Раздел 1.
Введение в архитектуру ЭВМ.

Форматы команд и способы адресации ЭВМ .

Десятичные числа **Двоично-десятичный код (BCD -Binary Coded Decimal)**

$1100_2 = C_{16}$ - пример кода знака «плюс»

$1101_2 = D_{16}$ - пример кода знака «минус»

Зонный формат: $1111_2 = F_{16}$ - пример кода «зона»

Байт		Байт			Байт		Байт	
Зона	Цифра	Зона	Цифра	…	Зона	Цифра	Знак	Цифра

Пример:

Число: -7396

Байт		Байт		Байт		Байт	
Зона	7	Зона	3	Зона	9	Минус	6
1111	0111	1111	0011	1111	1001	1101	0110

Упакованный формат:

Байт		Байт			Байт		Байт	
Цифра	Цифра	Цифра	Цифра	…	Цифра	Цифра	Цифра	Знак

Пример:

Число: -7396

Байт		Байт		Байт	
0	7	3	9	6	Минус
0000	0111	1111	1001	0110	1101

Вещественные числа

Форма с плавающей запятой(ПЗ), где число X = $\pm mq^{\pm p}$ – нормальная или полулогарифмическая форма

		Модуль порядка				Модуль мантиссы			
Знак m	Знак p	p_{r-1}	…	p_1	p_0	m_{-1}	m_{-2}	…	m_{-n}

Если поле p = 7 битам, то max значение, на которое умножается мантисса = 2^{128} (при **q = 2**) или 16^{128} (при **q = 16**), а диапазоны представления чисел:
$10^{-19}<|X|<10^{+19}$ **и** $10^{-76}<|X|<10^{+76}$ соответственно

Для упрощения операций над порядками их приводят к целым положительным числам, применяя **смещенный порядок**. Для этого к истинному порядку добавляется целое положительное число — **смещение**. Например, в системе со смещением 128 порядок -3 представляется как 125 (-3 + 128). Обычно смещение выбирается равным половине представимого диапазона порядков.

	Смещенный порядок				Модуль мантиссы			
Знак m	P_r	p_{r-1}	…	p_1	p_0	m_{-1}	m_{-2}	…	m_{-n}
$X=q2^p$	$X=q8^p$					$X=q16^p$			

Нормализованная мантисса, т.е. на она должна быть по модулю меньше единицы (|q|<1), а первая цифра после точки отличалась от нуля.

База	До нормализации		После нормализации	
	Порядок	Мантисса	Порядок	Мантисса
2	100	0,000110	001	0,110000
16	8	$0,001 \times 10^9$	6	$0,1 \times 10^9$

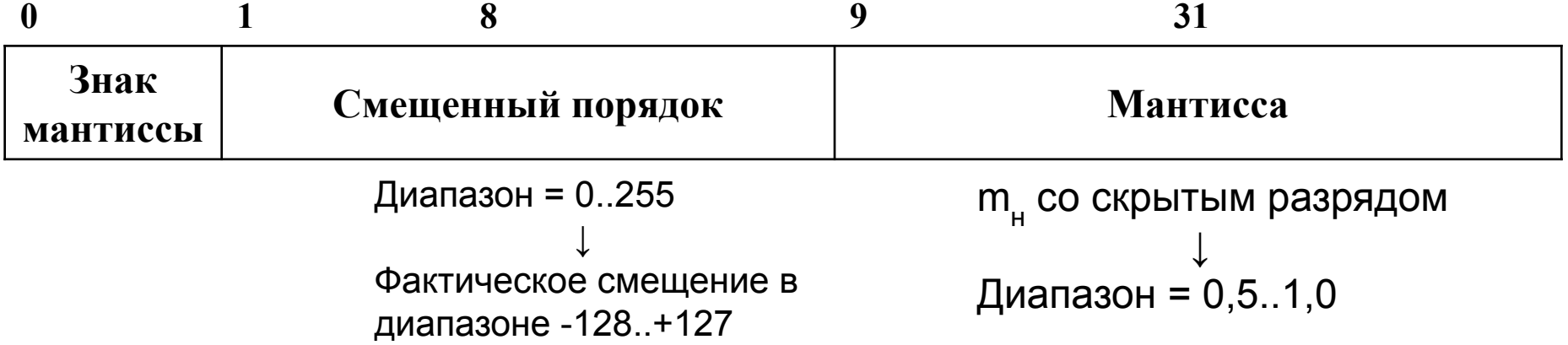
Способ повышения точности представления мантиссы – приемом скрытой единицы:

