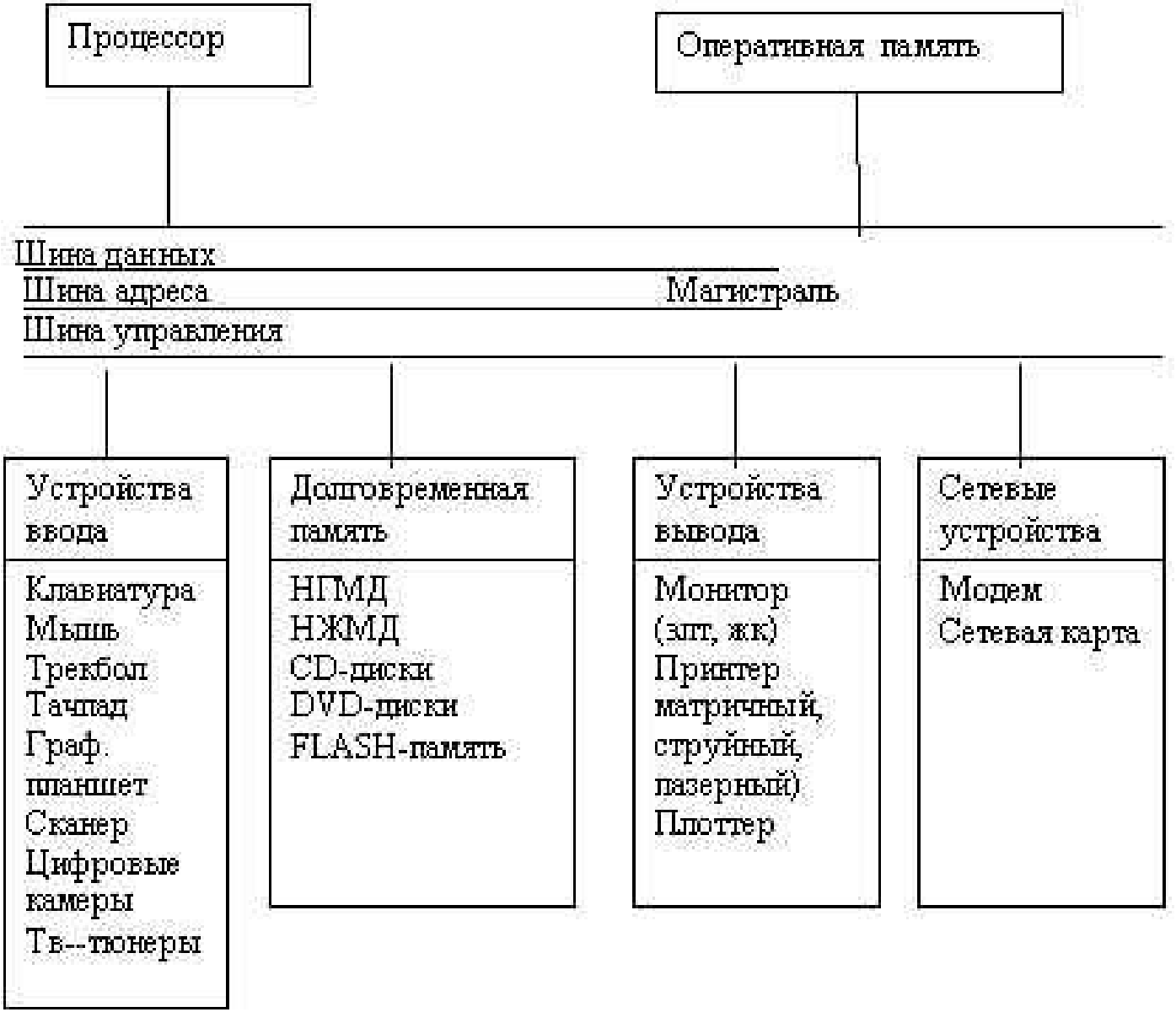
Лекция 19
Взаимодействие с внешними устройствами

