1. Архитектурные особенности процессоров PENTIUM (P5)

Процессор PENTIUM, являясь программно совместимым с предыдущими процессорами, в то же время определяет новую точку отсчета в развитии архитектур фирмы INTEL. Именно с этого процессора активно начинает использоваться идеология параллельных систем, т.е. достижение высокой производительности будет обеспечиваться в первую очередь за счет мультимикропроцессорных средств процессора.
Использование субмикронной технологии в процессорах PENTIUM позволяет разработчикам фирмы INTEL располагать больше транзисторов на каждой подложке. Это сделало возможным увеличение количества транзисторов для семейства x86 от 29 000 в 8086 процессоре до 1,2 миллионов в процессоре Intel486 DX2, с наивысшим достижением в PENTIUM процессоре.
Выполненный по 0,8 микронной BiCMOS технологии, он содержит 3.1 миллиона транзисторов.
Процессор PENTIUM включает все особенности процессора INTEL486 и имеет ряд новых существенных черт, таких как:
- СУПЕРСКАЛЯРНАЯ АРХИТЕКТУРА, включающая два конвейера и позволяющая за один такт процессора выполнить более одной команды;
- КОНВЕЙЕРНОЕ УСТРОЙСТВО для обработки данных с плавающей точкой (FPU);
- РАЗДЕЛЬНЫЕ КЭШ-ПАМЯТИ КОМАНД И ДАННЫХ емкостью 8 Кбайт каждая; для КЭШ-памяти команд (программ) не нужно производить запись в основную память при обновлении строк КЭШа;
- 64-х битовая ШИНА ДАННЫХ обеспечивает обмен данными с системной платой со скоростью 270 МБайт/с;
- 32-х битовая ШИНА АДРЕСА;
- конвейеризация цикла магистрали;
Термин «СУПЕРСКАЛЯРНАЯ АРХИТЕКТУРА» обозначает микропроцессорную архитектуру, которая содержит БОЛЕЕ ОДНОГО ВЫЧИСЛИТЕЛЬНОГО БЛОКА.
Для ПРЕДСКАЗАНИЯ ВЕТВЛЕНИЙ В ПРОГРАММЕ процессор PENTIUM содержит ДВА БУФЕРА ПРЕДВЫ-БОРКИ КОМАНД, один из которых обеспечивает предвыборку команд на линейном участке, а другой служит для предвыборки команд в соответствии с алгоритмом функционирования буфера целевого ветвления BTB (Branch Target Buffer).
Процессор содержит два КОНВЕЙЕРА КОМАНД U и V.

U-конвейер может выполнять команды над данными целого и вещественного типов. V-конвейер выполняет простые команды над целыми, и команды FXCH над данными вещественного типа.

2. Архитектурные особенности процессоров P6, P7

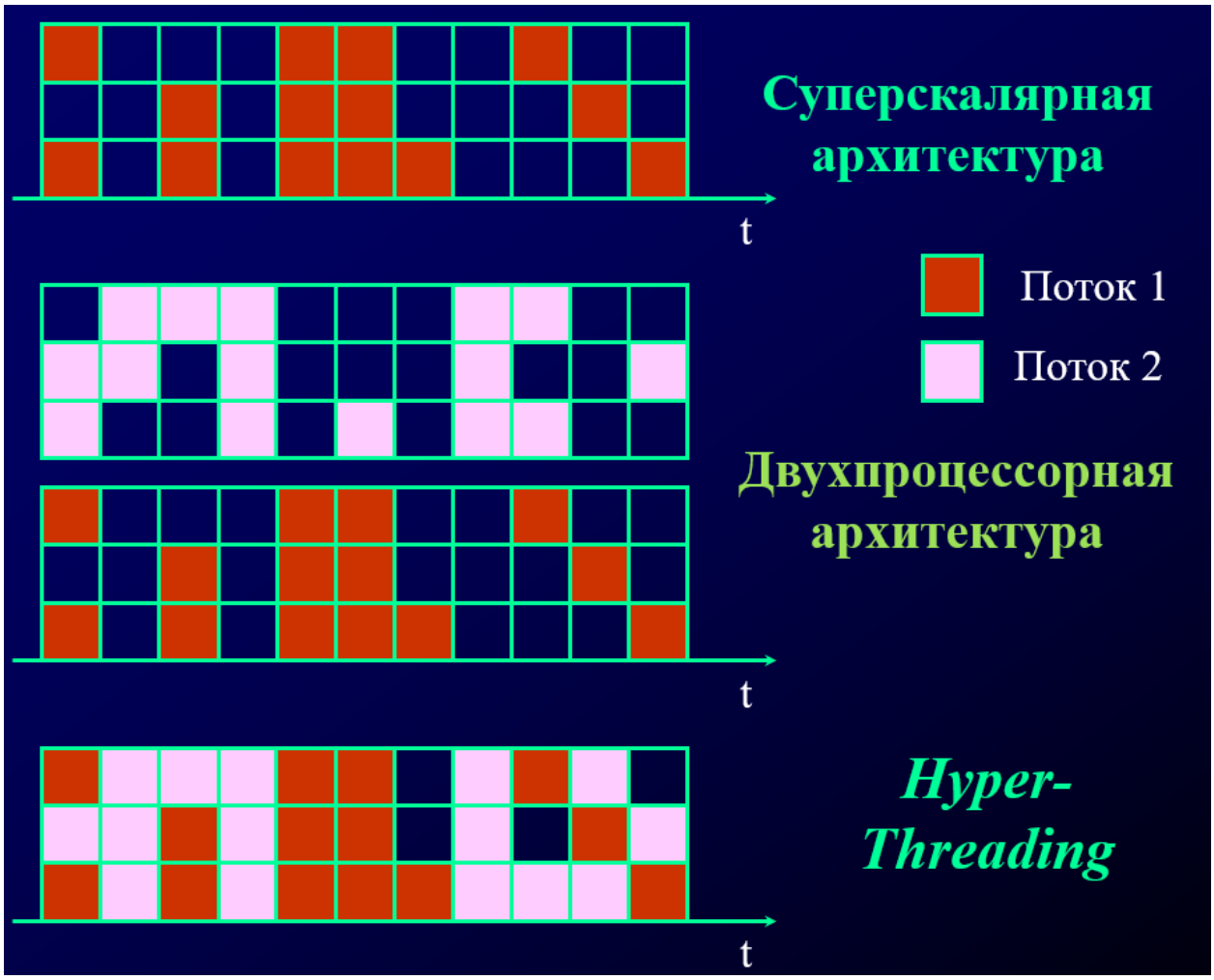
Главное преимущество и уникальная особенность МП PENTIUM PRO, именуемого «P6», – размещенная в одном корпусе с процессором вторичная статическая КЭШ-память размером 256 КБ, соединенная с процессором специально выделенной шиной.
Кристалл ЦПУ в P6 содержит 5,5 миллионов транзисторов, выполненных по 0,35 микронной технологии BiCMOS; кристалл КЭШ-памяти второго уровня – 15,5 миллионов.
PENTIUM содержит два 5-стадийных конвейера, которые могут работать параллельно и выполнять две целочисленные команды за машинный такт. При этом параллельно может выполняться только пара команд, следующих в программе друг за другом и удовлетворяющих определенным правилам, например, отсутствие регистровых зависимостей типа «запись после чтения».
В МП PENTIUM PRO (P6) для увеличения производительности осуществлен переход к одному 12-стадийному конвейеру.

Возможности суперскалярной архитектуры PENTIUM, с ее способностью к выполнению двух команд за такт, было бы трудно превзойти без совершенно нового подхода. В его основе лежит комбинация технологий, известная как DYNAMIC EXECUTION.
Примененный в PENTIUM PRO (P6) новый подход устраняет жесткую зависимость между тради-ционными фазами «ВЫБОРКИ» и «ВЫПОЛНЕНИЯ», когда последовательность прохождения команд через эти две фазы соответствует последовательности команд в программе.

В начале 21-го века появился новый процессор PENTIUM-4. Основные нововведения направлены на ускорение обработки потоковых данных, увеличено количество стадий конвейера до 20-ти.
При этом каждая элементарная операция в конвейере выполняется как по восходящему, так и по спадающему фронту тактовой частоты, т.е. процессор с тактовой частотой 2 ГГц выполняет 4 миллиарда микроопераций в сек.
Основные преимущества и характеристики процессоров PENTIUM-4 (Northwood):
- Тактовые частоты ядра от 2 ГГц и более
- Микроархитектура фирмы INTEL® NetBurst™
- 400, 500 и 533 МГц системная шина
- КЭШ-память 2 уровня 256 КБ с архитектурой Advanteced Transffer Cache
- Технология гиперконвейерной обработки
- Набор потоковых SIMD-расширений SSE2
- 128-ми разрядный блок вычислений с плавающей точкой
- 128-ми разрядный блок целочисленных вычислений с механизмом SIMD
- КЭШ-память 1-го уровня с отслеживанием исполнения команд (Execution Trace Cache)
- Усовершенствованная технология динамического исполнения
- Контроль температуры
- Встроенный механизм самотестирования (BIST)

ПРОЦЕССОР PENTIUM-4 3,06 ГГЦ С ТЕХНОЛОГИЕЙ HYPER-THREADING
14 ноября 2002 г. компания INTEL выпустила очередной процессор PENTIUM-4 с тактовой частотой 3,06 ГГц и поддержкой фирменной технологии HYPER-THREADING (HT), с помощью которой один физический процессор представляется операционной системе как два логических CPU.

На рис. приведен пример распределения работы в одном процессоре между ALU, блоком FPU (включая команды MMX) и блоком SIMD-FPU (8 регистров по 128 бит для работы с упакованными FP- или целочисленными данными).



Для реализации Hyper-Threading потребовалась небольшая модификация процессора, и часть блоков была дублирована (например, блоки ITLB – Instruction Translation Look-aside Buffer). Для воплощения HT число транзисторов в процессоре увеличилось не более, чем на 5%.

3. Выполнение арифметических и трансцендентных команд сопроцессорами x87

Математический сопроцессор x87 выполняет операции с плавающей точкой в 50..100 раз быстрее эквивалентных подпрограмм процессора x86. Программисты могут включать команды сопроцессора x87 в программы для процессора x86. Когда основной процессор встречает команду сопроцессора, он передает ее для выполнения этому сопроцессору x87. Возникает иллюзия того, что процессор x86 может выполнять команды с плавающей точкой.

При отсутствии сопроцессора x87 основной процессор x86, обнаружив команду сопроцессора, передает управление подпрограмме обработки особого случая, указанной операционной системой. Фирма INTEL поставляет подпрограмму E80x87, которая программно эмулирует действия сопроцессора x87. Эта подпрограмма создает иллюзию наличия в системе сопроцессора x87, но операции с плавающей точкой выполняются в 50..100 раз медленнее.

АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Базовые арифметические команды – сложение (FADD), вычитание (FSUB), умножение (FMUL) и деление (FDIV) – имеют два операнда (источник и приемник) и реализуют действия:

ПРИЕМНИК ← ПРИЕМНИК $ ИСТОЧНИК,

где $ - основные команды: +, –, *, /.

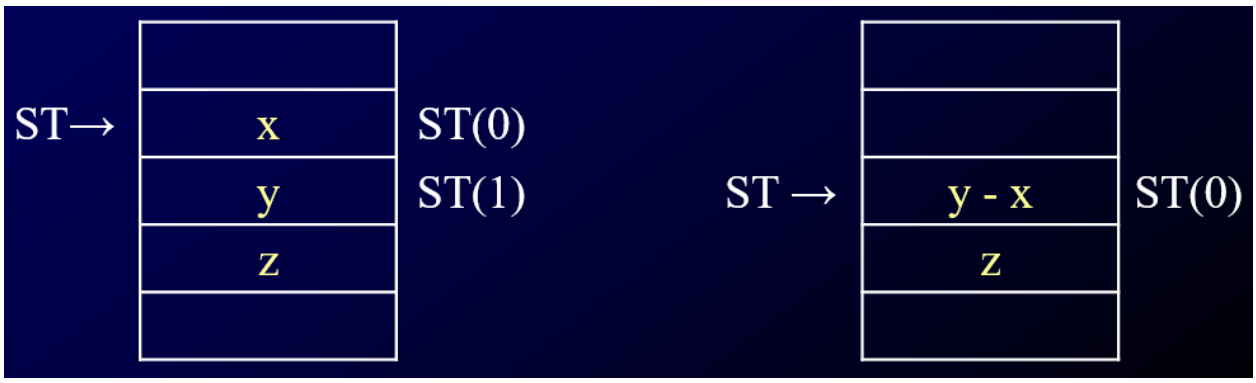
Для некоммутативных команд вычитания и деления имеются обратные варианты команд (в конце мнемоники добавляется буква R – reverse):

ПРИЕМНИК ← ИСТОЧНИК $ ПРИЕМНИК.