в нормализованной мантиссе **старшая цифра всегда равна единице** (для представления нуля используется специальная кодовая комбинация) → эта цифра может не записываться, а подразумеваться; запись мантиссы начинают с ее второй цифры, и это позволяет задействовать дополнительный значащий бит для более точного представления числа.
Следует отметить, что значение порядка в данном случае не меняется. Скрытая единица перед выполнением арифметических операций восстанавливается, а при записи результата — удаляется.
Пример: $m_н$ = 0,101000(1) → 0,010001 (старшая 1 – скрыта)

Для более существенного увеличения точности вычислений под число отводят несколько машинных слов.

ПРИМЕР:

типичный 32-битовый формат числа с ПЗ



В варианте с **ФЗ для целых чисел в дополнительном коде** могут быть представлены все целые числа от -2^{31} до $2^{31} - 1$, то есть всего 2^{32} различных чисел.

Для случая ПЗ возможны следующие диапазоны чисел:

1. отрицательные числа между $-(1 - 2^{-24}) \times 2^{127}$ и $-0{,}5 \times 2^{128}$;
2. положительные числа между $0{,}5 \times 2^{-128}$ и $(1 - 2^{-24}) \times 2^{127}$.

В эту область не включены пять участков:

1. отрицательные числа, меньшие чем $-(1 - 2^{24}) \times 2^{127}$ – ***отрицательное переполнение***;
2. отрицательные числа, большие чем $-0{,}5 \times 2^{128}$ – ***отрицательная потеря значимости***;
3. положительные числа, меньшие чем $0{,}5 \times 2^{-28}$ – ***положительная потеря значимости***;
4. положительные числа, большие чем $(1 - 2^{24}) \times 2^{127}$ – ***положительное переполнение***.

Эта запись **не учитывает нулевого значения** – используется специальная **кодовая комбинация**.

Переполнения возникают, когда в результате арифметической операции получается значение большее, чем можно представить порядком 127 ($2^{120} \times 2^{100} = 2^{230}$).

Потеря значимости — это когда результат представляет собой слишком маленькое дробное значение.

Для формата со смещенным порядком, имеет место **противоречие между диапазоном и точностью**. Если увеличить число битов, отведенных под порядок, расширяется диапазон представимых чисел. Но, т.к. может быть представлено только фиксированное число различных значений, уменьшается плотность и тем самым точность. Единственный путь увеличения как диапазона, так и точности — **увеличение количества разрядов**, поэтому в большинстве ВМ предлагается использовать числа в одинарном и двойном форматах.

Например:
число одинарного формата может занимать 32 бита, а двойного - 64 бита.

Одинарный формат:

	8 битов	23 бита
Бит знака	**Смещенный порядок**	**Мантисса**

Двойной формат:

	11 битов	52 бита
Бит знака	**Смещенный порядок**	**Мантисса**

Здесь представлены следующие классы чисел:

1. Порядки в диапазоне от 1 до 254 для одинарного формата и от 1 до 2036 — для двойного формата, используются для представления ненулевых нормализованных чисел. Порядки смещены так, что их диапазон составляет от -126 до +127 для одинарного формата и от -1022 до +1023 - для двойного формата. Нормализованное число требует, чтобы слева от двоичной запятой был единичный бит. Этот бит подразумевается, благодаря чему обеспечивается эффективная ширина мантиссы, равная 24 битам для одинарного и 53 битам — для двойного форматов.
2. Нулевой порядок совместно с нулевой мантиссой представляют положительный или отрицательный 0, в зависимости от состояния бита знака мантиссы.
3. Порядок, содержащий единицы во всех разрядах, совокупно с нулевой мантиссой представляют положительную или отрицательную бесконечность, в зависимости от состояния бита знака, что позволяет пользователю самому решить, считать ли это ошибкой или продолжать вычисления со значением, равным бесконечности.
4. Нулевой порядок в сочетании с ненулевой мантиссой представляют ненормализованное число. В этом случае бит слева от двоичной точки равен 0 и фактический порядок равен -126 или -1022. Число является положительным или отрицательным в зависимости от значения знакового бита.
5. Кодовая комбинация, в которой порядок содержит все единицы, а мантисса не равна 0, используется как признак «не числа» (NAN — Not a Number) и служит для предупреждения о различных исключительных ситуациях.