Структурная схема



Регистры внешних устройств

- Адресуемые «ячейки»
- Чтение — получение информации о состоянии внешних устройств
- Запись — изменение состояния внешнего устройства
- Режимы доступа определяются для каждого бита индивидуально

Порты ввода-вывода x86

- Отдельное адресное пространство: 2^16 доступных портов ввода-вывода
- Первые 256 портов доступны «прямой адресацией», остальные через BX
- IN PORT, REG — чтение из порта. REG: AL, AX, EAX. PORT: IMM8, BX.
- OUT REG, PORT — запись в порт.

Работа с клавиатурой

- Порт 0x60 — данные
- Порт 0x64 — статус
  - Бит 0 — клавиатура не готова принимать данные
  - Бит 1 — данные от клавиатуры готовы
- Отправка байта на клавиатуру:

```
Sendbyte:
        mov     %eax, %ecx
wait: in      $0x64, %al
         test    $1, %al
        jnz     wait
        mov     %ecx, %eax
        out     %al, $0x60
        ret
```

Работа с клавиатурой

- Включение CapsLock, ScrollLock, NumLock LED

```
mov $0xed, %al
call sendbyte
mov $0x7, %al  ; light all leds
call sendbyte
```

- Чтение скан-кода

```
w: in $0x64, %al
   test $2, %al
   jz w
   in $0x60, %al
```

Проблемы

- Неудобный доступ
- 64K — слишком мало для современных компьютеров
  - Много устройств
  - Много управляющих регистров у каждого устройства
- Сложности с автоматическим конфигурированием и «горячим подключением»

Отображение в память

- Память устройства, конфигурационные параметры, регистры ввода-вывода отображаются на адресное пространство памяти
- PCI на 32-битных PC-совместимых компьютерах:
  - Верхний гигабайт физической памяти использовался для отображения устройств ввода-вывода
  - Память не доступна, даже если установлена

Проблемы (2)

- Постоянный опрос состояния внешнего устройства (i/o polling)
  - Устройство работают МНОГО медленнее процессора — очень много циклов тратится впустую — BUSY WAIT (активное ожидание)
  - Энергопотребление!
  - Процессор не может выполнять другие задачи

Типы взаимодействия

- Синхронное – программа ожидает выполнения операции ввода-вывода, не выполняет других действий
  - Постоянный опрос готовности (busy wait)
  - Ожидание средствами аппаратуры/ОС
- Асинхронное – программа выполняет другие действия одновременно с операцией ввода-вывода
  - Периодический опрос готовности (polling)
  - Уведомление по готовности (сигналы, прерывания)

Прерывания

- На одном из входов процессора выставляется электрический уровень прерывания
- Процессор заканчивает выполнение текущей инструкции
- Запрещаются прерывания (флаг IF процессора)
- Сохраняется минимальная информация, необходимая для продолжения выполнения с текущей точки
- Выполняется переход на обработчик прерывания

Сохраняемая информация

- В реальном режиме: EFLAGS, CS, EIP
- Сохраняется в стек (SS, ESP)
- В конце обработчика прерывания состояние процессора восстанавливается с помощью инструкции IRET
- Обработчик прерывания начинает работу в режиме запрета прерываний. Как можно быстрее прерывания должны быть разрешены инструкцией STI

Источники прерываний

- Non-maskable interrupt (NMI)
- Обычные от внешних устройств (INTR)
- Исключения — специальные состояния при исполнении программы
  - Page fault
  - Division by zero
  - …
- Программные исключения — инициируются самой выполняющейся программой с помощью инструкции INT n

Номер прерывания

- X86 поддерживает 256 различных типов прерываний (для всех целей)
- 256 адресов начала обработчиков прерываний — таблица обработчиков прерываний
  - В реальном режиме располагается по адресу 0
  - В защищенном — где угодно
- Программное прерывание:
  - INT 3 — debug trap (занимает 1 байт вместо 2)
  - INTO — INT 4 if OF is set (1 байт вместо 2)
  - INT n

