x86. Векторные инструкции. Прерывания. BIOS. Ядро

Очередная лекция по АКОС

Команды x86

Байты	**0**	**1**	**2**	**3**	**4**	**5**
nop	0x00					
halt	0x10					
rrmovl *rA*, *rB*	0x20	*rA rB*				
irmovl *V*, *r*B	0x30	0x8 *rB*	*V*			
call *Des*t	0x80	*Dest*				
pushl *rA*	0xA0	rA 0x8				
popl *rA*	0xB0	rA 0x8				

Команды кодируются переменным количеством байт

Регистры общего назначения

- Intel 8080/8085 - 8 бит (A,B,C,D)
- Intel 8086 - 16 бит (AX, BX, CX, DX)
- Intel 80386 - 32 бит (EAX, EBX, ECX, EDX)
- AMD Opteron - 64 бит
  (RAX, RBX, RCX, RDX)

Процессоры 386+

- Сборки для i386 работают на любом процессоре
- Принципиально новая инструкция в i486 - CMPXCHG
- Для i686:
  - CMOV - Conditional Move
  - SYSENTER/SYSCALL

gcc -march=ИМЯ_ПРОЦЕССОРА или native

https://gcc.gnu.org/onlinedocs/gcc/x86-Options.html#x86-Options

Дополнительные наборы команд

- Векторные инструкции (MMX, SSE, AVX)
- Расширения для виртуализации (VT-X или AMD-V)

cat /proc/cpuinfo

Вещественная арифметика

- Исторически - отдельный сопроцессор x87
- При выполнении команд сопроцессора, основной процессор вынужден простаивать
- Сопроцессор работает в стековом режиме

Legacy FPU v.s. SSE

- Набор команд x87 доступен в 32 битном режиме
- Соглашения о вызовах для 32-битного режима: использовать x87
- Соглашения о вызовах для 64-битного режима: использовать SSE
- Для 32-битного режима можно выбрать, как генерировать код:
  gcc -mfpmath=387 или **gcc -mfpmath=sse**

Векторные инструкции

- Оперируют регистрами XMM/YMM/ZMM
- Горизонтальные - превращают вектор в скаляр
- Вертикальные - превращают два вектора в вектор

*Тот случай, когда тип **float** может быть полезен*

Пример: **dpps**

```
dpps xmm0, xmm1, 0xF1
```

- Вычисляет скалярное произведение 16 пар вещественных чисел
- Последний аргумент - только константа
  - Старшие 4 бита (0xF) определяют, какие из чисел участвуют в умножении
  - Младшие 4 бита (0x1) определяют, куда записать скалярный результат

Наборы инструкций Intel

- MMX - только целочисленные
- SSE - вещественные (128 бит)
- AVX и AVX-2 - вещественные (256 бит)
- AVX-512 - вещественные (512 бит)

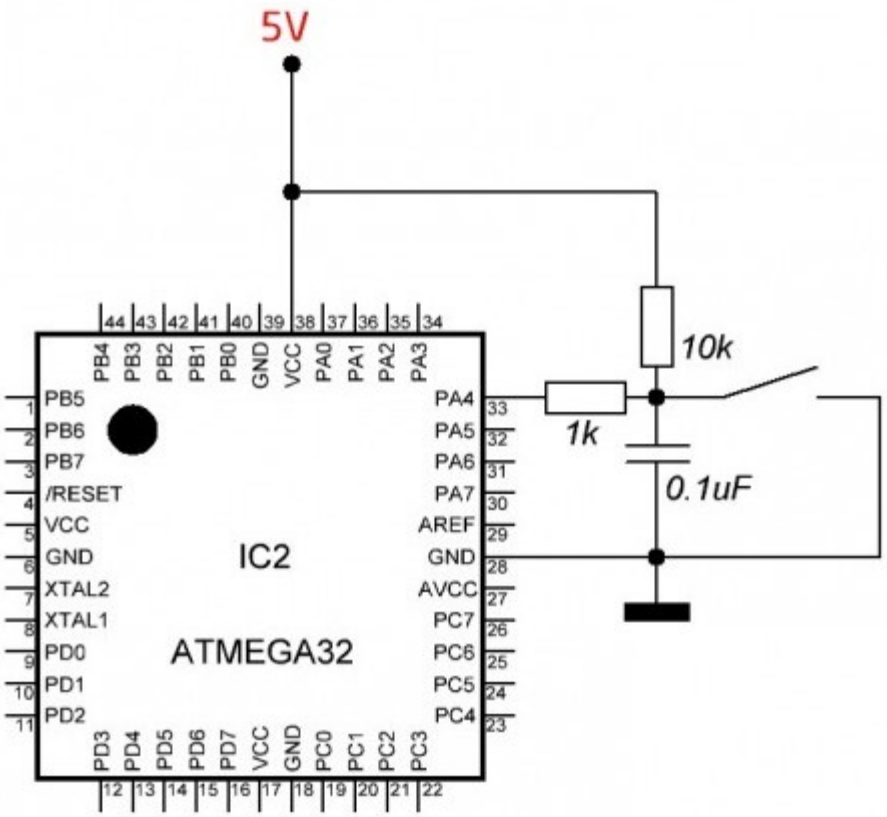
- Можно использовать Intrisicts в языке Си, вместо ассемблера:

https://software.intel.com/sites/landingpage/IntrinsicsGuide/

То, ради чего в курсе по операционным системам приходится изучать ассемблер

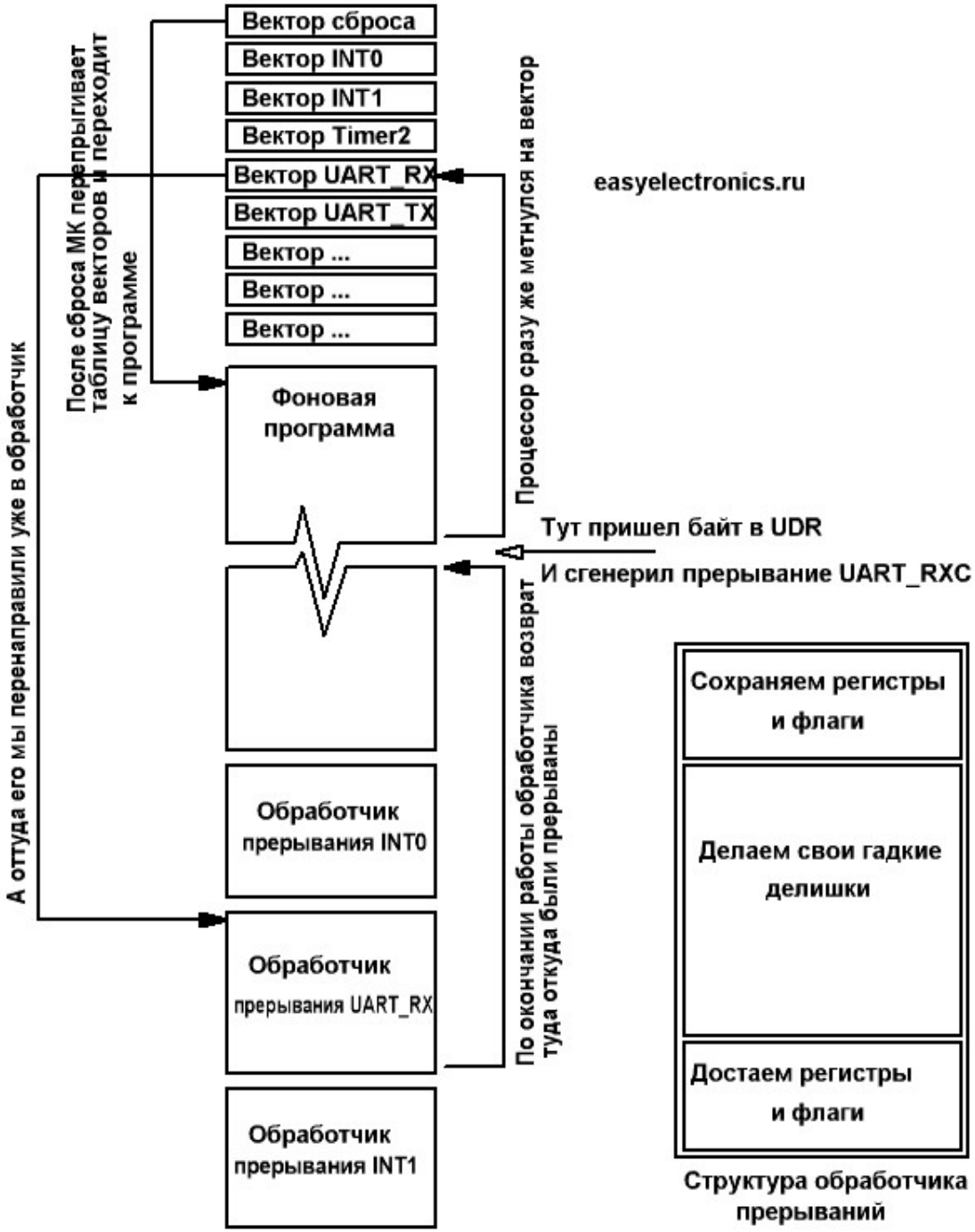
Прерывания

Прерывания в МК



Аппаратные события:

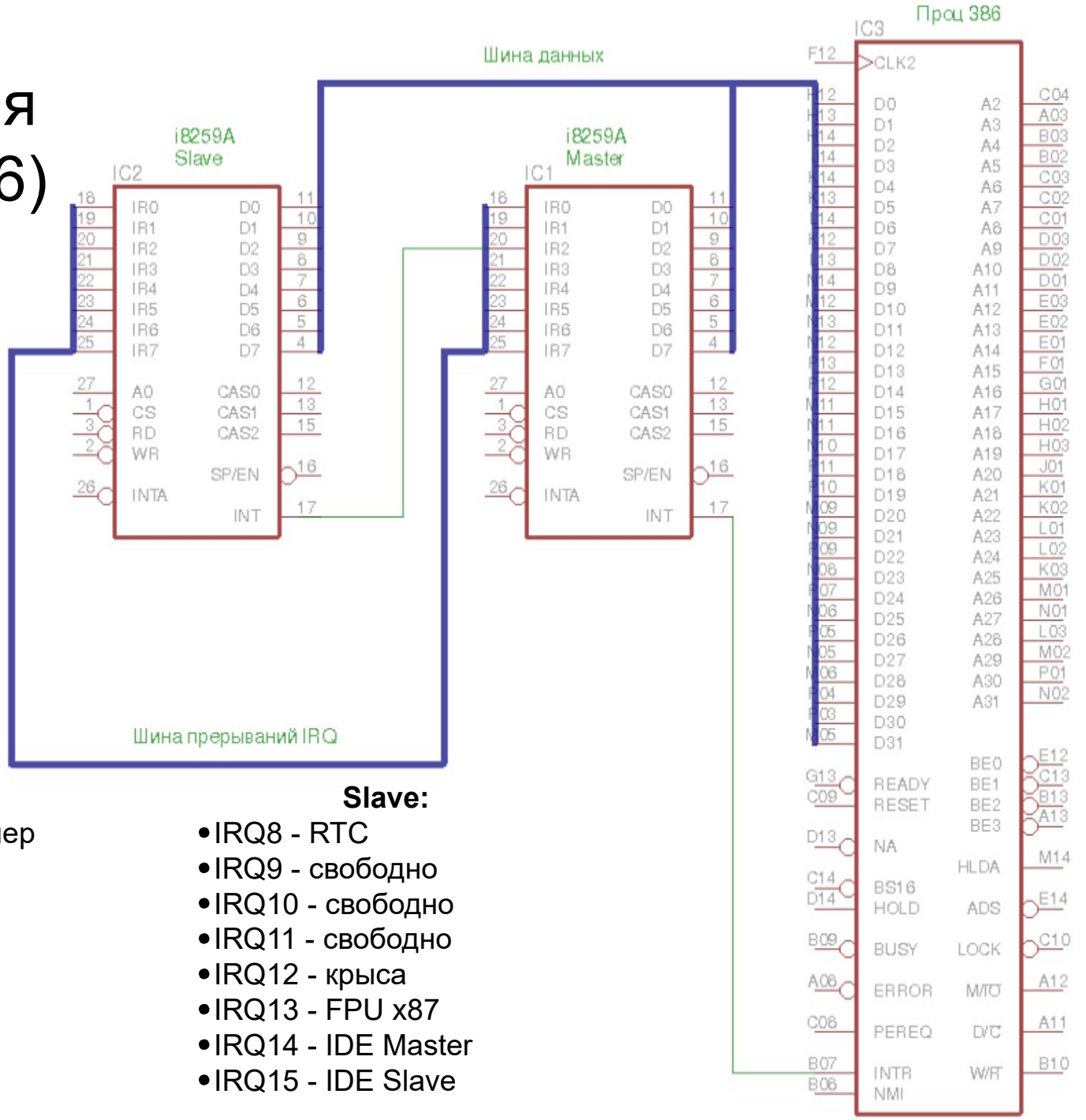
1. Нажатие кнопки
2. Приход очередного байта на шину данных от периферии
3. Сработал внутренний или внешний таймер
4. Прилетели инопланетяне



Что делать?

- Фиксированный вектор прерываний
- У каждого прерывания - свой адрес в векторе
- Операция либо NOP, либо JUMP на функцию-обработчик

Прерывания IBM PC (x86)



Master:

- IRQ0 - системный таймер
- IRQ1 - клава
- IRQ2 - slave
- IRQ3 - COM1
- IRQ4 - COM2
- IRQ5 - звуковуха
- IRQ6 - дисковод
- IRQ7 - LPT

Slave:

- IRQ8 - RTC
- IRQ9 - свободно
- IRQ10 - свободно
- IRQ11 - свободно
- IRQ12 - крыса
- IRQ13 - FPU x87
- IRQ14 - IDE Master
- IRQ15 - IDE Slave