Упакованные числа с плавающей запятой (2x32 бит).

63 32	31 0
D1	D0

Упакованные числа с плавающей запятой (4x32 бит). – Технология SSE.

127 96	95 64	63 32	31 0
D3	D2	D1	D0

Упакованные числа с плавающей запятой (2x64 бит). – Технология SSE2.

127 64	63 0
D1	D0

Технология SSE:
ориентирована на параллельную обработку упакованных чисел с ПЗ.

Здесь числа объединяются в группы длиной 128 бит, и это позволяет упаковать в группу четыре 32-разрядных числа с ПЗ (числа с одинарной точностью).

Позже, в технологии SSE2, которую можно считать дальнейшим развитием SSE, появился формат, где в группу из 128 бит упаковываются два 64-разрядных числа с ПЗ, то есть числа, представленные с двойной точностью.

Разрядность основных форматов числовых данных.

Общепринятые величины разрядности кодов чисел следующие:

1. бит;
2. полубайт (4 бита) – тетрада;
3. байт (8 бит);
4. полуслово (16 бит);
5. слово (32 бита);
6. двойное слово (64 бита);
7. счетверенное слово (128 бит).

Разрядность целочисленного АЛУ (с ФЗ) обычно выбирается равной ширине адреса.

↓

Наиболее выгодными в плане быстродействия являются такие целые числа, длина которых совпадает с разрядностью адреса.

Использование более коротких чисел позволяет сэкономить на памяти, но выигрыша в производительности не дает.

Блоки операций с ПЗ рассчитаны на обработку чисел в формате двойной длины (64 бита).

↓

Наилучшим вариантом при проведении вычислений с ПЗ можно считать формат двойного слова. При выборе формата длины (32 разряда) вычисления все равно ведутся с большей точностью, после чего результат округляется.

↓

Использование короткого формата чисел с ПЗ кроме экономии памяти никаких иных преимуществ также не дает.

Символы, строки и тексты.

Символьная информация.

Символ – это определенная двоичная комбинация.

↓

Совокупность возможных символов и назначенных им двоичных кодов образует **таблицу кодировки.**
Все кодировки объединяет их **весовой принцип**: веса кодов цифр возрастают по мере увеличения веса символов в алфавитном порядке.

Кодовые таблицы, где символы – это восьмиразрядные двоичные комбинации (байтов), позволяющие представить 256 различных символов:

1. расширенный двоично-кодированный код **EBCDIC** (Extended Binary Coded Decimal Interchange Code) или **ДКОИ** (двоичный код для обработки информации);

2. американский стандартный код для обмена информацией **ASCII** (American Standard Code for Information Interchange) – 7-миразрядный, восьмая позиция отводится для записи бита четности (обеспечивает представление 128 символов) → **Latin 1** (**стандарт ISO 8859-1**): в зависимости от использования кодов 128-255 различают несколько вариантов стандарта **ISO 8859** (**ISO 8859-N**, где N от 1 до 16). → 16-битовый, стандарт **ISO 10646**, определяющий универсальный набор символов (UCS, Universal Character Set) – **Unicode**, задающий до 65 536 символов. Существуют его варианты **UCS-2** (каждый символ описывается двумя последовательными байтами m и n, так что номеру символа соответствует численное значение 256 x m + n), **UTF-8** (коды символов меньшие, чем 128, представляются одним байтом; в зависимости от символа его код может занимать от двух до шести байтов; ноль в старшем бите байта означает, что код занимает один байт и совпадает по кодировке с ASCII) и **UTF-7**.

   Стандарт **Unicode** обратно совместим с кодировкой **ASCII**, но если в ASCII для представления схожих по виду символов (минус, тире, знак переноса) применялся общий код, в Unicode каждый из этих символов имеет уникальную кодировку.

Строки и текст.

Строки - это непрерывная последовательность битов, байтов, слов или двойных слов.

Битовая строка может начинаться в любой позиции байта и содержать до $1 + 2^{32}$ бит.

Байтовая строка может состоять из байтов, слов или двойных слов.

Длина такой строки варьируется от нуля до 2^{32} - 1 байт (4 Гбайт).

Если байты байтовой строки представляют собой коды символов, то говорят о **текстовой строке**.