Во всех командах один операнд должен быть В ВЕРШИНЕ СТЕКА.

6 форм арифметических команд:

- FSUB mem,
- FISUB mem - адресуемый операнд в памяти является источником, а регистр вершины стека ST(0) - приемником. Преобразование в расширенный формат с плавающей точкой осуществляется в процессе выполнения команды. УКАЗАТЕЛЬ СТЕКА НЕ МОДИФИЦИРУЕТСЯ.
- FSUB ST(0), ST(i) - любой численный регистр ST(i) служит источником, а ST(0) - приемником. УКАЗАТЕЛЬ СТЕКА НЕ МОДИФИЦИРУЕТСЯ.
- FSUB ST(i), ST(0) - вершина стека является источником, а ST(i) - приемником. УКАЗАТЕЛЬ СТЕКА НЕ МОДИФИЦИРУЕТСЯ.
- FSUBP ST(i), ST - вершина стека является источником, а ST(i) - приемником. По окончании операции источник ST(0) извлекается из стека с последующим ИНКРЕМЕНТОМ УКАЗАТЕЛЯ СТЕКА.
- FSUB (FSUBP ST(1),ST(0)) - извлекает из вершины стека источник (потом инкрементирует указатель стека), затем извлекает приемник (еще раз инкрементирует указатель стека), выполняет операцию и перед включением результата в стек декрементирует указатель. В итоге - ВЕРШИНА СТЕКА СДВИНУЛАСЬ В СТОРОНУ УВЕЛИЧЕНИЯ. Последняя форма является частным случаем предыдущей.



ДОПОЛНИТЕЛЬНЫЕ АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Эти команды без явных операндов выполняют действия над содержимым вершины стека, результат помещают туда же БЕЗ МОДИФИКАЦИИ УКАЗАТЕЛЯ СТЕКА.

- FABS – нахождение абсолютной величины.
- FCHS – изменение знака операнда.
- FRNDINT – округление операнда до целого в формате с плавающей точкой.
- FSQRT – извлечение квадратного корня.
- FPREM – вычисляет остаток от деления содержимого ST(0) на число из ST(1). Остаток замещает число в ST(0).
- FSCALE – масштабирование на степень числа 2 – прибавляет целое число из ST(1) к порядку в регистре ST(0), т.е. умножает (или делит) ST(0) на число 2ST(1). Эту команду можно использовать для возведения числа 2 в целую степень (положительную или отрицательную).

КОМАНДЫ СРАВНЕНИЙ

- FCOM ST(i)/mem - сравнивает содержимое ST(0) с операндом "x" (в численном регистре или в памяти), т.е. производит вычитание операндов без запоминания результата и устанавливает коды условий в регистре состояния
- FICOM mem – сравнивает содержимое вершины стека ST(0) с целым числом в памяти.
- FCOMP ST(i)/mem – аналогична команде FCOM, но после сравнения производит извлечение операнда из вершины стека.
- FCOMPP ST(i) – сравнивает ST(0) с ST(i) и извлекает из стека оба операнда.
- FTST – сравнивает вершину стека с НУЛЕМ.
- FXAM – сравнивает вершину стека с НУЛЕМ, но выставляет 4 флага условий (в частности, определяется ненормализованная мантисса, бесконечность, нечисло и др.).

Флаги условий (C0, C3) сопроцессора x87 используются для организации условных переходов микропроцессором x86. Для этого командой – FSTSW

AX – содержимое регистра состояния x87 копируется в аккумулятор AX микропроцессора x86. После этого командой – SAHF – старший байт аккумулятора (AH) передается в младший байт регистра флагов. При этом условию C0 соответствует флаг CF, а условию C3 - флаг ZF.

C3	C0	Условие
0	0	ST(0) > x
0	1	ST(0) < x
1	0	ST(0) = x
1	1	ST(0) и x – не сравнимы

- FCOMI ST(0),ST(i) – сравнение вещественных чисел и установка флагов в EFLAGS (P6+).
- FCOMIP ST(0),ST(i) – сравнение вещественных чисел и установка флагов в EFLAGS и извлечение операнда из вершины стека (P6+).

ТРАНСЦЕНДЕНТНЫЕ КОМАНДЫ

К элементарным трансцендентным функциям относятся:
- тригонометрические функции (**sin, cos, tg и др**.),
- обратные тригонометрические функции (**arcsin, arctg** и др.),
- логарифмические функции **($\log_2(x)$, $\log_{10}(x)$, $\log_e(x)$),**
- показательные функции **(x^y, 2^x, 10^x, e^x),**
- гиперболические функции **(sh, ch, th и др.),**
- обратные гиперболические функции (**arsh, arch, arth** и др.).

Сопроцессор **x87** вычисляет любую из элементарных трансцендентных функций от аргументов **двойной точности**, давая результат **двойной точности** с ошибкой младшего разряда округления. Аргументы трансцендентных команд должны быть нормализованными.

Мнемоника	Описание команды	Вычисляемая функция
FPTAN	Частичный тангенс	ST(1) / ST(0) = tg (ST(0))
FSIN	Синус(387+)	ST(0) = sin (ST(0))
FCOS	Косинус (387+)	ST(0) = cos (ST(0))
FSINCOS	Синус, косинус (387+)	ST(7)=sin(ST(0)); ST(0)=cos(ST(0))
FPATAN	Частичный арктангенс	ST(0) = arctg (ST(1)/ST(0)
FYL2X	Двоичный логарифм	$ST(0) = ST(1) * \log_2 (ST(0))$
FYL2XP1	Двоичный логарифм	$ST(0) = ST(1) * \log_2 (ST(0)+1)$
F2XM1	Показательная функция	$ST(0) = 2^{(ST(0))} - 1$

4. 1) Вычисление адресов в процессорах фирмы INTEL. 2) Сегментация памяти

Для выборки ячейки памяти в адресном пространстве 1 Мбайт необходимо формировать 20-ти битовые адреса. Однако МП оперирует с 16-ти битовыми числами и поэтому может обратиться к блоку памяти, размером не более 2 16 = 64 КБайт. Такой «блок» называется СЕГМЕНТОМ ПАМЯТИ. Сегменты могут располагаться в любом месте памяти. Для указания начального адреса любого сегмента используются 16-ти битовые СЕГМЕНТНЫЕ РЕГИСТРЫ. Поэтому полный 20-ти разрядный адрес начала сегмента всегда имеет еще и четыре младших нулевых бита. Из этого следует, что сегменты начинаются по адресам, кратным 16.

В любой момент времени программа может обращаться к четырем сегментам, которые называются:

текущий Сегмент Кода (команды);

текущий Сегмент Данных;

текущий Сегмент Стека;

текущий Дополнительный Сегмент.

Каждый текущий сегмент идентифицируется путем записи старших 16 бит адреса его первого байта в одном из четырех специальных СЕГМЕНТНЫХ РЕГИСТРОВ. Отметим, что сегменты в памяти могут быть соседними (смежными), не перекрывающимися, частично или полностью перекрывающимися. Физическая ячейка памяти может принадлежать одному или нескольким сегментам.

Для вычисления ФИЗИЧЕСКОГО АДРЕСА начальный адрес сегмента умножается на 16 (сдвигается влево на 4 бита) и суммируется со смещением. Перенос из старшего бита, который может возникнуть при суммировании, игнорируется. Это приводит к так называемой кольцевой организации памяти, при которой за ячейкой с максимальным адресом FFFFF следует ячейка с нулевым адресом. Аналогичную кольцевую организацию имеет и каждый сегмент.

ЛОГИЧЕСКИЙ АДРЕС ячейки памяти состоит из двух 16-ти битовых беззнаковых чисел: НАЧАЛЬНОГО АДРЕСА СЕГМЕНТА и ВНУТРИСЕГМЕНТНОГО СМЕЩЕНИЯ, которое определяет расстояние от начала сегмента до этой ячейки.

Сегментная структура памяти обеспечивает возможность создания позиционно независимых или динамически перемещаемых программ, что необходимо в мультипрограммной среде для эффективного использования оперативной памяти. Это позволяет произвольно перемещать программу в

адресном пространстве памяти, изменяя только содержимое сегментных регистров.

5. 1) Защищенный режим виртуальной адресации. 2) Типы дескрипторов

1). PROTECTED VIRTUAL ADDRESS MODE – защищенный режим виртуальной адресации . В этом режиме процессор позволяет адресовать до 16 Мбайт физической памяти, через которые могут отображаться до 1 Гбайта виртуальной памяти каждой задачи. Система команд в этом режиме также включает набор команд 8086, расширенный для аппаратной реали- зации функций супервизора многозадачной ОС и виртуальной памяти.

Переключение в защищенный режим осуществляется одной коман-дой достаточно быстро. Обратное переключение в реальный режим воз-можно только через аппаратный сброс процессора, что требует значитель-ных затрат времени. По составу и назначению В РЕАЛЬНОМ РЕЖИМЕ регистры I80286 в основном совпадают с регистрами I8086/88. Изменения касаются назначения бит регистра флагов и использования сегментных регистров в защищенном режиме. Как и I8086, процессор I80286 имеет 16-ти битовую шину данных и очередь команд 6 байт. За счет улучшения архитектуры сокращено среднее количество тактов выполнения операций.

Под управлением MS DOS процессор I80286 обычно используется в реальном режиме работы. Защищенный режим используют многозадачные ОС типа XENIX, UNIX, OS/2, NET-WARE-286, MS WINDOS.

2). В ЗАЩИЩЕННОМ РЕЖИМЕ работают все режимы адресации, допустимые для I8086 и реального режима I80286. Отличия касаются определения сегментов:

сегментные регистры – CS, DS, SS, ES – хранят не сами базовые адреса сегментов, а СЕЛЕКТОРЫ, по которым из таблицы дескрипторов, хранящейся в ОЗУ, извлекаются ДЕСКРИПТОРЫ СЕГМЕНТОВ;

дескриптор описывает базовый адрес, размер сегмента (1 ÷ 64 Кбайта) и его атрибуты;

6. Команды MMX, SSE, SSE-2, SSE-3, AVX

Идея **MMX** (первоначальное название этой технологии Multi-Media eXtension было изменено фирмой INTEL на Matrix-Math eXtension) родилась как средство, решающее часть проблем с нехваткой быстродействия центрального процессора для задач мульти-медиа приложений, характеризующихся следующими факторами:

-небольшие целочисленные данные(например, 8-ми битовые графические пиксели или 16-ти битовые элементы звукового потока), которые обрабатываются в значительных объемах;

-небольшие, многократно повторяющиеся циклы;

-частые умножения и накапливания результата (выражения типа:

«a = a + b»);

-постоянные циклические обращения к памяти.

Параллельность вычислений – ключевой момент в технологии MMX. Поэтому большинство новых MMX-команд построены по принципу «ОДНА КОМАНДА – МНОЖЕСТВО ДАННЫХ» (SINGLE INSTRUCTION MULTIBLE DATA – SIMD). Так как процессоры PENTIUM имеют 64-х разрядную шину данных, базовая длина слова данных новых MMX-инструкций выбрана тоже 64 разрядной.

MMX-регистры хранят только MMX-данные, остальные восемь

РОН-ов (EAX, EBX и др.) используются для адресации памяти, работы с

циклами или любых других операций.

MMX-данные хранятся в младших 64-х битах 80-ти битовых FP-

регистров (на месте мантиссы FP-данных). Поле порядка и знака (старшие 16 бит) устанавливаются в состояние «все единицы», что соответствует состоянию «минус бесконечность» или «нечисло» для FP-данных. Поэтому FP- и MMX-данные невозможно случайно неверно интерпретировать.

