**ПЛАН ЗАНЯТИЯ**

**Дисциплина:** МДК.01.04 Системное программирование

**Преподаватель:** Галузин А.Б.

**Курс:** 4

**Группа:** П-40

**Специальность:** 09.02.07 Информационные системы и программирование

**Дата:** 17.09.24

**Время проведения:** 11.50-13.25, 3 пара

**Тема:** Машинный язык и язык ассемблера. Программно-аппаратная архитектура IA-32 процессоров Intel

**Цель занятия:**

**дидактическая:** изучить архитектуру IA-32

**развивающая**: развивать абстрактное мышление, логику

**Вид занятия** лекция

**Литература**

Юров В.И. Assembler. Учебник для вузов. - 2-ое изд. – СПб.: Питер, 2003, стр. 20.

**Интернет-ресурсы:**

ЯЗЫК Ассемблера за 3 Минуты <https://youtu.be/_-H271NKYgM>

[**https://ru.wikibooks.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80\_%D0%B2\_Linux\_%D0%B4%D0%BB%D1%8F\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82%D0%BE%D0%B2\_C**](https://ru.wikibooks.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80_%D0%B2_Linux_%D0%B4%D0%BB%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82%D0%BE%D0%B2_C)

**ЗАДАНИЕ**: законспектировать лекцию с учетом контрольных вопросов.

**КОНСПЕКТ ЛЕКЦИИ**

План

1. Машинный язык и язык ассемблера

2. Программная модель ЦП IA-32

3. Режимы работы процессора архитектуры IA-32

4. Набор регистров

**1. Машинный язык и язык ассемблера**

Как вы уже знаете, компьютер понимает только один язык — язык машинных команд. Чтобы не постигать значение различных комбинаций двоичных чисел, еще в 50-е гг. программисты стали использовать для программирования символический аналог машинного языка, который назвали языком ассемблера. Этот язык точно отражает все особенности машинного языка. Именно поэтому, в отличие от языков высокого уровня, язык ассемблера для каждого типа компьютеров свой. Более того, бессмысленны разговоры о том, что ассемблер как язык программирования устарел и знать его необязательно. АССЕМБЛЕР ОБЪЕКТИВЕН, и подобные разговоры в определенной ситуации могут выглядеть довольно глупо (особенно для особо «продвинутых» программистов).

Самую эффективную программу можно написать только на ассемблере (при условии, что ее пишет квалифицированный программист), так как этот язык является «родным» для компьютера. Здесь есть одно маленькое «но»: это очень трудоемкий и требующий большого внимания и практического опыта процесс. Поэтому реально на ассемблере пишут в основном программы, которые должны обеспечить эффективную работу с аппаратной частью компьютера. Иногда на ассемблере пишутся критичные ко времени выполнения или расходованию памяти фрагменты программы. Впоследствии они оформляются в виде подпрограмм и совмещаются с кодом на языке высокого уровня.

Как уже было сказано, язык ассемблера является символическим представлением машинного языка, он неразрывно связан с архитектурой самого процессора. По мере внесения изменений в архитектуру процессора совершенствуется и сам язык ассемблера.

В настоящее время доминирующее положение на рынке центральных процессоров для ЭВМ занимает семейство Intel-совместимых процессоров. Под термином «Intel-совместимые процессоры» подразумеваются процессоры фирм Intel, AMD, VIA, Transmeta, полностью поддерживающие базовую систему команд процессоров Intel и полностью или частично поддерживающие различные расширения базовой системы команд процессоров Intel.

Рассмотрим язык ассемблера на примере процессора IA-32 (Intel Architecture 32-bit).

*Другие названия: x86, i386, i486, i586, i686. Для 64-битного варианта: Intel 64, AMD64, amd64, EM64T, IA-32e, x86\_64, x86-64, x64.*

Это семейство открывают 16-разрядные процессоры 8086 и 8088 с 16/8-битной шиной данных и 20-битной шиной адреса. Второе поколение процессоров представлено процессором Intel 80286, в котором шина адреса была расширена до 24 бит, что позволяло в особом режиме (protected mode) адресовать до 16Мбайт физической памяти. Начиная с третьего поколения (Intel386) архитектура процессоров этого семейства стала 32-битной, а основным режимом работы - защищенный. В новых моделях совершенствовался кэш (Intel486), появились параллельные конвейеры (Pentium), появились новые архитектурные блоки (Pentium MMX (Multimedia Extensions — мультимедийные расширения)), появился встроенный кэш второго уровня (P6). Эти изменения сопровождались также добавлением новых возможностей при работе в защищенном режиме: VME (Virtual Mode Extension) у Pentium, PAE (Physical Address Extension) у P6 и др.