Поскольку длина текстовой строки может меняться в очень широких пределах, то для указания конца строки в последний байт заносится **код-ограничитель** - обычно это нули во всех разрядах байта. Иногда вместо ограничителя длину строки указывают числом, расположенным в первом байте (двух) строки.

Логические данные.

Элемент логических данных – логическая (булева) переменная = 1 или 0 («истина» или «ложь»).

Кодирование логического значения осуществляется битом информации.

В ВМ оперируют наборами логических переменных длиной в машинное слово.

Обрабатываются такие слова с помощью команд логических операций (И, ИЛИ, НЕ и т. д.), при этом все биты обрабатываются одинаково, но независимо друг от друга, то есть никаких переносов между разрядами не возникает.

Битовые поля.

Битовое поле — это число, занимающее некоторый набор битов, напрямую не адресуемое процессором. Например: при 8-битном байте первые два поля протокола IP— версия и IHL— будут битовыми полями. На машинах с 32-битным байтом все поля IP-пакета (кроме IP-адресов отправителя и получателя) будут битовыми.

Обращение к битовым полям требует дополнительных команд процессора для маскирования и сдвига, и потому медленнее обращений к словам/байтам. Поэтому битовые поля применяются для максимально полной упаковки информации в местах, где не важна скорость доступа к информации.

Компиляторы, как правило, ограничивают работу с битовыми полями только извлечением значения битового поля и записью значения в битовое поле, а само битовое поле воспринимается как беззнаковое число. Реальный порядок следования битовых полей в структуре является системно-зависимым: в одних компиляторах битовые поля могут быть расположены начиная с младших битов, а в других — со старших.

Прочие виды информации.

Представляемая в ВМ информация может быть статической или динамической. Так, числовая, символьная и логическая информация является статической – ее значение не связано со временем.

Видеоинформация.

Видеоинформация бывает как **статической**, так и **динамической**. **Статическая видеоинформация** включает в себя **текст**, **рисунки**, **графики**, **чертежи**, **таблицы** и др. **Рисунки делятся** также на: плоские — **двумерные** и объемные — **трехмерные**.

Динамическая видеоинформация - это **видео**-, **мульт**- и **слайд-фильмы**. Она используется либо для передачи движущихся изображений (анимация), либо для последовательной демонстрации отдельных кадров (слайд-фильмы).

В вычислительной технике существует **два способа представления графических изображений**: **матричный** (**растровый**) и **векторный**. **Матричные (bitmap) форматы** хорошо подходят для изображений со сложными гаммами цветов, оттенков и форм, таких как фотографии, рисунки, отсканированные данные. **Векторные форматы** более приспособлены для чертежей и изображений с простыми формами, тенями и окраской.

В **матричных форматах** изображение представляется прямоугольной матрицей точек — **пикселов** (picture element), положение которых в матрице соответствует координатам точек на экране. Помимо координат каждый пиксел характеризуется своим цветом, цветом фона или градацией яркости. Количество битов, выделяемых для указания цвета пиксела, изменяется в зависимости от формата. В высококачественных изображениях цвет пиксела описывают 24 битами, что дает около 16 млн цветов. **Основной недостаток матричной (растровой) графики** заключается в большой емкости памяти, требуемой для хранения изображения, из-за чего для описания изображений прибегают к различным методам сжатия данных. В настоящее время существует множество форматов графических файлов, различающихся алгоритмами сжатия и способами представления матричных изображений, а также сферой применения, например:

- BMP,
- GIF,
- PCX,
- JPEG,
- TIFF,
- PNG.

Векторное представление, в отличие от матричной графики, определяет описание изображения не пикселами, а кривыми - **сплайнами**. **Сплайн** - это гладкая кривая, которая проходит через две или более опорные точки, управляющие формой сплайна. В векторной графике наиболее распространены **сплайны на основе кривых Безье**. **Суть сплайна**: любую элементарную кривую можно построить, зная четыре коэффициента P0, P1, P2 и P3, соответствующие четырем точкам на плоскости. Перемещение этих точек влечет за собой изменение формы кривой

Хотя это может показаться более сложным, но для многих видов изображений использование математических описаний является более простым способом. В векторной графике для описания объектов используются математические формулы. Это позволяет при рисовании объектов вычислять, куда необходимо помещать реальные точки изображения. Имеется ряд простейших объектов, или примитивов, например эллипс, прямоугольник, линия. Эти примитивы и их комбинации служат основой для создания более сложных изображений. В простейшем случае изображение может быть составлено из отрезков линий, для которых задаются начальные координаты, угол наклона, длина, толщина линии, цвет линии и цвет фона.