Таблица 2.35 – *Команды MMX.*

КОМАНДЫ ПЕРЕСЫЛКИ	
EMMS	Очистка стека регистров – установка всех единиц в слове тегов
MOVD mm, m32/ir32	Пересылка данных в младшие 32 бита регистра MMX с заполнением старших бит нулями
MOVD m32/ir32, mm	Пересылка данных из младших 32-х бит регистра MMX
MOVQ mm, mm/m64	Пересылка данных в регистр MMX
MOVQ mm/m64, mm	Пересылка данных из регистра MMX
PACKSSDW mm, mm/m64	Упаковка со знаковым насыщением двух двойных слов, расположенных в mm, и двух двойных слов mm/m64 в четыре слова, помещаемых в mm
PACKSSWB mm, mm/m64	Упаковка со знаковым насыщением четырех слов, расположенных в mm, и четырех слов mm/m64 в восемь байт, помещаемых в mm
PACKUSWB mm, mm/m64	Упаковка с насыщением четырех знаковых слов, расположенных в mm, и четырех слов mm/m64 в восемь беззнаковых байт, помещаемых в mm
PUNPCKHBW mm, mm/m64	Чередование в регистре-приемнике байтов старшей половины операнда-источника с байтами старшей половины операнда-приемника
PUNPCKHWD mm, mm/m64	Чередование в регистре-приемнике слов старшей половины операнда-источника со словами старшей половины операнда-приемника
PUNPCKHDQ mm, mm/m64	Чередование в регистре-приемнике двойного слова старшей половины операнда-источника с двойным словом старшей половины операнда-приемника
PUNPCKLBW mm, mm/m64	Чередование в регистре-приемнике байтов младшей половины операнда-источника с байтами младшей половины операнда-приемника
PUNPCKLWD mm, mm/m64	Чередование в регистре-приемнике слов младшей половины операнда-источника со словами младшей половины операнда-приемника
PUNPCKLDQ mm, mm/m64	Чередование в регистре-приемнике двойного слова младшей половины операнда-источника с двойным словом младшей половины операнда-приемника

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ КОМАНДЫ	
PADDB mm, mm/m64 PADDW mm, mm/m64 PADDD mm, mm/m64	Сложение упакованных байт (слов или двойных слов) без насыщения (с циклическим переполнением)
PADDSB mm, mm/m64 PADDSW mm, mm/m64	Сложение упакованных байт (слов) со знаковым насыщением
PADDUSB mm, mm/m64 PADDUSW mm,mm/m64	Сложение упакованных байт (слов) с беззнаковым насыщением
PSUBB mm, mm/m64 PSUBW mm, mm/m64 PSUBD mm, mm/m64	Вычитание упакованных байт (слов, двойных слов) без насыщения (с циклическим переполнением)
PSUDSB mm, mm/m64 PSUDSW mm, mm/m64	Вычитание упакованных знаковых байт (слов) с насыщением
PSUBUSB mm, mm/m64 PSUBUSW mm,mm/m64	Вычитание упакованных беззнаковых байт (слов) с насыщением
PMULHW mm, mm/m64	Умножение упакованных знаковых слов с сохранением только старших 16 бит элементов результата
PMULLW mm, mm/m64	Умножение упакованных знаковых или беззнаковых слов с сохранением только младших 16 бит результата
PMADDWD mm, mm/m64	Умножение четырех знаковых слов операнда-источника на четыре знаковых слова операнда приемника. Два двойных слова результатов умножения младших слов суммируются и записываются в младшие 32 бита операнда-приемника. Два двойных слова результатов умножения старших слов суммируются и записываются в старшие 32 бита операнда-приемника
PAND mm, mm/m64	Логическое "И"
PANDN mm, mm/m64	Логическое "И-НЕ"
POR mm, mm/m64	Логическое "ИЛИ"
PXOR mm, mm/m64	Исключающее "ИЛИ"
PCMPEQB mm,mm/m64 PCMPEQW mm,mm/m64 PCMPEQD mm,mm/m64	Сравнение (на равенство) упакованных байт (слов, двойных слов). Все биты элемента результата будут единичными (True) при равенстве соответствующих элементов операндов и – нулевыми (False) при неравенстве
PCMPGTB mm,mm/m64 PCMPGTW mm,mm/m64 PCMPGTD mm,mm/m64	Сравнение (по величине) упакованных знаковых байт (слов, двойных слов). Все биты элемента результата будут единичными (True), если соответствующий элемент операнда-получателя больше элемента операнда-источника, и нуле-

SSE. Основное отличие новой технологии «MMX-2», так называемое, потоковое расширение – SSE (Streaming SIMD Extensions), получившей офи-

циальное название «Новый набор инструкций KATMAI» – KNI (Katmai New Instruction), – использование параллельных вычислений над числами с плавающей запятой одинарной точности (SIMD-FP, Single Instruction Multiple Data — Floating Point). В добавление к 57 MMX-командам процессор PENTIUM-3 (более раннее название – KATMAI) имеет еще 70 команд, ориентированных на SIMD-FP (направленных на обработку 3D-приложений), а также дополнительные целочисленные инструкции с регистрами MMX и команды управления кэшированием. Добавлены специализированные команды для ускорения трансформации трехмерных объектов, эффектов освещения, атмосферных эффектов и др.

Расширение SSE-2. В новом процессоре PENTIUM-4 (2000 г.) основные нововведения направлены на ускорение обработки потоковых данных, что обеспечивает максимальную производительность при обработке видео, графических изображений, работе с мультимедиа и в других сложных задачах. Поток в данном контексте подразумевает, что с его данными должны выполняться однотипные операции. Кроме того, данные, уже прошедшие обработку, в дальнейшем этим вычислительным процессом использоваться не будут и ими не следует засорять КЭШ.

Добавлены дополнительные 144 команды (SSE-2), ускоряющие работу широкого спектра потоковых ресурсоемких приложений.

В процессоре PENTIUM-4 добавлены новые форматы данных:

-упакованная пара чисел с плавающей точкой двойной точности (2 * 64 = 128 бит);

-упакованные целые числа: 16 байт (16 * 8 = 128 бит), 8 слов (8 * 16 = 128 бит), 4 двойных слова (4 * 32 = 128 бит) и 2 счетверенных слова (2 * 64 = 128 бит).

Расширение SSE-3. В процессор PENTIUM-4 на ядре PRESCOTT (который появился в конце 2003 г.) добавлены еще 13 новых команд SIMD-FP, получившие название SSE-3.

Advanced Vector Extensions (AVX) – расширение системы команд x86 для микропроцессоров Intel и AMD, предложенное Intel в марте 2008.

Новые возможности AVX:

-Новая схема кодирования инструкций VEX;

-Ширина векторных регистров SIMD увеличивается со 128 (XMM) до 256 бит (регистры YMM0 – YMM15). Существующие 128-битовые SSE инструкции будут использовать младшую половину новых YMM регистров, не изменяя старшую часть. Для работы с YMM регистрами добавлены новые 256-битовые AVX инструкции. В будущем возможно расширение векторных регистров SIMD до 512 или 1024 бит. Например,

процессоры с архитектурой Larrabee уже имеют векторные регистры (ZMM) шириной в 512 бит, и используют для работы с ними SIMD команды с MVEX и VEX префиксами, но при этом они не поддерживают AVX;
-Неразрушающие операции. Набор AVX инструкций использует трёхоперандный синтаксис. Например, вместо a = a + b можно использовать c = a + b, при этом регистр a остаётся неизменённым. В случаях, когда значение a используется дальше в вычислениях, это повышает производительность, так как избавляет от необходимости сохранять перед вычислением и восстанавливать после вычисления регистр, содержавший a, из другого регистра или памяти;
-Для большинства новых инструкций отсутствуют требования к выравниванию операндов в памяти. Однако, рекомендуется следить за выравниванием на размер операнда, во избежание значительного снижения производительности.
-Набор инструкций AVX содержит в себе аналоги 128-битовых SSE инструкций для вещественных чисел. При этом, в отличие от оригиналов, сохранение 128-битового результата будет обнулять старшую половину YMM регистра. 128-битовые AVX инструкции сохраняют прочие преимущества AVX, такие как новая схема кодирования, трехоперандный синтаксис и невыровненный доступ к памяти. Рекомендуется отказаться от старых SSE и MMX инструкций в пользу новых 128-битовых AVX инструкций, даже если достаточно двух операндов.

7.Организация кэш-памяти. Методы записи кэш-памяти

Вместо медленной динамической памяти можно было бы использовать более быструю статическую память, которая строится на триггерах, и имеет примерно такое же быстродействие, что и процессор.

Однако статическая память более дорогая, чем динамическая, что ведет к удорожанию всего ПК. Поэтому в ЭВМ с целью установления приемлемого соотношения стоимость/производительность используются различные структурные решения, как:1)конвейеризация процедур цикла выполнения команд;

2)расслоение модуля оперативной памяти;

3)буферизация.

В простейшем случае конвейеризация заключается в выполнении операций в процессоре параллельно с выборкой из памяти следующей команды и операндов. Такое техническое решение позволяет существенно снизить простой процессора из-за ожидания поступления из памяти очередной команды, поскольку команда будет читаться из памяти заранее, когда процессор занят выполнением предыдущей команды.

При записи новой информации в КЭШ операция сравнения тегов должна предшествовать всем остальным действиям, поскольку проверка тегов не может выполняться параллельно с другой работой, то операция записи занимает больше времени, чем операция чтения.

Возможны два способа записи в КЭШ память:

1)метод сквозной записи;

2)метод обратной записи.

Первый метод предполагает наличие двух копий данных: одной в КЭШ памяти, а другой в ОЗУ. Запись выполняется одновременно и в КЭШ и в ОЗУ.

При использовании метода обратной записи цикл записи происходит только в КЭШ памяти, если в КЭШе находится строка, к которой идет обращение (КЭШ попадание). Если адресуемой строки в КЭШ нет, то информация записывается сразу в ОЗУ. При КЭШ попадании запись в ОЗУ происходит только при замещении строки КЭШа.

8.Организация прерываний в IBM PC

К аппаратным средствам системы прерываний относятся:

1) выводы микропроцессора (INTR,INTA,NMI);

NMI - немаскируемое прерывание, используется обычно для запросов прерываний по нарушению питания;

+При возникновении аппаратного прерывания инициируется выход INT контроллера. Он напрямую соединен с входом INTR процессора. Если флаг IF=0, прерывание отбрасыватеся. Процессор опрашивает вход INTR после выполнения каждой инструкции. Как только обнаруживается сигнал, процессор сразу же подтверждает прерывание через выход INTA. Контроллер прерываний принимает сигнал INTA и выставляет на шину данных значение номера прерывания. Процессор считывает номер прерывания и входит в прерывание по описанной выше схеме.

2) программируемый контроллер прерываний 8259A (предназначен для фиксирования сигналов прерываний от восьми различных внешних устройств; выполнен в виде микросхемы; обычно используют две последовательно соединенные микросхемы, поэтому кол-во возможных источников внешних прерываний до 15) (см. рис.);

9. Организация прерываний в микроконтроллерах фирмы INTEL MCS-51

Запросы от внешних прерываний ~INT0, ~INT1 фиксируются в триг-герах IE0, IE1 Регистра Управления T/C и внешними прерываиями (TCON). Установка этих триггеров осуществляется низким уровнем на вхо-дах ~INT0, ~INT1 (если сброшены биты IT0 = 0, IT1 = 0 регистра TCON), или по фронту «1-0» (если биты установлены: IT 0 = 1, IT 1 = 1).

Запросы прерываний от Таймеров/Счетчиков фиксируются в триггерах TF0, TF1 регистра управления TCON.

Запрос прерывания последовательного порта вызывается установкой флага прерывания приемника RI или флага прерывания передатчика TI в ре-гистре SCON. В отличие от всех остальных флагов, RI и TI сбрасываются только программным путем (обычно в пределах подпрограммы обработки прерывания, где определяется: какому из флагов RI или TI соответствует прерывание).

Все перечисленные флаги прерываний : IE0, IE1, TF0, TF1, RI, TI – могут быть установлены (или сброшены) программно и вызвать соответ-ствующие прерывания.

Прерывание по каждому из перечисленных источников может быть разрешено или запрещено установкой или сбросом соответствующего бита в РЕГИСТРЕ МАСКИ (Разрешения) ПРЕРЫВАНИЙ – (IE)

РЕГИСТР ПРИОРИТЕТОВ ПРЕРЫВАНИЙ – IP предназначен для установки уровня приоритета прерывания для каждого из пяти источников прерываний :

 PS – установка уровня приоритета прерывания от последовательного порта;

 PT1 – установка уровня приоритета прерывания от T/C 1;

 PX1 – установка уровня приоритета прерывания от внешнего источника ~INT1;

 PT0 – установка уровня приоритета прерывания от T/C 0;

 PX0 – установка уровня приоритета прерывания от внешнего источника ~INT0.

Наличие в разряде регистра IP «1» устанавливает для соответствую-щего источника высокий уровень приоритета, а наличие «0» – низкий уро-вень приоритета.