В семействе IA-32 декларируется программная совместимость моделей процессоров сверху вниз. Это значит, что код, написанный для 8086 должен работать и на Intel386, и на Pentium 4. С другой стороны, программы, разработанные для более поздних процессоров, могут не работать на более ранних, если в них используются какие-либо специфические особенности новой модели.

Рассмотрим базовую программную модель, общую для всех существующих на данный момент 32-разрядных процессоров: 80386, 486, Pentium, PentiumPro, Pentium II/III/IV и Celeron. Эта модель охватывает набор регистров процессора, организацию памяти и ввода/вывода, типы данных, систему команд, прерывания и исключения.

**2. Программная модель IA-32**

Программную модель процессора в архитектуре IA-32 представляет следующий набор ресурсов:

* пространство адресуемой памяти до 232-1 байт (4 Гбайт), для Pentium III/IV -до 236-1 байт (64 Гбайт);
* набор регистров для хранения данных общего назначения;
* набор сегментных регистров;
* набор регистров состояния и управления;
* набор регистров устройства вычислений с плавающей точкой (сопроцессора);
* набор регистров целочисленного MMX-расширения, отображенных на регистры сопроцессора (впервые появились в архитектуре процессора Pentium MMX);
* набор регистров MMX-расширения с плавающей точкой (впервые появились в архитектуре процессора Pentium III);
* программный стек — специальная информационная структура, работа с которой предусмотрена на уровне машинных команд.

Это основной набор ресурсов. Кроме того, к ресурсам, поддерживаемым архитектурой IA-32, необходимо отнести порты ввода-вывода, счетчики мониторинга производительности.

В программную модель данных процессоров входят 8- и 16-разрядные регистры общего назначения, сегментные регистры, регистры FLAGS, IP. Свойства некоторых программно-доступных ресурсов определяются текущим режимом работы процессора.

**3. Режимы работы процессора архитектуры IA-32**

В рамках архитектуры IA-32 доступные следующие режимы работы процессора.

***Режим реальных адресов****,* или просто *реальный режим*, в котором работал і8086.

***Защищенный режим*** позволяет максимально реализовать все идеи, заложенные в процессорах архитектуры IA-32, начиная с i80286. Программы, разработанные для реального режима, не могут функционировать в защищенном режиме.

***Режим виртуального процессора 8086*** предназначен для организации многозадачной работы программ, разработанных для реального режима (процессора і8086), совместимо с программами защищенного режима. Переход в этот режим (virtual 8086 mode) возможен, если процессор уже находится в защищенном режиме.

***Режим системного управления***— это новый режим, обеспечивает ОС механизмом для выполнения машинно-зависимых функций, таких как перевод компьютера в режим сниженного энергопотребления или выполнения действий по защите системы.

Процессор всегда начинает работу в реальном режиме.

**4. Набор регистров**

***Регистрами*** называются области высокоскоростной памяти, расположенные внутри процессора в непосредственной близости от его исполнительного ядра. Доступ к ним осуществляется несравнимо быстрее, чем к ячейкам ОП. Соответственно, машинные команды с операндами в регистрах выполняются максимально быстро, поэтому в программах на языке ассемблера регистры используются очень интенсивно.

Большинство регистров имеют определенное функциональное назначение. С точки зрения программиста, их можно разделить на две большие группы.

**Первую группу образуют регистры**, **предназначенные для пользователя**:

* регистры общего назначения (РОН) для хранения данных и адресов;
* сегментные рег. CS, DS, SS, ES, FS, GS - для хранения адресов сегментов в ОП;
* регистры сопроцессора ST(0), … ST(7) - для написания программ, использующих тип данных с плавающей точкой;
* целочисленные регистры MMX-расширения ММХ0, …, ММХ7;
* регистры MMX-расширения с плавающей точкой XMM0, XMM1, …,ХММ7;
* регистры состояния и управления (регистр флагов EFLAGS/FLAGS и регистр-указатель команды EIP/IP) содержат информацию о состоянии процессора, выполняемой программы и позволяют изменить это состояние

**Во вторую группу входят системные регистры,** то есть регистры, предназначенные для поддержки разных режимов работы, сервисных функций, а также регистры, специфические для определенной модели процессора.

