1/(3+1/(1+1/14+...)))))))))))). Эта дробь называется правильной (знаменатели =1) и подходящей, т.к. доказано, что любая дробь с меньшим знаменателем дает худшую аппроксимацию. В 1954 г. Лемер в этом разложении вычислил 100 членов. Общая формула в 1985 г. была неизвестна. Сколько точных знаков числа Пи после запятой дает эта формула? Pi/4 = arctg(1.0)
10. Один из быстрых методов способов подсчета числа E основывается на непрерывной дроби E = 2+1/(1+1/(2+1/(1+1/(1+1/(4+1/(1+1/(1+1/(6+1/(1+1/(1+1/(8+1/1+...))))))))))). Эта дробь называется подходящей, т.к. доказано, что любая дробь с

меньшим знаменателем дает худшую аппроксимацию. В этом разложении закономерность видна 1,2,1,1,4,1,1,6,1,1,8,1,1,10,… Сколько точных знаков числа Е после запятой дает эта формула взятая до знаменателя 10 ?.E = Exp(1.0) либо воспользуйтесь рядом Тейлора.

11. Аппроксимация 6(Pi). Один из быстрых методов подсчета числа Пи основывается на непрерывной дроби Пи = $4/(1+1^2/(3+2^2/(5+3^2/(7+4^2/(9+\ldots)))))))))$. В этом разложении закономерность видна. Числители 1^2, 2^2, 3^2, 4^2 ,.. Знаменатели – 1,3,5,7,9,… Сколько точных знаков числа Пи после запятой дает эта формула взятая до знаменателя 15?
12. Треугольник задан координатами своих вершин на плоскости x1, y1, x2, y2, x3, y3. Найти его площадь и внутренние углы при вершинах.
13. Треугольник задан координатами своих вершин на плоскости x1, y1, x2, y2, x3, y3. Найти сумму длин медиан треугольника.
14. Треугольник задан координатами своих вершин на плоскости x1, y1, x2, y2, x3, y3. Найти координаты точки пересечения биссектрис (центр вписанной окружности).
15. Треугольник задан координатами своих вершин на плоскости x1, y1, x2, y2, x3, y3. Найти координаты точки центра окружности, описанной вокруг треугольника.
16. В пространстве заданы три вектора своими координатами x1, y1, z1, x2, y2, z2, x3, y3, z3. Найти объем пирамиды , построенной на этих векторах, как на сторонах.
17. Квадрат задан координатами своих противоположных точек на плоскости. Найти координаты 2-х других сторон. Подсказка. Стороны квадрата не обязательно параллельны осям координат.
18. Квадрат задан координатами своих 3-х угловых точек в пространстве. Найти координаты оставшейся вершины. Подсказка. Стороны квадрата не обязательно параллельны осям координат.
19. Куб задан координатами трех своих вершин, находящихся на одной грани и 4-й находящейся на другой грани, так что три ребра куба(построенные на этих вершинах)сходятся в одной точке- вершине этого куба. Найти координаты других 4-х вершин. Подсказка. Стороны куба не обязательно параллельны осям координат.
20. Спутник стационарно висит над земным шаром на высоте H = 100 км. На какой площади можно принимать телепередачи с этого спутника? В нулевом приближении Землю можно считать шаром с средним радиусом прибл. = 6371,032 км. (Экваториальный радиус прибл.= 6378,16 км).
21. С плоской крыши 9-этажного дома упала дробинка. За какое время она долетит до земли? Какую скорость при этом приобретет? Сопротивлением воздуха пренебречь. Считать, что высота одного этажа = 3,5м; чердака нет. В нулевом приближении Землю можно считать шаром с средним радиусом прибл.= 6371,032 км. (Экваториальный радиус прибл.= 6378,16 км.). Известно, что из-за выпуклости Земли у судна сперва бывает видна самая высокая точка, а только потом – оно само. В море плавают 2 яхты с высотой мачты h=15 м. (цифра условная). На каком максимальном расстоянии эти яхты могут заметить друг друга при идеальных условиях?
22. В нулевом приближении Волгоград м. считать состоящим из прямоугольников. Официальная длина г. Волгограда в 1985 г. – 72 км. Пусть ширина – 2 км. Кроме этого есть Дзержинский район длиной 4 км и шириной 4 км. Вы хотите построить сеть для мобильных телефонов. Одна единица

(сота) – это шестиугольник. Пусть сторона его равна 1,5 км.. Оцените: каково наименьшее число таких сот, чтобы покрыть весь Волгоград? 80% площади Волгограда?

23. Робинзон дежурит на самой высокой горе своего острова. Он знает – мимо может пройти корабль. Известно, что из-за выпуклости Земли у судна сперва бывает видна самая высокая точка, а только потом – оно само. Пусть самая высокая точка на острове – 50 м., а высота мачты на корабле – 20 м. На каком расстоянии от острова Робинзон может заметить корабль в идеальных условиях? В нулевом приближении Землю можно считать шаром с средним радиусом прибл.= 6371,032 км. (Экваториальный радиус прибл.= 6378,16 км.).

Контрольные вопросы

1. Объясните, почему в двоичной системе счисления не все вещественные числа могут быть представлены точно.
2. Перечислите методы перевода вещественного числа в двоичный формат.
3. Какие форматы вещественных чисел предусматривает IEEE?
4. Какие еще форматы чисел обрабатывает вещественный сопроцессор?
5. Назовите регистры стека. Объясните, как они реализованы и почему?
6. Как используются служебные регистры?
7. Для чего в сопроцессоре применяются регистры указателей?
8. Поясните систему наименования команд сопроцессора.
9. Назовите основные команды передачи данных.
10. Перечислите основные арифметические команды. Поясните особенности их выполнения.
11. Расскажите, в каком виде фиксируется результат операции сравнения вещественных чисел. Как его использовать в основном процессоре?
12. Перечислите особенности использования трансцедентальных функций.
13. Перечислите наиболее важные возможности, реализуемые с использованием команд управления сопроцессором.