Программа обработки прерывания с низким уровнем приоритета мо-жет быть прервана запросом прерывания с высоким уровнем приоритета, но не может быть прервана другим запросом прерывания с низким уровнем приоритета. Программа обработки прерывания с высоким уровнем приори-тета не может быть прервана никаким другим запросом прерывания.

Если два запроса с разными уровнями приоритета приняты одновре-менно, сначала будет обслужен запрос с высоким уровнем приоритета.

Если одновременно приняты запросы с одинаковым уровнем приори-тета, обработка их будет производиться в порядке, задаваемом последова-тельностью внутреннего опроса флагов прерываний. Таким образом, в пре-делах одного приоритетного уровня существует еще одна структура приори-тетов (табл. 15):

При переходе по вектору на подпрограмму обработки прерывания ап-паратно запрещаются все прерывания с уровнем приоритета, равным (или меньшим) уровню приоритета обслуживаемого прерывания.

Подпрограмма обслуживания прерывания должна заканчиваться вы-полнением команды RETI, которая восстанавливает состояние логики пре-рывания и загружает из стека в счетчик команд (PC) адрес возврата в ис-ходную программу. При использовании команды RET восстанавливается только счетчик команд (PC) из стека. Состояние логики прерывания команда RET не меняет, т. е. сохраняется запрет на прерывания с равным (или мень-шим) приоритетом.

Больше можно узнать в методе к первой лабе!!!

10.Очередь команд процессоров фирмы INTEL. Назначение, размер и особенности работы

Новым элементом архитектуры МП I8086/88, по сравнению со схе-мой на рис. 1.2, является Очередь Команд (ОК), которая представляет со-бой набор байтовых регистров и выполняет роль регистра команд. Коды команд, выбираемые из памяти через Буфер Данных, записываются в Очередь Команд, длиной 6 байт, что соответствует максимально длинному формату команды. Наличие Очереди Команд позволяет совместить во времени выпол-нение команды в АЛУ и выборку следующей команды из памяти. Выборка команд приостанавливается в случаях, когда очередь команд заполнена или МП обменивается данными с памятью (или портами). Таким образом, достигается высокая плотность загрузки шины и повышение скорости вы-полнения программы.

По составу и назначению В РЕАЛЬНОМ РЕЖИМЕ регистры I80286 в основном совпадают с регистрами I8086/88. Изменения касаются назначения бит регистра флагов и использования сегментных регистров в защищенном режиме. Как и I8086, процессор I80286 имеет 16-ти битовую шину данных и очередь команд 6 байт. За счет улучшения архитектуры сокращено среднее количество тактов выполнения операций.

В устройстве предвыборки имеется очередь команд, которая хранит 16 байт команд (у процессоров i486 – 32 байта очереди команд). Код команды, считываемый из очереди, подается в устройство ДЕШИФРИРОВАНИЯ КОМАНД, а смещение (т.е. константа в команде) подается в УСТРОЙСТВО СЕГМЕНТАЦИИ, где участвует в вычислении адреса.

УСТРОЙСТВО ДЕШИФРИРОВАНИЯ КОМАНД получает коды (байты префиксов, коды операций, байты адресации и др.) от устройства предвыборки и в двухступенчатом процессе преобразует их в управляющие сигналы низкого уровня и точки входа микрокода т.е. микропрограмм в ПЗУ УСТРОЙСТВА УПРАВЛЕНИЯ.

УСТРОЙСТВО СЕГМЕНТАЦИИ преобразует логический (или виртуальный) адрес в ЛИНЕЙНЫЙ АДРЕС, привлекая дескрипторы сегментов и смещения, выделенные из команд. Параллельно с вычислением линейного адреса производится контроль атрибутов сегментов.

11. Последовательные порты в IBM PC

В состав IBM PC могут входить до четырех последовательных интерфейсов, работающих в стандарте RS-232 (отечественный аналог СТЫК–С2) и именуемых COM1 – COM4. Им выделены следующие адреса в области портов ввода-вывода:

COM1: 3F8h-3FFh COM3: 338h-33Fh

COM2: 278h-2FFh COM4: 238h-23Fh

(интерфейсы COM3 и COM4 поддерживаются только в моделях PS/2). Каждый интерфейс связан с определенным уровнем контроллера прерываний:
COM1 вызывает прерывание IRQ4 (Int 0Ch)
COM2 вызывает прерывание IRQ3 (Int 0Bh)
COM3 и COM4 не имеют стандартных векторов прерываний.
Каждое из устройств RS-232 представляет собой контроллер i8250, оснащенный 25- или 9- штырьковым разъемом на задней стенке корпуса ПЭВМ. Этот разъем может использоваться для подключения мыши, графопостроителя, организации связи между ПЭВМ или с другими устройствами. Контакты стыка RS-232 (COM-порта) описаны в табл. 4.5.
При отсутствии передачи – на выходе последовательного порта TxD устанавливается сигнал «логической 1» (напряжение –12В). Передача каждого символа (от 5 до 8 бит) начинается со Старт-бита, имеющего всегда нулевой логический уровень – напряжение +12В (рис. 4.8). Символ передается «младшим битом – вперед».

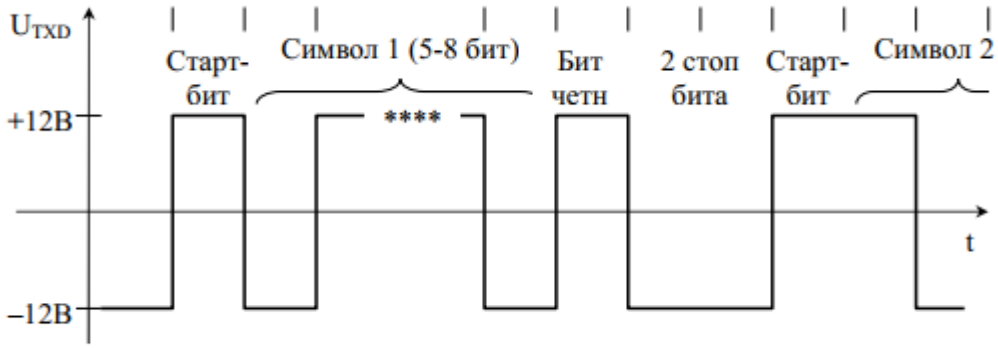


Рис. 4.8– *Передача сигналов через COM-порт*

В конце каждого символа можно запрограммировать передачу бита контроля на четность (или на нечетность). Завершается передача символа одним или двумя Стоп-битами (для 5-ти битового символа вместо двух Стоп-битов – используют 1,5). После этого может передаваться Старт-бит следующего символа (а может и не передаваться).

Таблица 4.5 – *Назначение сигналов COM-портов*

Назван. сигнала	Имя цепи		Номер контакта		Назначение	Направление
	EIA	CCITT	9-шт	25-шт		
DCD	CF	109	1	8	Связь модемов установлена	В ПЭВМ
RX	BB	104	2	3	Принимаемые данные	В ПЭВМ
TX	BA	103	3	2	Передаваемые данные	Из ПЭВМ
DTR	CD	108/2	4	20	Готовность ПЭВМ к работе	Из ПЭВМ
SG	AB	102	5	7	Сигнальная земля	
DSR	CC	107	6	6	Готовность модема к работе	В ПЭВМ
RTS	CA	105	7	4	Запрос на передачу	Из ПЭВМ
CTS	CB	106	8	5	Готовность модема к передачи	В ПЭВМ
RI	CE	125	9	22	Индикатор вызова	В ПЭВМ
FG	AA	101	--	1	Защитная земля	

Контроллер стыка RS-232 является полностью программируемым устройством; можно запрограммировать следующие параметры обмена: количество битов данных и стоп-битов, вид четности и скорость обмена в бодах (бит/с). Максимальная скорость обмена информацией по последовательному COM-порту – 115,2 кбит/с.

12. Последовательный и параллельные порты в микроконтроллерах MCS-51

Прием и выдача внешних сигналов осуществляется через 4 универсальных восьмибитовых порта P0..P3 (параллельных). Каждый порт может быть запрограммирован на ввод или вывод байта или одного бита. При обращении к внешней памяти программ (ВПП) или памяти данных (ВПД) порты P0 и P2 используются как мультиплексированная внешняя шина Адрес/Данные. Линии порта P3 могут выполнять также альтернативные функции:

Таблица 2 – *Альтернативные функции порта P3*

P3.0	Вход приемника последовательного порта – RxD;
P3.1	Выход передатчика последовательного порта – TxD;
P3.2	Вход внешнего прерывания 0 – ~INT0;
P3.3	Вход внешнего прерывания 1 – ~INT1;
P3.4	Внешний вход таймера/счетчика 0 – T0;
P3.5	Внешний вход таймера/счетчика 1 – T1;
P3.6	Выход строб. сигнала при записи в ВПД – ˜WR;
P3.7	Выход строб. сигнала при чтении из ВПД – ˜RD.

На выходе порта P2 формируются сигналы типа «бегущий нуль» для динамического управления индикаторами и сканирования столбцов матрицы клавиатуры.
При расширении памяти программ или памяти данных, кроме собственно чипов памяти, необходим дополнительный параллельный регистр (см. рис. 5) для запоминания младшего байта адреса с выходов порта P0. После этого микроконтроллер обменивается с внешней памятью одним байтом данных по линиям порта P0. Запоминать старший байт адреса с выходов порта P2 не нужно, потому что адресная информация на этих выводах не изменяется во время всего цикла обмена.

Через Универсальный Асинхронный Приемо-Передатчик (УАПП) осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена (как у COM-порта компьютера). В состав УАПП (или Последовательного Порта) входят:

- принимающий и
- передающий сдвигающие регистры, а также
- специальный буферный регистр (SBUF) приемопередатчика.

Запись байта в буфер SBUF приводит к автоматической перезаписи байта в сдвигающий регистр передатчика и инициирует начало последовательной передачи байта. Наличие буферного регистра приемника SBUF позволяет совмещать операцию чтения ранее принятого байта с последовательным приемом очередного байта. Если к моменту окончания приема байта предыдущий байт не был считан из SBUF, то он будет потерян.
Управление режимами работы УАПП определяется кодом, записанным в РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА ПОСЛЕДОВАТЕЛЬНОГО ПОРТА (SCON):

Прямой байтовый адрес SCON: dir – 98h.
Допускается адресация отдельных бит SCON : bit – 98h_9Fh.

Адрес : bit	7	6	5	4	3	2	1	0
	9Fh	9Eh	9Dh	9Ch	9Bh	9Ah	99h	98h
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0, SM1 – определяют режимы работы УАПП (см. табл. 16);
SM2 – разрешение многопроцессорной работы; (в режимах 2 и 3 при SM2 = 1 бит прерывания R1 не устанавливается, если принятый девятый бит данных RB8 = 0);

REN – разрешение ПРИЕМА последовательных данных: REN = 1 – разрешение приема, REN = 0 – запрет приема;
TB8 – девятый бит передаваемых данных в режимах 2 и 3; устанавливается и сбрасывается программно;
RB8 – девятый бит принятых данных в режимах 2 и 3;
TI – флаг прерывания передатчика; устанавливается аппаратно в конце выдачи 8-го бита в режиме 0 или в начале стоп-бита – в других режимах; сбрасывается программой;
RI – флаг прерывания приемника; устанавливается аппаратно в конце приема 8-го бита в режиме 0 или в середине стоп-бита – в других режимах; сбрасывается программой.

Таблица 16 - *Режимы работы последовательного порта УАПП*

SM0,SM1	Режим	Наименование	Скорость обмена
0 0	0	Передача и прием 8-ми битовых данных через двунаправленный вывод RxD ; через вывод TxD выдаются синхроим-пульсы сдвига	Ft / 12
0 1	1	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 8-ми битовых данных и стоп-бита (1)	Fov / 16, при SMOD=1 Fov / 32 при SMOD=0
1 0	2	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 9-ти битовых данных и стоп-бита (1)	Ft / 32, при SMOD=1 Ft / 64 при SMOD=0
1 1	3	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 9-ти битовых данных и стоп-бита (1)	Fov / 16, при SMOD=1 Fov / 32 при SMOD=0

13. Применение ассоциативных ЗУ в процессорах фирмы INTEL

Существенное сокращение времени преобразования адресов в МП 386+ достигается путем использования внутренней КЭШ-ПАМЯТИ, которая называется БУФЕРОМ АССОЦИАТИВНОЙ ТРАНСЛЯЦИИ – (TRNSLATION LOOKASIDE BUFFER – TLB).
TLB представляет собой память с ассоциативной выборкой, которая содержит 20-ти разрядные адреса 32-х страниц, т.е. старшие 20 разрядов ФИЗИЧЕСКОГО АДРЕСА.