Список прерываний

- 0-31 — исключения процессора
  - 0 — divide error
  - 3 — breakpoint
  - 4 — overflow
  - 6 — invalid opcode
  - 13 — general protection fault
  - 14 — page fault
- 32-255 - пользовательские

APIC (advanced programmable interrupt controller)

- Отображает 16 (32, 64) входных линий прерываний от внешних устройств в линию INTR на процессор (или на ядра процессора)
- Отображает входные номера прерываний IRQ0 … IRQ15 на прерывания процессора (например, INT32 .. INT47)
- Маршрутизация прерываний на ядра процессора
- Приоретизация прерываний
- Блокировка прерываний на время обработки

Финализация прерываний

- Когда APIC отправляет сигнал прерывания INTR на ЦП, отправка дальнейших сигналов блокируется
- ЦП должен явно разблокировать отсылку прерываний

```
mov $0x20, %al
out   %al, $0x20
```

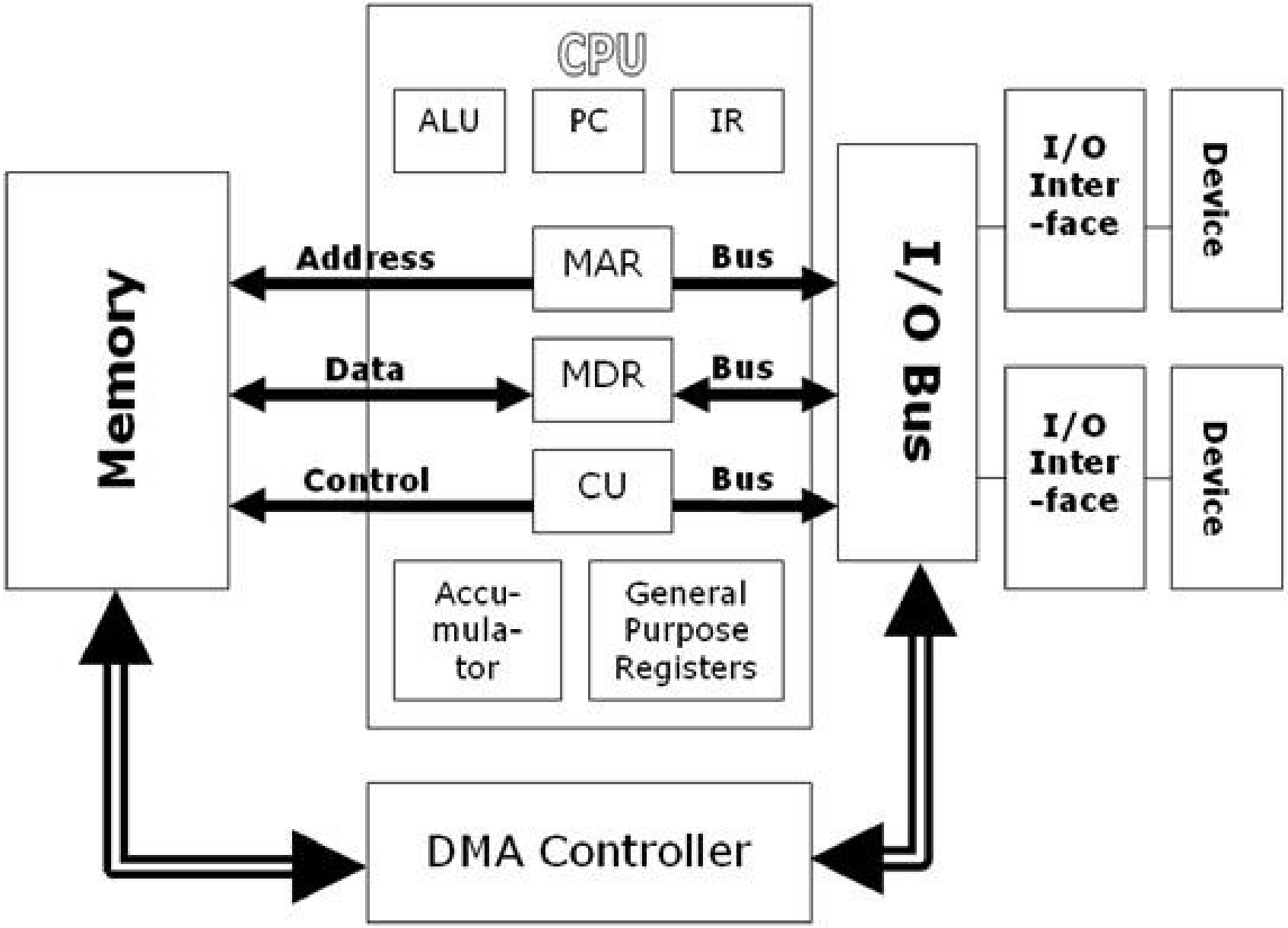
Прерывания от внешних устройств

```
           CPU0
 0:        134    XT-PIC-XT-PIC    timer
 1:        297    XT-PIC-XT-PIC    i8042
 2:          0    XT-PIC-XT-PIC    cascade
 8:          0    XT-PIC-XT-PIC    rtc0
 9:       5495    XT-PIC-XT-PIC    acpi, vboxguest, p7p1
10:       2088    XT-PIC-XT-PIC    p2p1
11:      22494    XT-PIC-XT-PIC    ahci, ohci_hcd:usb1
12:       1580    XT-PIC-XT-PIC    i8042
14:          0    XT-PIC-XT-PIC    ata_piix
15:       4186    XT-PIC-XT-PIC    ata_piix
```