Перечислим системные регистры, поддерживаемые IA-32:

* управляющие регистры CRO...CR4 определяют режим работы процессора и характеристики текущей выполняемой задачи;
* регистры управления памятью GDTR, IDTR, LDTR и TR используются в защищенном режиме работы процессора для локализации управляющих структур этого режима;
* наладочные регистры DR0.. .DR7 предназначенные для мониторинга и управления разными аспектами отладки;
* регистры типов областей памяти MTRR используются для аппаратного управления кэшированием в целях назначения соответствующих свойств областям памяти;
* машинно-зависимые регистры MSR используются для управления процессором контроля за его производительностью, получение информации об ошибках.

В обозначениях многих РОН присутствующая наклонная разделительная черта - это части одного большого 32-разрядного регистра, но их можно использовать в программе как отдельные объекты.

**Регистры общего назначения**

РОНы используются в программах для хранения :

* операндов логических и арифметических операций;
* компонентов адреса;
* указателей на ячейки памяти.

Эти регистры доступные для хранения операндов без особенных ограничений, хотя

в определенных условиях некоторые из них имеют жесткое функциональное назначение

Перечислим регистры, которые относятся к группе РОН и физически находятся в процессоре внутри арифметико-логического устройства (*регистры АЛУ):*

* ***регистр-аккумулятор*** EAX/AX/AH/AL – для хранения промежуточных данных, в некоторых командах его использования обязательное;
* ***базовый регистр*** *EBX*/BX/BH/BL - для хранения базового адреса объекта;
* ***регистр-счетчик***ECX/CX/CH/CL - в командах, выполняющих повторяющиеся действия;
* ***регистр данных*** EDX/DX/DH/DL-хранит промежуточные данные.
* ***регистр индекса источника*** ESI/SI и ***регистр индекса приемника*** EDI/DI - текущий адрес элемента в источнике и приемнике цепочки последовательных операций.

Для работы со стеком существуют специальные регистры:

* ***регистр указателя стека ESP/SP-*** на вершину стека в текущем сегменте стека;
* ***регистр указателя базы кадра стека EBP/BP – для организации*** произвольного доступа к данным внутри стека.

**Сегментные регистры**

Процессоры Intel аппаратно поддерживают *сегментную* организацию программы. Это означает, что любая программа состоит из трех сегментов: кода, данных и стека. При выборке каждой команды для доступа к данным программы или к стеку неявно используется информация из вполне определенных сегментных регистров. В программной модели IA-32 есть шесть ***сегментных регистров*** CS, SS, DS, ES, GS, FS, служащих для доступа к четырем типам сегментов.

***Сегмент кода*** содержит команды программы. Для доступа к этому сегменту служит *регистр сегмента кода* CS. Он содержит адрес сегмента с машинными командами, к которому имеет доступ.

***Сегмент данных*** содержит обрабатываемые программой данные. Для доступа к этому сегменту служит регистр сегмента данных DS, который хранит адрес сегмента данных текущей программы.

***Сегмент стека*** SS является областью памяти, называемой стеком. Работу со стеком процессор организует по следующему принципу: последний записанный в эту область элемент выбирается первым. Для доступу к этой области служит регистр сегмента стека SS, который содержит адрес сегмента стека.

***Дополнительный сегмент данных****.* Неявно алгоритмы выполнения большинства машинных команд допускают, что обрабатываемые ими данные расположены в сегменте данных, адрес которого находится в регистре сегмента данных DS. Если программе не хватает| одного сегмента данных, то она имеет возможность задействовать еще три дополнительных сегмента данных. Но в отличие от основного сегмента данных, адрес которого|какого| содержится в регистре DS, при использовании дополнительных сегментов данных их адреса требуется указывать явно с помощью специальных префиксов переопределения сегментов в команде. Адреса дополнительных сегментов данных должны содержаться в регистрах дополнительного сегмента данных ES, GS, FS.