Основное достоинство векторной графики - описание объекта, является простым и занимает мало памяти. Кроме того, векторная графика в сравнении с матричной имеет **следующие преимущества:**

простота масштабирования изображения без ухудшения его качества;

независимость емкости памяти, требуемой для хранения изображения, от выбранной цветовой модели.

Недостатком векторных изображений является их некоторая искусственность, заключающаяся в том, что любое изображение необходимо разбить на конечное множество составляющих его примитивов. Как и для матричной графики, существует несколько форматов графических векторных файлов. Некоторые из них:

- DXF,
- CDR,
- HPGL,
- PS,
- SVG,
- VSD.

Матричная и векторная графика существуют не обособленно друг от друга. Так, векторные рисунки могут включать в себя и матричные изображения. Кроме того, векторные и матричные изображения могут быть преобразованы друг в друга. Графические форматы, позволяющие сочетать матричное и векторное описание изображения, называются **метафайлами**. Метафайлы обеспечивают достаточную компактность файлов с сохранением высокого качества изображения.

Аудиоинформация.

Прежде чем быть представленной в ВМ, аудиоинформация должна быть преобразована в цифровую форму (оцифрована). Для этого значения звуковых сигналов (выборки, samples), взятые через малые промежутки времени, с помощью аналого-цифровых преобразователей (АЦП) переводятся в двоичный код. Обратное действие выполняется цифро-аналоговыми преобразователями (ЦАП). Чем чаще производятся выборки, тем выше может быть точность последующего воспроизведения исходного сигнала, но тем большая емкость памяти требуется для хранения оцифрованного звука.

Цифровой эквивалент аудиосигналов обычно хранится в виде файлов, причем широко используются различные методы сжатия такой информации. В настоящее время распространен целый ряд форматов хранения аудиоинформации:

- AVI,
- W,
- MIDI.
- AIF,
- MPEG,
- RA.

Классификация машинных операций

Классификация:

1. команды пересылки данных;
2. команды арифметической и логической обработки;
3. команды работы со строками;
4. команды SIMD;
5. команды преобразования;
6. команды ввода/вывода;
7. команды управления потоком команд.

Команды пересылки данных.

В таких командах должна содержаться следующая информация:

1. **адреса источника и получателя операндов** — адреса ячеек памяти, номера регистров процессора или информация о том, что операнды расположены в стеке;
2. **длина подлежащих пересылке данных** (обычно в байтах или словах), заданная явно или косвенно;
3. **способ адресации каждого из операндов**, с помощью которого содержимое адресной части команды может быть пересчитано в физический адрес операнда.

Рассматриваемая группа команд обеспечивает передачу информации между процессором и ОП (**тип «регистр-память»**), внутри процессора (**тип «регистр-регистр»**) и между ячейками памяти (**тип «память-память»**).

Команды арифметической и логической обработки.

Для каждой формы представления чисел обычно предусматривается некий стандартный набор операций.

Помимо вычисления результата выполнение арифметических и логических операций сопровождается формированием в АЛУ признаков (флагов), характеризующих этот результат. Наиболее часто фиксируются такие признаки, как:

Z (Zero) - нулевой результат;

N (Negative) - отрицательный результат;

V (oVerflow) — переполнение разрядной сетки;

C (Carry) — наличие переноса.

Операции с целыми числами.

К **стандартному набору операций над целыми числами**, представленными в форме с ФЗ, следует отнести:

1. двухместные арифметические операции (операции с двумя операндами):
   - сложение,
   - вычитание,
   - умножение,
   - деление;
2. одноместные арифметические операции (операции с одним операндом):
   - вычисление абсолютного значения (модуля) операнда,
   - изменение знака операнда;
3. операции сравнения, обеспечивающие сравнение двух целых чисел и выработку признаков, характеризующих соотношение между сопоставляемыми величинами (=, <>, >, <, <=, >=).

Часто **этот перечень дополняют такими операциями**, как:

1. вычисление остатка от целочисленного деления,
2. сложение с учетом переноса,
3. вычитание с учетом заема,
4. увеличение значения операнда на единицу (инкремент),
5. уменьшение значения операнда на единицу (декремент).

Операции с числами в форме с плавающей запятой.