Каждый из базовых адресов имеет свой признак – ТЭГ. В качестве тэга используются старшие 20 разрядов ЛИНЕЙНОГО АДРЕСА.
При поступлении в блок управления страницами ЛИНЕЙНОГО АДРЕСА – его старшие 20 разрядов сравниваются одновременно со всеми тэгами в TLB. Если обнаруживается совпадение этих разрядов с каким-либо тэгом, то из ячейки TLB выбирается соответствующий ему БАЗОВЫЙ АДРЕС СТРАНИЦЫ. Блок страничной адресации формирует физический адрес из старших 20-ти разрядов, считанных из ассоциативной памяти TLB, и младших 12-ти разрядов поля BYTE линейного адреса.
Случай, когда базовый адрес страницы находится в TLB, называется КЭШ-ПОПАДАНИЕМ. При этом не требуется обращаться к оперативной памяти для выборки указателей входа в таблицы и кадры страниц. Формирование физического адреса не требует дополнительных машинных циклов.
Если базовый адрес нужной страницы отсутствует в TLB, то такое обращение называется КЭШ-ПРОМАХОМ. При этом МП выполняет полную процедуру формирования физического адреса с обращениями к каталогу разделов и таблице страниц, затрачивая на это значительное время. Полученный из таблицы страниц 20-ти битовый БАЗОВЫЙ АДРЕС вместе с соответствующими 20-тью битами ЛИНЕЙНОГО АДРЕСА (ТЭГ) заносятся в свободную ячейку TLB, или занимают ячейку, в которой хранился адрес, введенный в TLB ранее других. Таким образом обеспечивается непрерывное обновление содержимого TLB.
Кроме базовых адресов в TLB хранится дополнительная информация, определяющая правила доступа к ним. Эта информация задается следующими битами:

- V (бит достоверности) – сбрасывается в состояние 0 при записи нового содержимого в регистр управления CR3 (базовый адрес каталога разделов). После преобразования очередного линейного адреса в физический – для этого адреса в TLB будет установлено значение V = 1, указывающего на допустимость использования данного базового адреса;
- D (бит мусора) – принимает значение 1 при записи на данную страницу;
- U (бит пользователя/супервизора) и W (бит записи/чтения) – аналогичны описанным ранее битам.

ПОЛНОСТЬЮ АССОЦИАТИВНАЯ АРХИТЕКТУРА КЭШ-памяти предусматривает разделение адреса ячейки памяти только на две части: младшие разряды - смещение в строке (offset) и старшие разряды - информация о тэге (tag) (см. рис. 4.4).

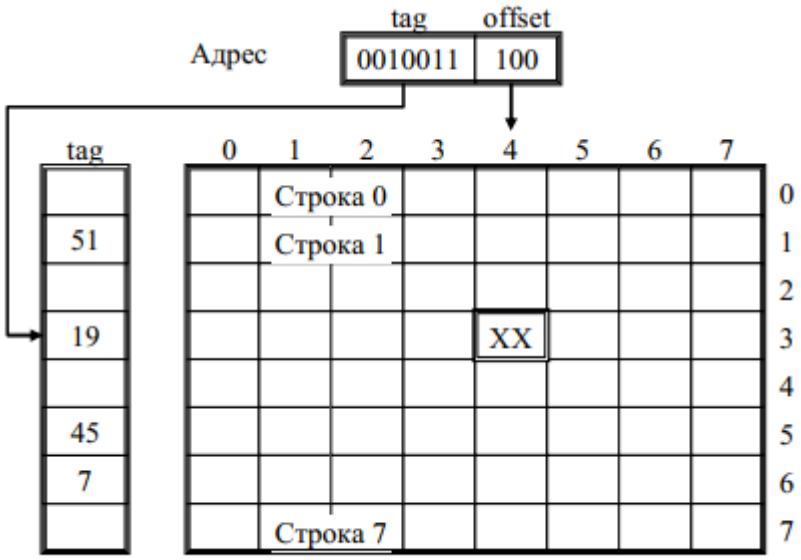


Рис. 4.4 – Полностью ассоциативная КЭШ-память

В этом случае количество разделов (блоков) основной памяти равно: 1024 / 8 = 128. Полностью ассоциативная архитектура решает проблему конфликта адресов, однако сама КЭШ-память требует для своей реализации больших аппаратных затрат, поскольку значения тэгов должны сравниваться для всех линий КЭШа.

14. Программная модель и форматы данных сопроцессоров x87

Математический сопроцессор x87 выполняет операции с плавающей точкой в 50..100 раз быстрее эквивалентных подпрограмм процессора x86.
Программисты могут включать команды сопроцессора x87 в программы для процессора x86. Когда основной процессор встречает команду сопроцессора, он передает ее для выполнения этому сопроцессору x87.
Сопроцессор x87 распознает три формата чисел с плавающей точкой, хранимых в памяти (рис. 2.6). Внутри сопроцессора все числа преобразуются в расширенный формат.
Каждый формат состоит из трех полей: знак (S), порядок и мантисса (рис. 2.6). Числа в этих форматах занимают в памяти: 4, 8 или 10 байт.
Байт с наименьшим адресом в памяти является младшим байтом мантиссы. Байт с наибольшим адресом содержит семь старших бит порядка и БИТ ЗНАКА (S). Знак кодируется: 0 – плюс, 1 – минус.
В ПОЛЕ МАНТИССЫ хранятся только нормализованные числа. Для этого необходимо скорректировать порядки (т.е. сдвинуть запятую) так, чтобы в целой части числа (в двоичной системе счисления) до запятой была 1. Поэтому все мантиссы представляются в форме:

1.XXXXXXX...XXX , где: X = 0 или 1

Рис. 2.6 – *Форматы чисел с плавающей точкой*

Но если старший бит всегда содержит 1, ее можно не хранить в каждом числе с плавающей точкой. Поэтому ради дополнительного бита точности сопроцессор x87 хранит числа одинарной и двойной точности без старшего бита мантиссы (с неявным старшим битом). Исключением является кодирование нуля – нулевые поля мантиссы и порядка.

Числа с расширенной точностью хранятся и обрабатываются внутри сопроцессора с явным старшим битом.

ПОЛЕ ПОРЯДКА определяет степень числа 2, на которую нужно умножить нормализованную мантиссу для получения исходного значения числа с плавающей точкой.

Чтобы хранить отрицательные порядки, в ПОЛЕ ПОРЯДКА находится сумма истинного порядка и положительной константы, называемой СМЕЩЕНИЕМ. Для одинарной точности смещение равно – 127, для двойной точности – 1023, для расширенной точности – 16383 (т.е. половине максимального порядка). Двоичный код смещения для всех порядков равен: 0111…111. Числа в поле порядка: 00000..00 и 11111..111 – зарезервированы для спецкодирования или обработки ошибок.

Запись чисел с плавающей точкой в памяти:

Одинарная точность: $(-1)^S (1 . X1\ X2 \ldots X23) * 2^{(E - 127)}$,

Двойная точность: $(-1)^S (1 . X1\ X2 \ldots X52) * 2^{(E - 1023)}$,

Расширенная точность: $(-1)^S (X1 . X2 \ldots X64) * 2^{(E - 16383)}$,

где S – значение знакового бита;
X1 X2 .. – биты, хранимые в поле мантиссы;
E – число, хранимое в поле порядка.

Таблица 2.23 – *Диапазон представления чисел в десятичной системе*

Формат	Значащих десятичных цифр	Наименьшая степень числа 10	Наибольшая степень числа 10
Одинарный	7	-37	38
Двойной	15	-307	308
Расширенный	19	-4931	4932

Сопроцессор x87 имеет команды, которые преобразуют целые числа в числа с плавающей точкой и – наоборот. Это необходимо при вычислениях, в которых имеются числа обоих форматов. Допустимые форматы ЦЕЛЫХ ЧИСЕЛ приведены на рис. 2.8:

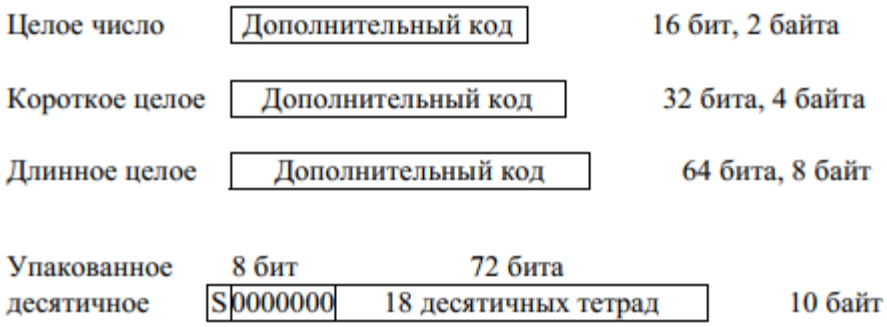


Рис 2.8 – *Форматы целых чисел*

ЦЕЛОЕ СЛОВО – это 16-ти битовое целое число процессора x86 в дополнительном коде. КОРОТКОЕ ЦЕЛОЕ и ДЛИННОЕ ЦЕЛОЕ похожи на целое слово, но их длина больше.
УПАКОВАННОЕ ДЕСЯТИЧНОЕ ЧИСЛО состоит из 18 десятичных цифр, размещенных по две в байте. Знаковый бит находится в отдельном 10-м байте. Младшие семь бит этого байта должны быть нулевыми.

15. Страничная организация памяти. Аппаратные средства поддержки страничной организации памяти

4) Страничная организация памяти. Аппаратные средства поддержки страничной организации памяти

Страничная организация виртуальной памяти

В большинстве современных операционных систем виртуальная память организуется с помощью страничной адресации. Оперативная память делится на страницы: области памяти фиксированной длины (например, 4096 байт), которые являются минимальной единицей выделяемой памяти (то есть даже запрос на 1 байт от приложения приведёт к выделению ему страницы памяти). Исполняемый процессором пользовательский поток обращается к памяти с помощью адреса виртуальной памяти, который

делится на номер страницы и смещение внутри страницы. Процессор преобразует номер виртуальной страницы в адрес соответствующей ей физической страницы при помощи буфера ассоциативной трансляции (TLB). Если ему не удалось это сделать, то требуется дозаполнение буфера путём обращения к таблице страниц, что может сделать либо сам процессор, либо операционная система (в зависимости от архитектуры). Если страница была выгружена из оперативной памяти, то операционная система подкачивает страницу с жёсткого диска. При запросе на выделение памяти операционная система может «сбросить» на жёсткий диск страницы, к которым давно не было обращений. Критические данные (например, код запущенных и работающих программ, код и память ядра системы) обычно находятся в оперативной памяти (исключения существуют, однако они не касаются тех частей, которые отвечают за обработку аппаратных прерываний, работу с таблицей страниц и использование файла подкачки).

Аппаратные средства в современных компьютерах обычно представлены специальным блоком MMU – Memory Management Unit, входящим в состав центрального процессора (CPU). Начиная с модели Intel 80386, была реализована аппаратная поддержка как сегментной, так и страничной организации памяти. При этом блок MMU был разделен на две относительно независимые части в виде SU – Segment Unit и PU – Page Unit.

16. Структуры и программные модели микроконтроллеров MCS-51 Общие характеристики

Микроконтроллеры Intel семейства MCS-51 (8051, 80C51, 8052 87C51) и соответствующие отечественные однокристальные микроЭВМ семейства МК51 (КР1816ВЕ51, КМ1816ВЕ751) имеют:

- внутреннее ОЗУ объемом 128 или 256 байт;
- четыре двунаправленных побитно настраиваемых восьмиразрядных
- порта ввода-вывода;
- два 16-разрядных таймера-счетчика;
- встроенный тактовый генератор;
- адресация 64 КБайт памяти программ и 64 Кбайт памяти данных;
- две линии запросов на прерывание от внешних устройств;

- интерфейс для последовательного обмена информацией с другими
- микроконтроллерами или персональными компьютерами.
- Микроконтроллер 8751 снабжен УФ ПЗУ объемом 4 Кбайт.