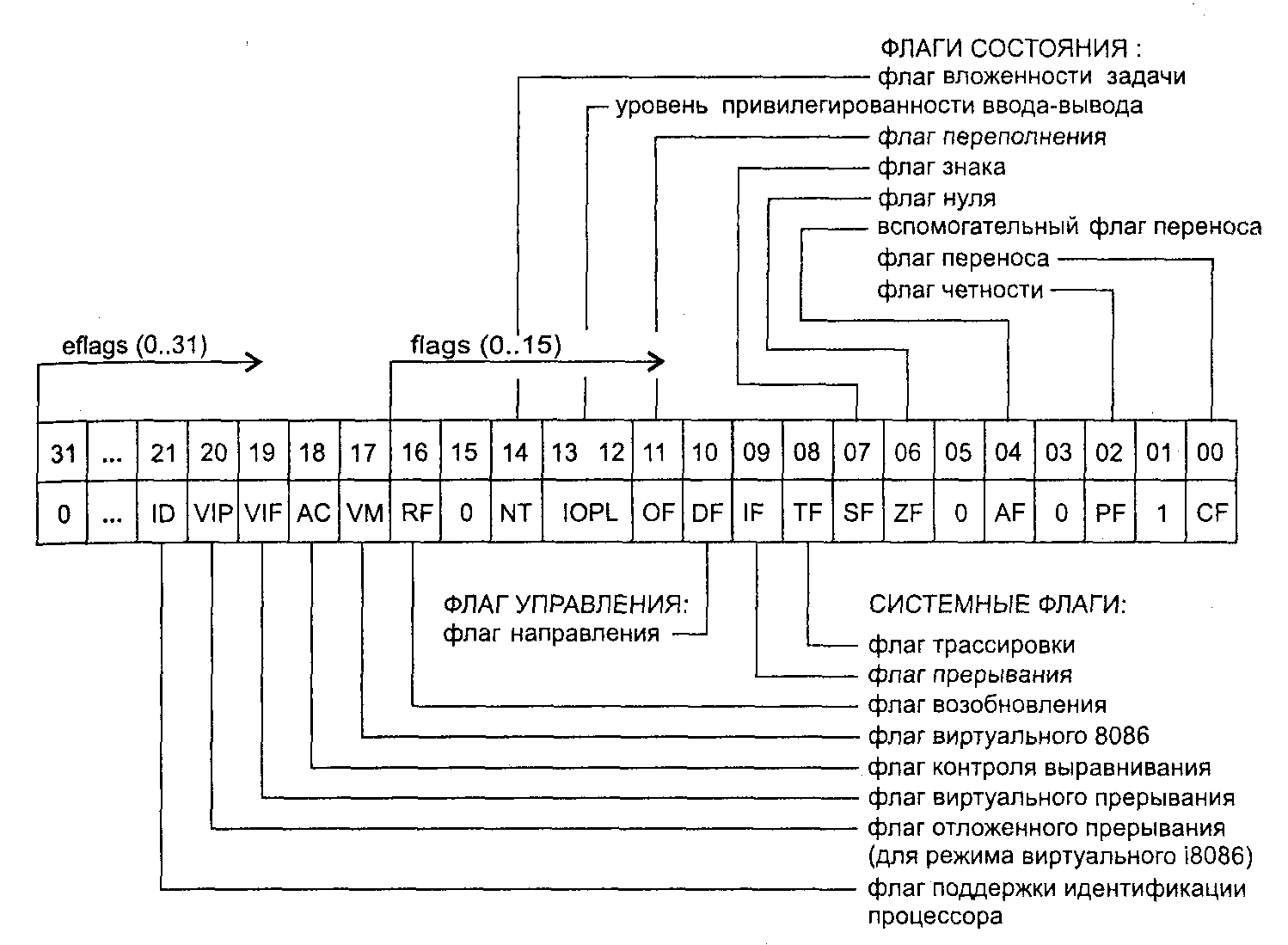
**Регистры состояния и управления**

В процессор включены два регистра, которые постоянно содержат информацию о состоянии как самого процессора, так и программы, команды которой он в данный момент обрабатывает: регистр-указатель команд EIP/IP и регистр флагов EFLAGS/FLAGS.

С помощью этих регистров можно также управлять состоянием процессора.

***Регистр-указатель команд***  EIP/IP имеет разрядность 32(16) биты и содержит смещение следующей подлежащей выполнению команды относительно содержимого регистра сегмента кода CS в текущем сегменте команд. Этот регистр непосредственно недоступный программисту, но загрузка и изменение его значения проводятся разными командами управления, к которым относятся команды условных и безусловных переходов, вызова процедур и возвращения из процедур. Возникновение прерываний также приводит к модификации регистра EIP/IP.

Разрядность ***регистра флагов*** (flag register) EFLAGS/FLAGS равная 32(16) битам. Отдельные биты данного регистра имеют определенное функциональное назначение и называются ***флагами.*** Младшая часть регистра EFLAGS/FLAGS полностью аналогичная регистру FLAGS процессора і8086. На рис.1 показанное содержимое имое регистра ЕFLAGS.



Исходя из особенностей использования, флаги регистра EFLAGS/FLAGS можно разделить на три группы.

***В первую группу входят 8 флагов состояния****.* Эти флаги могут изменяться после выполнения машинных команд. Флаги состояния регистра EFLAGS отображают особенности результата выполнения арифметических или логических операций. Это дает возможность анализировать состояние вычислительного процесса и реагировать на него с помощью команд условных переходов и вызовов подпрограмм.