Для работы с числами, представленными в форме с плавающей запятой предусмотрены:

1. основные арифметические операции:
   - сложение,
   - вычитание,
   - умножение,
   - деление;
2. операции сравнения, обеспечивающие сравнение двух вещественных чисел с выработкой признаков: =, <>, >, <, <=, >=;
3. операции преобразования:

- формы представления (между фиксированной и плавающей запятой),
- формата представления (с одинарной и двойной точностью).

Логические операции.

Стандартная система команд ВМ содержит команды для выполнения различных логических операций над отдельными битами слов или других адресуемых единиц. Такие команды предназначены для обработки символьных и логических данных. Минимальный набор поддерживаемых логических операций – это:

1. «НЕ»,
2. «И»,
3. «ИЛИ»,
4. сложение по модулю 2.

Операции сдвигов.

В дополнение к побитовым логическим операциям предусмотрены **команды сдвигов**:

1. логические сдвиги влево и вправо;
2. арифметические сдвиги вправо и влево;
3. циклические сдвиги влево и вправо.

При ***логическом сдвиге влево или вправо***, сдвигаются все разряды слова. Биты, вышедшие за пределы разрядной сетки, теряются, а освободившиеся позиции заполняются нулями.

При ***арифметическом сдвиге*** данные трактуются как целые числа со знаком, причем бит знака не изменяет положения. При сдвиге вправо освободившиеся позиции заполняются значением знакового разряда, а при сдвиге влево - нулями. Арифметические сдвиги позволяют ускорить выполнение некоторых арифметических операций. Так, если числа представлены двоичным дополнительным кодом, то сдвиги влево и вправо эквивалентны соответственно умножению и делению на 2.

При ***циклическом сдвиге*** смещаются все разряды слова, причем значение разряда, выходящего за пределы слова, заносится в позицию, освободившуюся с противоположной стороны, то есть потери информации не происходит. Одно из возможных применений циклических сдвигов - это перемещение интересующего бита в крайнюю левую (знаковую) позицию, где он может быть проанализирован как знак числа.

Операции с десятичными числами.

Десятичные числа представляются в ВМ в двоично-кодированной форме. В ВМ первых поколений для обработки таких чисел предусматривались специальные команды, обеспечивавшие выполнение основных арифметических операций (сложение, вычитание, умножение и деление). В современных машинах подобных команд обычно нет, а соответствующие вычисления имитируются с помощью команд целочисленной арифметики

Команды работы со строками.

Для работы со строками обычно предусматриваются команды, обеспечивающие перемещение, сравнение и поиск строк. В большинстве машин перечисленные операции просто имитируются за счет других команд.

Команды управления системой.

Команды, входящие в эту группу, являются привилегированными и могут выполняться, только когда центральный процессор ВМ находится в привилегированном состоянии или выполняет программу, находящуюся в привилегированной области памяти (обычно привилегированный режим используется лишь операционной системой). Так, лишь эти команды способны считывать и изменять состояние ряда ' регистров устройства управления.

Команды преобразования.

Команды преобразования осуществляют изменение формата представления данных. Примером может служить преобразование из десятичной системы счисления в двоичную или перевод 8-разрядного кода символа из кодировки ASCII в кодировку EBCDIC, и наоборот.

Команды ввода/вывода.

Команды этой группы могут быть **подразделены на**:

1. команды управления периферийным устройством (ПУ),
2. проверки его состояния,
3. ввода,
4. вывода.

Команды управления периферийным устройством служат для запуска ПУ и указания ему требуемого действия. Трактовка подобных инструкций зависит от типа ПУ.

Команды проверки состояния ввода/вывода применяются для тестирования различных признаков, характеризующих состояние модуля ввода/вывода и подключенных к нему ПУ. Благодаря этим командам центральный процессор может выяснить, включено ли питание ПУ, завершена ли предыдущая операция ввода/вывода, возникли ли в процессе ввода/вывода какие-либо ошибки и т. п. Собственно обмен информацией с ПУ обеспечивают команды ввода и вывода. Команды ввода предписывают модулю ввода/вывода получить элемент данных (байт или слово) от ПУ и поместить его на шину данных, а команды вывода — заставляют модуль ввода/вывода принять элемент данных с шины данных и переслать его на ПУ.

Команды SIMD.

Single Instruction Multiple Data — буквально «одна инструкция — много данных».
SIMD-команды обрабатывают сразу две группы чисел
Операнды таких команд представлены в одном из упакованных форматов.