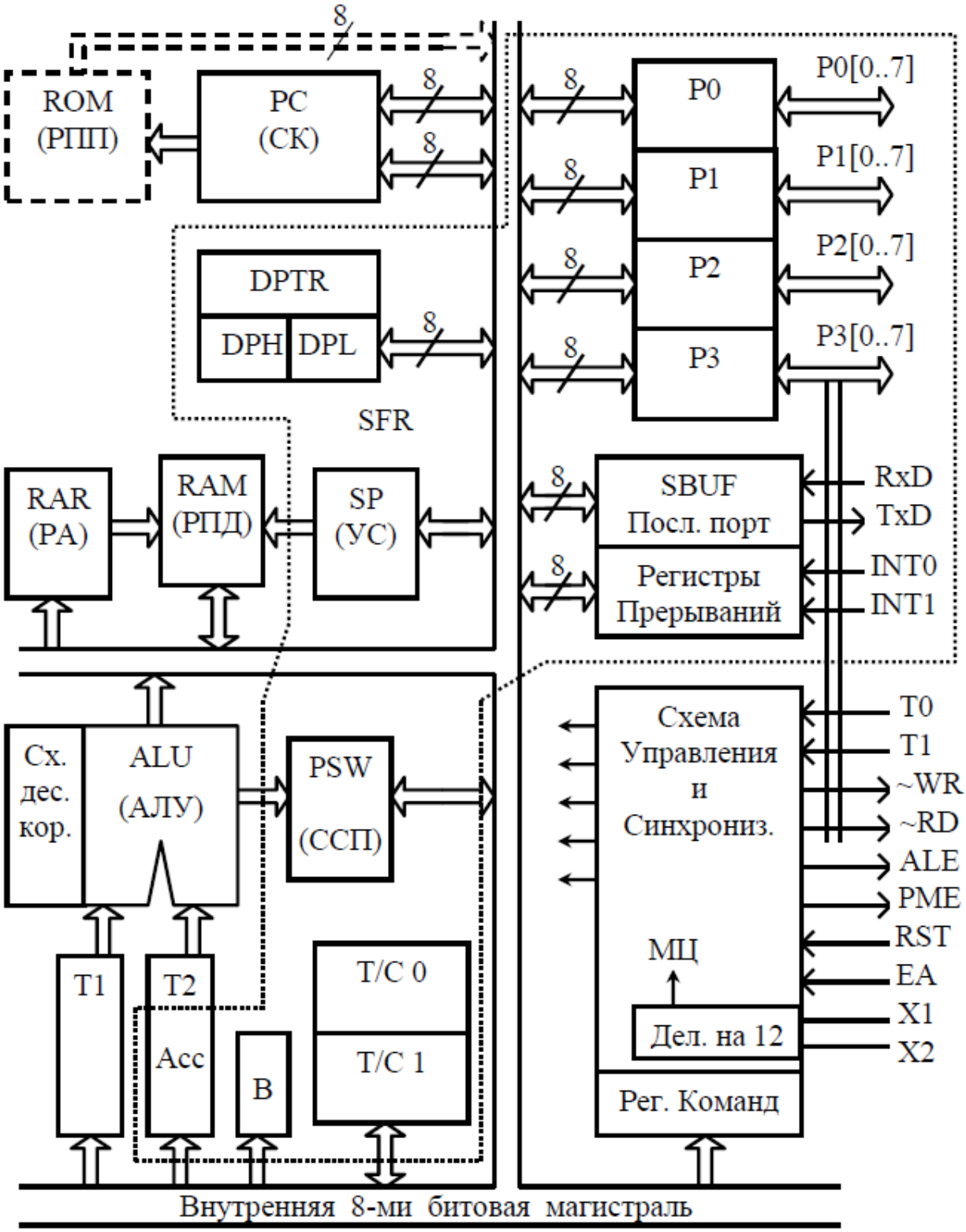


Рис. 1 – *Структурная схема микроконтроллеров семейства MCS-51*

17 Структуры и программные модели микропроцессоров фирмы INTEL

Базовая архитектура 32-х разрядных процессоров (обозначаемых: 386+) является общей для существующих на данный момент процессоров фирмы INTEL – 386, 486, PENTIUM и его модификаций. МП состоит из 3 основных частей:

- устройства обработки;
- устройства управления ЗУ;
- интерфейсного блока.

УСТРОЙСТВО ОБРАБОТКИ состоит из исполнительного устройства (операционной части) и блока команд (управляющей части). Содержит 8 32-х разрядных РОН, 64-х битовый циклический сдвигатель. Умножение и деление осуществляется на 1 бит за цикл. Алгоритм умножения такой, что процесс прекращается, когда наиболее значащий бит, умножается на все нули. Типичное время умножения 32-х разрядных чисел около 1 мкс (для процессора I80386).

УСТРОЙСТВО УПРАВЛЕНИЯ ЗУ состоит из сегментного и страничного блоков. Сегментный блок позволяет работать с логическими адресами со всеми вытекающими отсюда преимуществами. Страничная организация используется внутри сегмента и управляет физическими адресами. Каждая задача может иметь до 16381 (214) сегмента до 4 Гбайт каждый (2^{32}), т.е. виртуальная память может быть размером 64 Тбайт (2^{46}).

ИНТЕРФЕЙСНЫЙ БЛОК обеспечивает взаимодействие с внешними устройствами, включая автоматическое управление разрядностью шины, и формирование сигналов активности байтов.

МП 386+ могут функционировать в трех режимах:

➢ **REAL ADDRESS MODE** – режим реальной адресации (РРА) – характеризуется тем, что МП работает как очень быстрый 8086 с 32-битовым расширением; в этом режиме возможна адресация 1 Мбайта физической памяти (на самом деле, как у I80286, - почти на 64 Кбайта больше);

➢ **PROTECTED ADDRESS MODE** – режим защищенной виртуальной адресации (РВА) – реализует все достоинства МП (режим параллельного выполнения нескольких задач несколькими 8086 – по одному на задачу). На одном процессоре в таком режиме могут одновременно исполняться несколько задач с изолированными друг от друга реальными ресурсами. При этом использование физического адресного пространства памяти управляется механизмами сегментации и трансляции страниц. Попытки выполнения

недопустимых команд, выхода за рамки отведенного пространства памяти и разрешенной области ввода-вывода контролируются системой защиты.

ПРОГРАММНАЯ МОДЕЛЬ 32-х РАЗРЯДНЫХ ПРОЦЕССОРОВ (386+)

МП 386+ имеет 31 регистр (у PENTIUM+ – 32 регистра), разбитые на следующие группы:

- регистры общего назначения;
- сегментные регистры;
- указатель команд и регистр флагов (признаков);

РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ

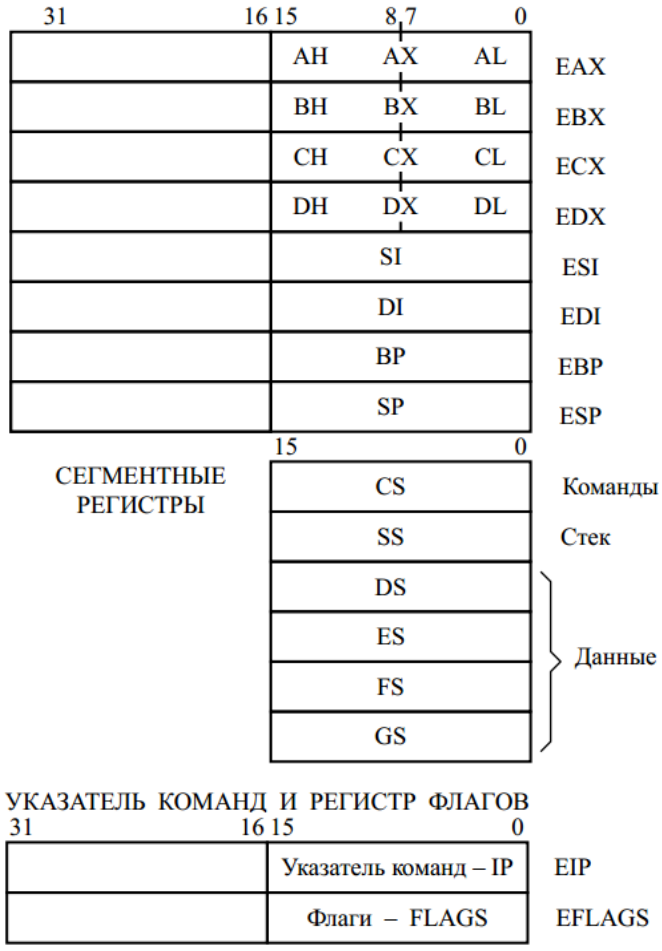


Рис. 1 – *Регистры 32-х разрядных МП (386+)*

4

- управляющие регистры;
- регистры системных адресов;
- отладочные регистры;
- тестовые регистры.

Набор **РЕГИСТРОВ ОБЩЕГО НАЗНАЧЕНИЯ** (рис. 1) включает соответствующие регистры процессоров I8086 и I80286. Все эти регистры,

кроме сегментных, имеют разрядность 32 бита и к прежнему обозначению их имен добавилась приставка «E» (Extended – расширенный). Отсутствие приставки «E» в имени означает ссылку на младшие 16 бит расширенных регистров. Обратиться к старшим 16-ти битам расширенных регистров ни одна команда не может. Как и в I8086, возможно независимое обращение к младшему и старшему байтам регистров AX, BX, CX, DX.
Архитектура МП 386+ позволяет непосредственно обращаться к 6 сегментам (размером до 4 Гбайт каждый) при помощи специальных селекторов, которые загружаются в сегментные регистры программно. Содержимое РОНов, селекторов, указателя команд и регистра флагов (признаков) зависит от выполняемой задачи и автоматически перегружается при переключении задач.
Остальные регистры МП используются, главным образом, для упрощения проектирования и отладки операционной системы.
Регистры общего назначения (РОН) – используются для хранения операндов и адресов. Могут работать с операндами, имеющими длину 1, 8, 16, 32 и 64 бита или с битовыми полями длиной от 1 до 32 бит.
УКАЗАТЕЛЬ КОМАНД – EIP – хранит смещение, которое всегда складывается со значением кодового сегментного регистра (CS) и определяет адрес следующей команды. При 16-ти битовой адресации используются только младшие 16 бит (IP).

РЕГИСТР ФЛАГОВ (признаков) – EFLAGS (рис. 2) – отражает состояние МП. При использовании только 16-ти младших разрядов – регистр флагов совместим с предыдущими моделями МП