DMA (Direct Memory Access)

- «Быстрые» устройства могут использовать DMA для пересылки данных в память в обход ЦП.
  - SATA-контроллер копирует считанный блок непосредственно в ОЗУ.
  - Ethernet-контроллер забирает пакет для передачи в сеть непосредственно из ОЗУ
- ЦП освобождается от пересылки данных
- Не «загрязняется» кеш процессора

DMA


CPU
ALU
PC
IR
Memory
Address
MAR
Bus
Data
MDR
Bus
Control
CU
Bus
I/O Bus
I/O Inter -face
Device
I/O Inter -face
Device
Accu- mula- tor
General Purpose Registers
DMA Controller

Алгоритм чтения сектора SATA

- ЦП выдает команду «считать сектор S по адресу в памяти M и выдать прерывание по готовности»
- SATA-контроллер считывает сектор S во внутренний буфер
- SATA-контроллер запрашивает DMA и копирует содержимое по адресу в памяти M
- После окончания копирования шлет запрос на прерывание на ЦП

Мультипрограммирование

- Организация выполнения нескольких программ одновременно (с перекрытием по времени) на одном компьютере
  - Распределение ОЗУ между работающими программами
  - Распределение времени ЦП (ядер ЦП) между программами
  - Управление внешними устройствами и организация доступа к ним

Требования к аппаратуре

- Поддержка прерываний
  - Для обработки событий от внешних устройств независимо от выполняющихся процессов
- Таймер
  - Для «справедливого» распределения времени между программами, защиты от “зависаний” процесса
- Защита памяти
  - Программа может работать только со своей памятью, доступ в чужую должен быть закрыт
- Режим супервизора
  - Чтобы предотвратить несанкционированное использование пп. 1-3