Фирма Intel, добавила в систему команд своего микропроцессора Pentium MMX 57 SIMD-команд, под названием **MMX** (MultiMedia eXtention - мультимедийное расширение). Команды MMX обеспечивали параллельную обработку упакованных целых чисел. При выполнении арифметических операций каждое из чисел, входящих в группу, рассматривается как самостоятельное, без связи с соседними числами. Учитывая специфику обрабатываемой информации, команды MMX реализуют так называемую **арифметику с насыщением**: если в результате сложения образуется число, выходящее за пределы отведенных под него позиций, оно заменяется наибольшим двоичным числом, которое в эти позиции вмещается.

↓

Следующим шагом стало создание новых наборов SIMD-команд, работающих также с операндами, представленными в виде упакованных чисел с плавающей запятой (фирма AMD в микропроцессоре K6-2 – 21 SIMD-команда под названием 3DNow!)

↓

Фирма Intel ввела так называемые **потоковые SIMD-команды**, обозначив их аббревиатурой SSE - Streaming SIMD Extension (потоковая обработка по принципу «одна команда - много данных»). Сначала это были 70 команд в микропроцессоре Pentium III. Команды дополняли групповые целочисленные операции MMX и расширяли их за счет групповых операций с 32-разрядными вещественными числами.

В зависимости от типа чисел (целые или вещественные) **команды SSE делятся на три категории:**

1. работа с упакованными группами целых чисел, которые могут иметь размер байта, слова, двойного слова или квадрослова (количество чисел в группе зависит от их разрядности и от разрядности всей группы — 64 или 128);
2. оперирование одной парой 32-разрядных или 64-разрядных чисел с плавающей запятой (обычная или двойная точность);
3. обработка четырех пар вещественных чисел обычной точности или двух пар вещественных чисел двойной точности.

↓

SSE2 в Pentium 4: 271 команда, за один такт обрабатываются четыре 32-разрядных числа с ПЗ, упакованных в 128-разрядное слово.

↓

Технология 3DNow! → Enhanced 3DNow!. Этот набор команд близок к SSE2.

↓

Фирма IBM, процессоры серии PowerPC - реализация под названием AltiVec.

Команды управления потоком команд.

В адресной части таких команд содержится адрес точки перехода.

Переход реализуется путем загрузки адреса точки перехода в счетчик.

В системе команд ВМ можно выделить **три типа команд, способных изменить последовательность вычислений:**

1. безусловные переходы – **jump** (прыжок) – обеспечивает переход по заданному адресу без проверки каких-либо условий - **доминируют**;
2. условные переходы (ветвления) – **branch** (ветвление) – происходит только при соблюдении определенного условия, в противном случае выполняется следующая по порядку команда программы;
3. вызовы процедур и возвраты из процедур.

Условные переходы:

Условием, на основании которого осуществляется переход, - это ***признаки результата*** предшествующей арифметической или логической операции. Каждый из признаков фиксируется в своем разряде регистра флагов процессора. Возможен и иной подход, когда решение о переходе принимается в зависимости от состояния одного из регистров общего назначения, куда предварительно помещается результат операции сравнения. Третий вариант — это объединение операций сравнения и перехода в одной команде.

В системе команд ВМ для каждого признака результата предусматривается своя команда ветвления (иногда — две: переход при наличии признака и переход при его отсутствии). Большая часть условных переходов связана с проверкой взаимного соотношения двух величин или с равенством (неравенством) некоторой величины нулю. Последний вид проверок используется в программах наиболее интенсивно.

Одной из форм команд условного перехода являются ***команды пропуска***. В них адрес перехода отсутствует, а при выполнении условия происходит пропуск следующей команды, то есть предполагается, что отсутствующий в команде адрес следующей команды эквивалентен адресу текущей команды, увеличенному на длину пропускаемой команды. Такой прием позволяет сократить длину команд передачи управления.

Вызовы процедур и возвраты из процедур:

Процедурный механизм базируется на ***командах вызова процедуры***, обеспечивающих переход из текущей точки программы к начальной команде процедуры, и командах возврата из процедуры, для возврата в точку, непосредственно расположенную за командой вызова. Такой режим предполагает наличие средств для сохранения текущего состояния содержимого счетчика команд в момент вызова (запоминание адреса точки возврата) и его восстановления при выходе из процедуры.