18. Таймер 8254 в структуре IBM PC. Режимы и программирование

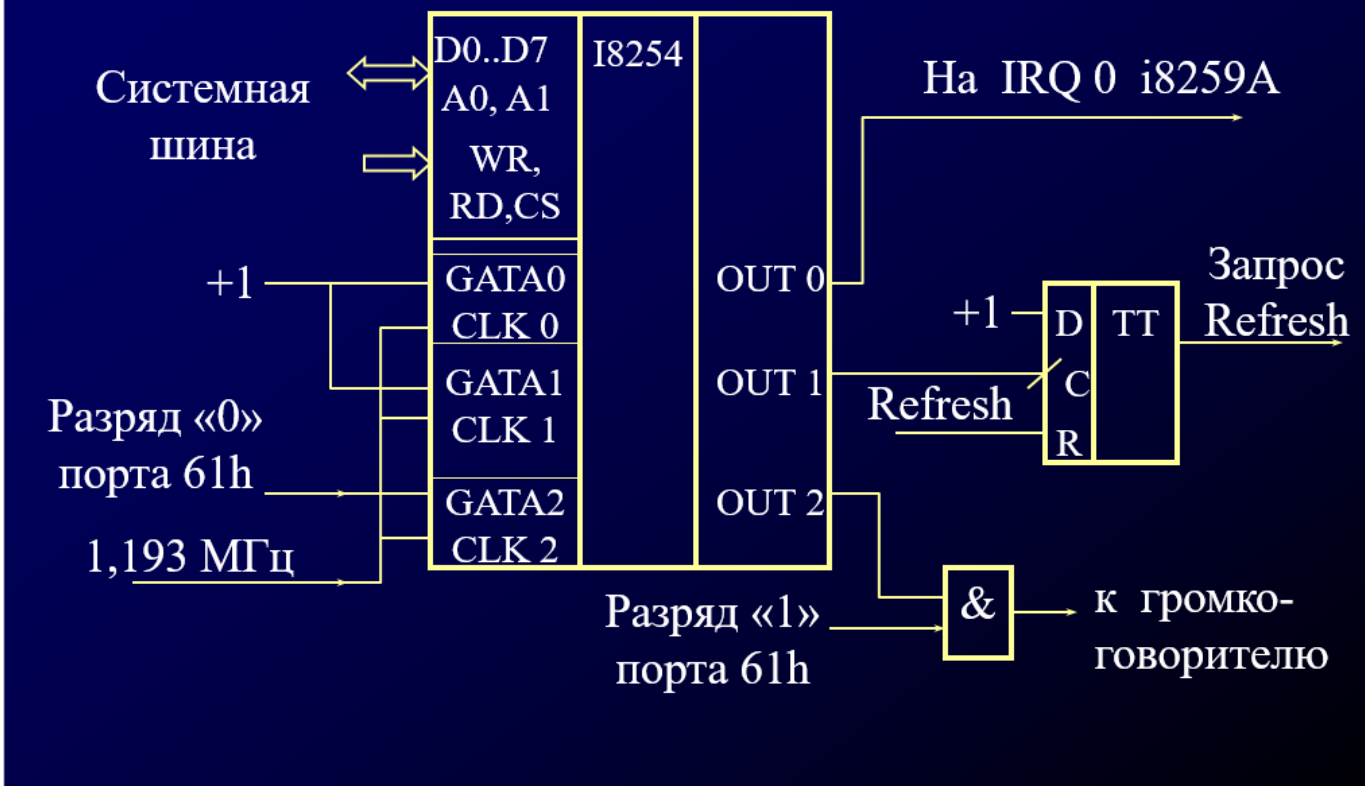
Таймер i8254 в архитектуре IBM PC AT+

- Архитектура **IBM PC AT+** использует подсистему трехканального 16-ти разрядного **ПРОГРАММИРУЕМОГО ТАЙМЕРА i8254** в качестве системного таймера:
  - ➢ Выход **КАНАЛА 0 (OUT0),** использующего **режим 3**, связан с **IRQ0** контроллера прерываний **i8259A** и обеспечивает прерывание системных часов реального времени с частотой **18,2 Гц**.
  - ➢ Выход **КАНАЛА 1 (OUT1),** использующего **режим 2**, генерирует сигнал запроса **регенерации динамической RAM**.
  - ➢ Выход **КАНАЛА 2**, используя **режим 3**, генерирует тональный сигнал для **громкоговорителя**.

Порты, используемые для программирования таймера

Адрес порта	Функция
40h	Чтение или запись счета для канала 0 (системные часы)
41h	Чтение или запись счета для канала 1 (запрос регенерации)
42h	Чтение или запись счета для канала 2 (тональный сигнал громкоговорителя)
43h	Запись управляющего слова и команд CLC, RBC

Структурная схема подсистемы таймера в IBM PC



19.Таймеры в микроконтроллерах фирмы INTEL MCS-51

4.1 ТАЙМЕРЫ / СЧЕТЧИКИ ВНЕШНИХ СОБЫТИЙ

Два программируемых 16-ти битовых таймера/счетчика (T/C 0 и T/C 1) могут быть использованы в качестве таймеров или счетчиков внешних событий. При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т. е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое T/C инкрементируется под воздействием перехода "1-0" внешнего входного сигнала, подаваемого на соответствующий вывод (T0 или T1). Максимальная входная частота счетчиков: Ft / 24.

РЕЖИМЫ РАБОТЫ T/C определяются кодом, записанным в РЕГИСТР РЕЖИМОВ T/C (TMOD):

РЕГИСТР РЕЖИМОВ ТАЙМЕРОВ / СЧЕТЧИКОВ (TMOD)

Прямой адрес TMOD: dir – 89H.

	7	6	5	4	3	2	1	0
TMOD :	GATE 1	C/~T 1	M1.1	M0.1	GATE 0	C/~T 0	M1.0	M0.0

Таблица 13 – *Выбор режимов таймеров / счетчиков (*TMOD*)*

Биты TMOD	Обозначение	Выбор режима
0, 1 4, 5	M0, M1	Определяют один из 4-х режимов работы, отдельно для T/C 1 и T/C 0 :
2 6	C/~T 0 C/~T 1	Определяет работу отдельно для каждого счетчика в режиме : C/~T = 0 – таймера; C/~T = 1 – счетчика внешних событий.
3 7	GATE 0 GATE 1	Разрешает управлять счетчиком от внешнего вывода (~INT0 – для T/C 0, ~INT1 – для T/C 1) : GATE = 0 – управление запрещено, GATE = 1 – управление разрешено.

M1	M0	Режим
0	0	0
0	1	1
1	0	2
1	1	3

РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА Т/С и внешними прерываниями (TCON) предназначен для приема и хранения кодов управляющего слова.

РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА Т/С И ВНЕШНИМИ ПРЕРЫВАНИЯМИ (TCON)

Прямой байтовый адрес TCON: dir – 88h.
Допускается адресация отдельных бит TCON : bit – 88h_8Fh.

	7	6	5	4	3	2	1	0
Адрес : bit	8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h
TCON	TF 1	TR 1	TF 0	TR 0	IE 1	IT 1	IE 0	IT 0

Таблица 14 – *Назначение битов TCON*

Биты TMOD	Обозначение	Назначение разрядов TCON
5 7	TF 0 TF 1	Флаги переполнения Т/С, устанавливаются аппаратно при переполнении соответствующего Т/С (переходе из состояния «все единицы» в состояние «все нули»). Если прерывание от соответствующего Т/С разрешено, то установка флага TF вызовет прерывание. Флаги TF 0 или TF 1 сбрасываются аппаратно при передаче управления подпрограмме обработки соответствующего прерывания
4 6	TR 0 TR 1	Разрешение счета отдельно для каждого Т/С : TR = 0 – счет остановлен, TR = 1 – разрешение счета.
1 3	IE 0 IE 1	Флаги запроса внешних прерываний по входам ˜INT0 и ˜INT1 соответственно; устанавливаются аппаратно (от внешних устройств) или программно и вызывают подпрограмму обработки прерываний. Если прерывание вызвано по фронту сигнала, эти флаги сбрасываются аппаратно при переходе к подпрограмме. Если прерывание было вызвано низким уровнем на входе ~INT0 (~INT1), то сброс флага должна выполнять подпрограмма обслуживания прерывания, воздействуя на источник прерывания для снятия запроса.
0 2	IT 0 IT 1	Управление видом прерывания отдельно по входам ˜INT 0 или ˜INT 1 : IT = 0 – прерывание по уровню (низкому), IT = 1 – прерывание по фронту «1–0»

РЕЖИМ РАБОТЫ «0» (M0=0, M1=0) функционально совместим с таймером/счетчиком микроконтроллера MCS-48. Деление импульсов Машинных Циклов (МЦ) на 32 выполняют 5 младших разрядов регистров TL 0, TL 1.

Логика работы в РЕЖИМЕ 0 на примере Т/С 0 показана на рис. 6 Для Т/С 1 логика работы аналогична.

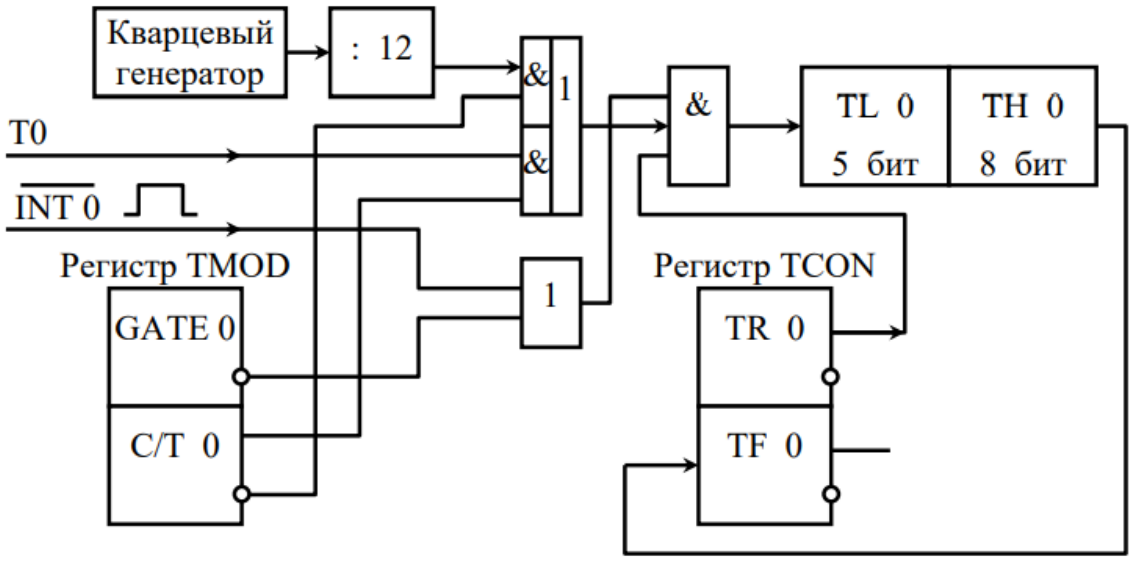


Рис. 6 – *Логика работы Т/С 0 в РЕЖИМЕ 0 (в РЕЖИМЕ 1 TL 0 – 8 бит)*

Счет начинается при установке бита TR 0 регистра TCON в состояние «1». (Если бит TR = 0, то регистры соответствующих таймеров/ счетчиков TH и TL могут использоваться как дополнительные РОНы).

Установка бита GATE в единичное состояние позволяет в режиме внутреннего таймера измерять длительность импульсного сигнала, подаваемого на вход внешнего прерывания ~INT.

РЕЖИМ РАБОТЫ «1» (M0=1, M1=0) аналогичен РЕЖИМУ «0». Отличие состоит в том, что таймерные регистры TL, TH - 16-ти разрядные.

РЕЖИМ РАБОТЫ «2» (M0=0, M1=1) представляет собой 8-ми разрядный делитель TL 0 (или TL 1) с переменным (программируемым) коэффициентом деления. При каждом переполнении 8-ми разрядного счетчика TL 0 устанавливается флаг TF 0 и происходит перезагрузка счетчика TL 0 из регистра TH 0 (рис. 7). Для T/C 1 логика работы аналогична.

РЕЖИМ РАБОТЫ «3» различный для T/C 0 и T/C 1.

Счетчик T/C 1 бессмысленно программировать в режиме «3», потому что он будет заблокирован (сохраняет свое текущее значение).

Счетчик T/C 0 в РЕЖИМЕ «3» представляет собой два независимых 8-ми разрядных счетчика TL 0 и TH 0.

TL 0 может работать в режиме таймера и в режиме счетчика. За ним сохраняются все биты управления T/C 0 и входные сигналы T0, ~INT0 (см. рис. 8). TH 0 может работать только в режиме таймера, использует бит включения TR 1 и выставляет флаг переполнения TF 1 (рис. 8).

Этот режим позволяет реализовать два восьмибитовых таймера из T/C 0, если T/C 1 уже занят – формирует частоту обмена для последовательного интерфейса (последовательного порта).

20. Типы микросхем памяти, применяемых в IBM PC (ПЗУ, ОЗУ: динамическая, статическая, CMOS-setup)

Вопрос оч большой, так что читай презенташку: АК_13_ОЗУ_ПЗУ_Кэш

АК_13__ОЗУ_ПЗУ_Кэш (3).ppt

21. Форматы данных и методы адресации микроконтроллеров MCS-51

2.3 МЕТОДЫ (СПОСОБЫ) АДРЕСАЦИИ MCS–51

1. РЕГИСТРОВАЯ АДРЕСАЦИЯ – 8–ми битовый операнд находится в РОНе выбранного (активного) банка регистров;
2 НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ (обозначается знаком – #) – операнд находится во втором (а для 16–ти битового операнда и в третьем) байте команды;
3 КОСВЕННАЯ АДРЕСАЦИЯ (обозначается знаком – @) – операнд находится в Памяти Данных (РПД или ВПД), а адрес ячейки памяти содержится в одном из РОНов косвенной адресации (R0 или R1); в командах PUSH и POP адрес содержится в указателе стека SP; регистр DPTR может содержать адрес ВПД объемом до 64К;
4 ПРЯМАЯ БАЙТОВАЯ АДРЕСАЦИЯ – (dir) – используется для обращения к ячейкам РПД (адреса 00h…7Fh) и к регистрам специальных функций SFR (адреса 80h…0FFh);
5 ПРЯМАЯ БИТОВАЯ АДРЕСАЦИЯ – (bit) – используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках РПД по адресам 20H…2FH и к отдельно адресуемым битам регистров специальных функций (см. табл. 3 и программную модель);
6 КОСВЕННАЯ ИНДЕКСНАЯ АДРЕСАЦИЯ (обозначается знаком – @)– упрощает просмотр таблиц в Памяти Программ, адрес ПП определяется по сумме базового регистра (PC или DPTR) и индексного регистра (Аккумулятора);
7 НЕЯВНАЯ (ВСТРОЕННАЯ) АДРЕСАЦИЯ – в коде команды содержится неявное (по умолчанию) указание на один из операндов (чаще всего на Аккумулятор).