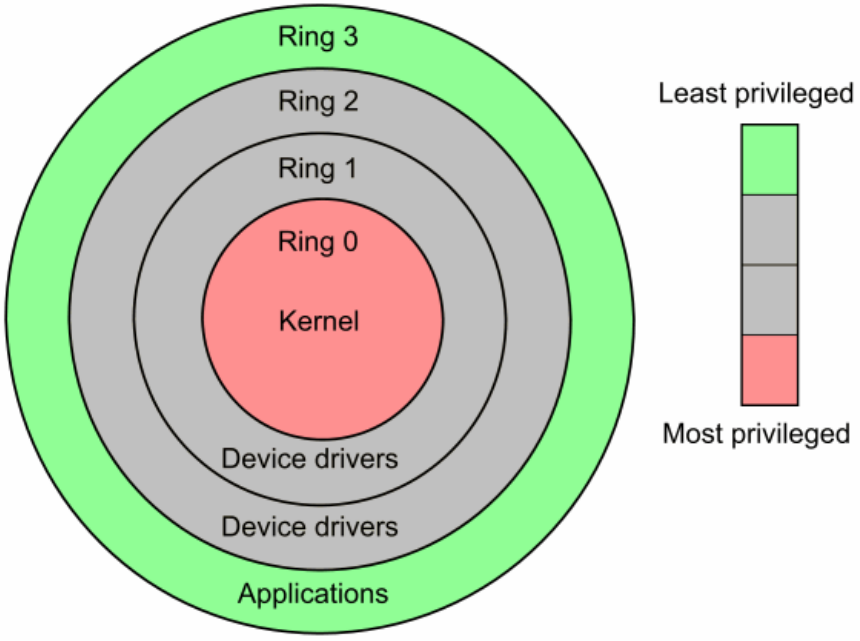
Защищенный режим x86

- При включении процессор работает в реальном режиме (real mode)
- Может быть переключен в защищенный режим (protected mode)
  - Регистр CR0 содержит бит включения (0)

```
mov %cr0, %eax
or    $1, %eax
mov %eax, %cr0
```

Кольца защиты

- 0 — самое защищенное (kernel mode)
- 3 — наименее защищенное
- Уровень защиты текущей программы — биты IOPL регистра EFLAGS



Пользовательский режим

- Попытка выполнения привилегированной инструкции — исключение GPF (General Protection Fault) (#13)
- Доступ к портам ввода-вывода может быть разрешен с помощью карты доступа в TSS (Task State Segment)
- Векторы обработки прерываний располагаются с адреса в регистре IDTR (Interrupt Descriptor Table Register)
- При переключении уровня привилегий из TSS загружаются регистры SS, ESP и для стека используются они
- Все регистры процессора сохраняются в TSS

Операционная система

- Операционная система – комплекс взаимосвязанных программ, управляющих аппаратурой компьютера и предоставляющих услуги для пользовательских прогргграмм, в том числе интерфейс взаимодействия с человеком
- Состоит из:
  - Загрузчика
  - Ядра
  - Системных утилит

Ядро операционной системы

- Ключевой компонент операционной системы
- Управляет компьютером и распределяет ресурсы компьютера между работающими программами (процессами)
- Изолирует процессы друг от друга, виртуализирует ресурсы компьютера
- Предоставляет универсальный (не зависящий от аппаратуры) интерфейс процессам
- Находится в памяти от включения до выключения компьютера (все время работы ОС)
- Работает в привилегированном режиме

Компоненты ядра ОС

- Интерфейс системных вызовов
- Планировщик (управляет распределением времени между процессами)
- Управление памятью
- Файловая система
- Управление внешними устройствами
- Сетевая подсистема (реализация сетевых протоколов)

Драйвер

- Программа в составе ядра ОС
- Реализует общий интерфейс класса устройств
  - Драйвер файловой системы
  - Драйвер видеокарты
  - Драйвер сетевой карты
- Скрывает специфику конкретного устройства от вышележащих компонентов ядра
  - Драйвера ext4 и vfat
  - Видеокарты nvidia и intel
  - Разные сетевые карты

Linux Kernel structure