***Флаг переноса*** CF (Carry Flag):

1 — арифметическая операция выполнила перенос из старшего бита результата;

0 — переноса не было.

***Флаг четности*** PF (Parity Flag): 1(0)—8 младших битов результата содержат четное (нечетное) число единиц.

***Вспомогательный флаг*** переноса AF (Auxiliary Carry Flag) фиксирует факт переноса или займа при выполнении операций сложения и вычитания:

1 — в результате операции сложения был перенос в старший разряд или при вычитании была ссуда в младший разряд;

0 — переноса и ссуды из третьего разряда младшей тетрады результата не было.

***Флаг нуля*** ZF (Zero Flag): 1 — результат нулевой; 0 — ненулевой.

***Флаг знака*** SF (Sign Flag) отображает состояние старшего бита результата:

1 — старший бит результата равный 1; 0 —0.

***Флаг переполнения*** OF (Overflow Flag) используется для фиксации факта потери значимого бита при арифметических операциях: 1 — в результате операции происходит перенос в старший знаковый бит результата или заем из старшего знакового бита результата ; 0 — в результате операции не происходит перенос в старший знаковый бит результата или заём из старшего знакового бита результата.

***Уровень привилегированного ввода-вывода*** IOPL используется в защищенном режиме работы процессора для контроля доступа к командам ***ввода-вывода*** в зависимости от привилегированной задачи.

***Флаг вложенности задачи*** NT используется в защищенном режиме работы процессора для фиксации того факта, что одна задача вложена в другую.

**Во вторую группу флагов (управления)** регистра ЕFLAGS/FLAGS входит всего один флаг ***направления***  DF. Он находится в десятом бите регистра ЕFLAGS и используется цепными командами. Значение флага DF определяет направление поэлементной обработки в этих операциях: от начала строки до конца или напротив, от конца строки к ее началу (DF =1). Для работы с флагом DF существуют специальные команды CLD (снять флаг DF) и STD (установить флаг DF). Приложение этих команд позволяет привести флаг DF в соответствие с алгоритмом и обеспечить автоматическое увеличение или уменьшение счетчиков при выполнении операций со строками.

В **третью группу флагов регистра EFLAGS/FLAGS входит 8 *системных флагов****,* ***которые*** управляют ***вводом-выводом***, маскируемыми прерываниями, отладкою, переключением между задачами и режимом виртуального процессора 8086. Прикладным программам не рекомендуется модифицировать без необходимости эти флаги, поскольку в большинстве случаев это ведет к прерыванию работы программы.

**Системные флаги и их назначения**.

***Флаг трассировки***  TF (Trap Flag) предназначен для организации пошаговой работы процессора:

1— процессор генерирует прерывание с номером 1 после выполнения каждой машинной команды; 0 — обычная работа.

***Флаг прерывания***  IF (Interrupt Flag) предназначен для разрешения или запрещения аппаратных прерываний: 1 — аппаратные прерывания разрешены; 0 —запрещены.

***Флаг возобновления*** RF (Resume Flag) - при обработке прерываний от регистров отладки.

***Флаг режима виртуального процессора 8086***  VM является признаком работы процессора в режиме виртуального 8086: 1 — процессор работает в режиме виртуального процессора 8086; 0 — в защищенном режиме.

***Флаг контроля выравнивания***  АС (Alignment check Flag) - для разрешения контроля выравнивание при обращениях к памяти. Используется совместно с битом AM в системном регистре CRO.

***Флаг виртуального прерывания*** VIF, что появился в процессоре Pentium, при определенных условиях (одно из которых — работа процессора в v-режиме) является аналогом флага IF.

***Флаг отложенного виртуального прерывания***  VIP (Virtual interrupt pending Flag), что появился в процессоре Pentium, устанавливается в 1 для индикации отложенного прерывания. Используется при работе в v-режиме совместно с флагом VIF.

***Флаг идентификации***  ID (IDentification Flag) используется для того, чтобы показать факт поддержки процессором инструкции CPUID. Если программа может установить или сбросить этот флаг, это значит, что данная модель процессора поддерживает инструкцию CPUID.

**Контрольные вопросы:**

1. Что собой представляет язык ассемблер?
2. Назовите три основных режима работы процессоров семейства IA – 32.
3. Перечислите все восемь 32-разрядных регистров общего значения.
4. Перечислите все шесть сегментных регистров.
5. Для какой цели используется регистр ECX?
6. Назовите 8 флагов состояния ЦП?