22. Форматы данных и методы адресации микропроцессоров фирмы INTEL

Новые команды процессоров 386+ поддерживают БИТОВЫЕ ДАННЫЕ:

- БИТ – одиночный двоичный разряд.
- БИТОВОЕ ПОЛЕ – группа до 32-х битов.
- ЦЕПОЧКА БИТОВ (СТРОКА) – набор последовательных битов, длиной до 4 Гбит.

8

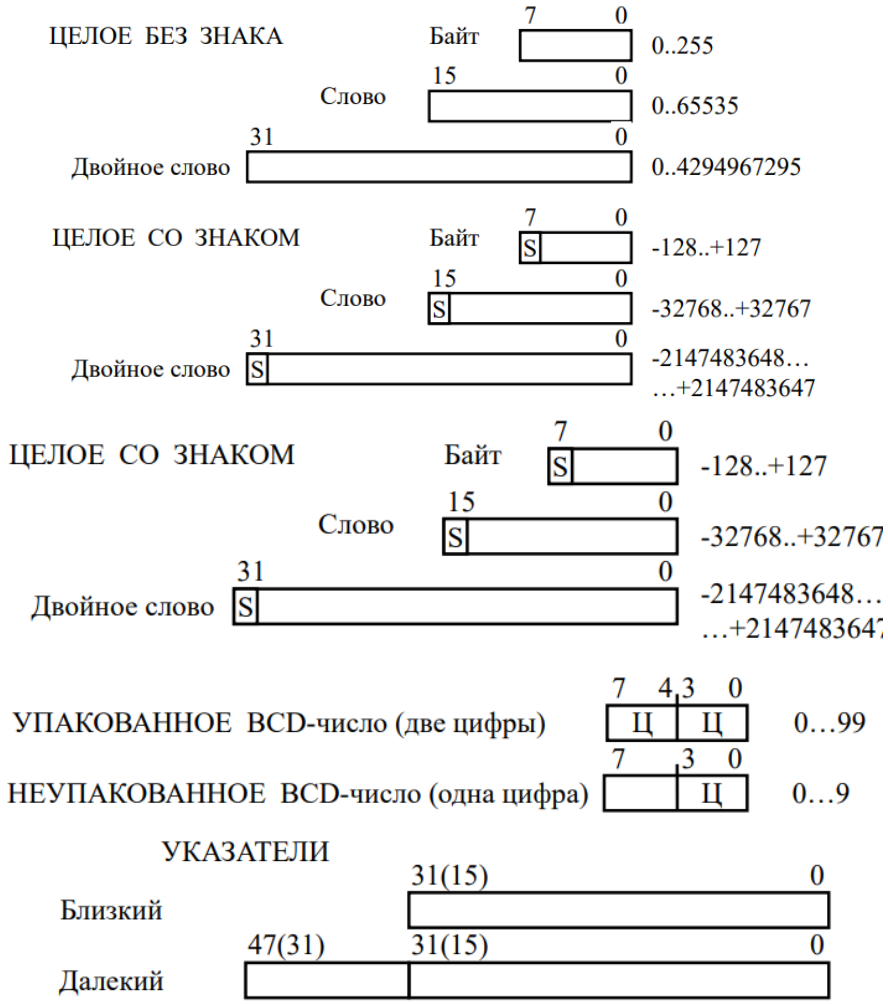


Рис. 4 – *Типы данных 32-х разрядных процессоров (386+)*

Процессор может легко оперировать с цепочками бит, байт, слов и двойных слов. Под ЦЕПОЧКОЙ (string) понимается последовательность практически любой длины отдельных, но взаимосвязанных элементов данных, ХРАНЯЩИХСЯ ПО СОСЕДНИМ АДДРЕСАМ. УКАЗАТЕЛИ применяются для обращения к некоторым объектам в памяти, например, адресам подпрограмм. Близкие (NEAR) или внутрисегментный указатель (см. рис. 4) – это 16-ти битовое или 32-х битовое смещение внутри текущего сегмента. Далекий (FAR) или межсегментный указатель применяется в тех случаях, когда программа осуществляет передачу управления в другой сегмент. Такой указатель определяет новый сегмент (с помощью селектора) и 16-ти или 32-х битовое смещение

внутри этого сегмента. При размещении операндов в памяти необходимо учитывать, что процессоры 386+ не накладывают ограничения на размещение данных. Однако производительность процессора повышается, если слова размещены по четным адресам, а двойные слова – по адресам, кратным четырем. Такой принцип называется ВЫРАВНИВАНИЕ АДРЕСОВ по границам слов или двойных слов. Выравнивание особенно важно для стека, который работает только со словами или двойными словами.

ВНИЗ ˅

3.1 РЕЖИМЫ (МЕТОДЫ) АДРЕСАЦИИ

Процессоры 386+ обеспечивает 13 режимов адресации, которые рассчитаны на эффективное выполнение программ, написанных на языках высокого уровня (ЯВУ) типа: C++, Фортран и др..

НЕЯВНАЯ АДРЕСАЦИЯ. Операнд адресуется неявно, если в команде нет специальных полей для его определения, т.е. операнд задается

полем команды. В ассемблерных кодах с неявной адресацией поле операнда пустое. Примеры команд с неявной адресацией:

```
AAA     ; Коррекция регистра AL после сложения
CMC     ; Инверсия флага переноса
STD     ; Установить в 1 флаг направления.
```

РЕЖИМ РЕГИСТРОВОЙ АДРЕСАЦИИ и
РЕЖИМ НЕПОСРЕДСТВЕННОЙ АДРЕСАЦИИ – предназначены, соответственно, для адресации одного из регистров регистрового блока или непосредственного операнда в команде с разрядностью 8, 16 или 32 бита :

```
INC     esi         ; Инкремент регистра ESI
SUB     ECX, ECX    ; Сбросить регистр ECX
MOV     EAX, CR0    ; Передать в EAX содержимое CR0.

MOV     EAX, 0F0F0F0F0h     ; Загрузить константу в EAX
AND     AL, 0FH     ; Выделить младшую тетраду регистра AL
BT      EDI, 3      ; Передать во флаг CF третий бит
                    ; регистра EDI
```

Имеется 10 режимов АДРЕСАЦИИ ПАМЯТИ. Исполнительный адрес включает в себя два компонента адреса ячейки памяти – сегмент и эффективный адрес (внутрисегментное смещение). ЭФФЕКТИВНЫЙ АДРЕС (EA) вычисляется суммированием следующих элементов :

- СМЕЩЕНИЕ (отклонение) – целая 8-ми или 32-х битовая величина со знаком, непосредственно задаваемая в команде (16-ти битовые отклонения могут использоваться при помощи префикса);
- БАЗА – содержимое любых РОНов. Базовые регистры обычно используются компиляторами в качестве точки отсчета локальной области памяти;
- ИНДЕКС – содержимое любых РОНов, исключая ESP. Индексные регистры используются для доступа к элементам строк или массивов.
- МНОЖИТЕЛЬ f - указывает шаг (1, 2, 4 или 8) для индексного регистра. Шаг индексации позволяет успешно адресовать массивы или структуры, содержащие многобайтовые операнды.

EA = БАЗА + ИНДЕКС * (ШАГ ИНДЕКСАЦИИ) + ОТКЛОНЕНИЕ.

Вычисление эффективного адреса (EA) практически не ухудшает производительность процессора из-за использования конвейерного режима.

РЕЖИМЫ АДРЕСАЦИИ ПАМЯТИ :

ПРЯМАЯ АДРЕСАЦИЯ – смещение (отклонение) адреса операнда содержится в 8, 16 или 32 разрядах команды :

```
MOV     AL, [2000h]          ; Передать байт в регистр AL
```



```
INC     dword prt [123456h]  ; Инкремент двойного слова
                             ; в памяти.
```

РЕГИСТРОВЫЙ КОСВЕННЫЙ МЕТОД АДРЕСАЦИИ – базовый или индексный регистр содержат адрес операнда :

```
MOV     AL, [ECX]        ; Передать в AL байт по адресу из ECX
DEC     word prt [ESI]   ; Декремент слова по адресу из ESI.
```

БАЗОВАЯ АДРЕСАЦИЯ – базовый регистр суммируется с отклонением:

```
MOV     EAX, [EBX+4]     ; Передать двойное слово из памяти
ADD     [ECX+10h], DX    ; Прибавить к слову в памяти.
```

ИНДЕКСНАЯ АДРЕСАЦИЯ – индексный регистр (любой РОН кроме ESP) суммируется с отклонением :

```
SUB     array[ESI], 2    ; Вычесть 2 из элемента массива
IMUL    vector[ECX]      ; Умножить EAX на элемент массива.
```

ИНДЕКСНАЯ АДРЕСАЦИЯ С ШАГОМ – содержимое индекс-ного регистра умножается на шаг «f» и суммируется с отклонением :

```
MOV     EAX, vec[ECX*4]          ; Переслать в EAX двойное слово
                                 ; из массива.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ. – EA = БАЗА + ИНДЕКС :

```
ADD     EAX, [EBX][ESI]          ; Прибавить к EAX двойное
                                 ; слово  из памяти.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ШАГОМ. – EA = БАЗА + ИНДЕКС * ШАГ:

```
INC     word prt [EDX][EDI*4]  ; Инкремент ячейки памяти.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ОТКЛОНЕНИЕМ. – EA = БАЗА + ИНДЕКС + ОТКЛОНЕНИЕ:

```
MOV     AX, [ECX][ESI+20h]     ; Переслать слово из памяти
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ОТКЛОНЕНИЕМ И С ШАГОМ. – EA = БАЗА + ИНДЕКС * ШАГ + ОТКЛОНЕНИЕ:

```
ADD     AX, [EDX][EDI*4+10h]; Сложить AX с ячейкой памяти.
```

СТЕКОВАЯ АДРЕСАЦИЯ (можно рассматривать как вариант регистровой косвенной адресации) – в указателе стека ESP (SP) формируется 32-х битовое (16-ти битовое) внутрисегментное смещение для операнда в стековом сегменте :

```
PUSH    ECX     ; Включить в стек содержимое регистра
PUSHFD          ; Включить в стек содержимое EFLAGS
```



```
PUSH    4000h   ; Включить в стек константу
POP     EDX     ; Извлечь из стека в регистр
POPFD           ; Извлечь из стека в регистр EFLAGS
POP     [ESI]   ; Извлечь из стека в ячейку памяти
```

В таблице 1 показана разница в использовании базовых и индексных регистров для 16-ти и 32-х битовых адресов.

Для обеспечения совместимости ПО процессоров необходимо программы (с 16-ти битовыми командами МП 86 и 286) выполнять на МП 386+ в реальном или защищенном режимах. Процессор определяет размерность адреса, анализируя бит **D** (Default) в дескрипторе сегмента. Если D=0, то все длины операндов и эффективных адресов составляют 16 бит. Если D=1, – 32 бита. В реальном режиме – 16 бит.

Изменение размерности адреса и данных, задаваемых битом **D**, обеспечивают два префикса, выбираемые перед командами:

- ПРЕФИКС РАЗМЕРНОСТИ ОПЕРАНДА (OperandSize),
- ПРЕФИКС ДЛИНЫ АДРЕСА (AddressSize).

Наличие префикса коммутирует (переключает) размер операнда или размер эффективного адреса на значение, противоположное принимаемому по умолчанию (по биту **D**).

Префиксы могут использоваться совместно с любой инструкцией и в любом режиме – реальном, виртуальном и V86. Префикс длины адреса не обеспечивает размерность адреса более 64 Кбайт в режиме реальной адресации. Адрес свыше 0FFFFh будет рассматриваться как ошибка.

Таблица 1 – *Базовые и индексные регистры для 16-ти и 32-х битовых адресов*

	16-ти битовый адрес	32-х битовый адрес
Базовый регистр	BX, BP	Любой 32-х битовый РОН
Индексный регистр	SI, DI	Любой 32-х битовый РОН, исключая ESP
Шаг индексации «f»	нет	1, 2, 4, 8
Смещение	0, 8, 16 бит	0, 8, 32 бит