Прерывания в 86/286/386/486

- Каждое устройство подает электрический сигнал
- Сигналы мультиплексируются с приоритетом
- Процессору отправляется только сигнал о факте прерывания
- Процессор опрашивает PIC о том, кто именно посмел его отвлечь от важных дел

Прерывания с PCI/PCIExpress

- Используется умная схема I/O APIC (Advanced Programmable Interrupt Controller)
- Устройства посылают сообщения о прерывании, которые выстраиваются в очередь
- Приоритетность обработки сообщений определяется программно, а не аппаратно

Немаскируемые прерывания

- NMI - отдельный сигнал процессору
- Имеет самый высокий приоритет

Что происходит с CPU

- Сохраняется IP в стеке
- Проставляется флаг IF
- Переход на инструкцию из регистра IDTR + смещение
  Регистр IDTR:
  - младшие 16 бит - размер таблицы
  - старшие биты - физический адрес

Что ещё происходит

До вызова обработчика:

- Сброс конвейеров и отклонение Out-of-order execution
- Переключается таблица отображения виртуальных адресов
- Меняется указатель на стек

Выполняется обработчиком:

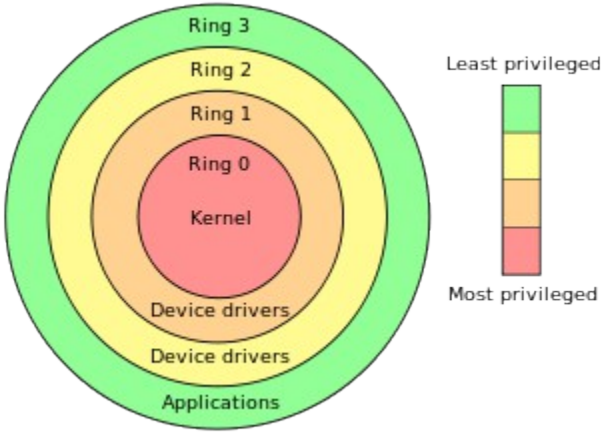
- Сохранение состояния в стек: регистры + флаги
- Восстановление состояния по завершению работы обработчика

Что делает обработчик

- Взаимодействует с железякой, которая вызвала прерывание.
- Для этого необходим доступ к портам ввода-вывода и физической памяти
- У обычных программ таких прав нет

Защищенный режим x86

- Каждой программе видна "своя" память
- Рядовые приложения не могут ничего сделать с оборудованием
- Два бита IOPL
  в регистре флагов



Программные прерывания

- Команда int NN
- С точки зрения обработки ничем не отличаются от аппаратных

- До загрузки ОС - функции BIOS
- ОС может (но не обязана) модифицировать таблицу прерываний

Некоторые прерывания

В AH хранится команда, в AL - аргумент

BIOS

- INT 0x10 - управление текстом на экране
- INT 0x13 - управление дисками
- INT 0x15 - управление UART
- INT 0x16 - опрос клавиатуры
- INT 0x16 - опрос клавиатуры

DOS

- INT 0x21 - взаимодействие с DOS API

Ядро ОС (Kernel)

- Программа, которая работает на (почти) самом привелигерованном уровне процессора
- Имеет доступ ко всему

Системный вызов

- Функция из API ядра операционной системы
- Выполняется в режиме ядра, то есть с высокими привилегиями на уровне процессора
- После заврешения выполнения процессор переходит обратно на непривелигированый уровень

INT 0x80

- Стандартный номер прерывания для инициирования системных вызовов в Linux
- Регистр EAX - номер системного вызова (см. /usr/include/asm/unistd_32.h)
- Параметры - в EBX, ECX, EDX, ESI, EDI, EBP
- Возвращаемое значение - в EAX

Системные вызовы Linux

- 2 раздел man
- В стандартной библиотеке - именованные оболочки со стандартным Си-соглашением о вызовах функций

Пример: вывод строки

```
static const char S[] = "Hello";
write(1, S, sizeof(S));
/* man 2 write
    #include <unistd.h>
    ssize_t write(int fd, const void *buf, size_t count);
*/
```

```
#include <asm/unistd_32.h>

        mov eax, __NR_write  // номер системного вызова
        mov ebx, 1           // первый аргумент
        mov ecx, S_ptr       // второй аргумент
        mov edx, 6           // третий аргумент
        int 0x80             // do it!

S:      .string "Hello"
S_ptr: .dword S
```

linux-vdso.so (linux-gate.so)

```
$ ldd /usr/bin/cat
        linux-vdso.so.1 (0x00007ffe3ea87000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f9cea333000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f9cea6d6000)
```

- Виртуальная "библиотека", отображаемая в адресное пространство пользовательского процесса
- Отображаются критичные по времени функции ядра, которые не обязательно выполнять с высокими привилегиями:
  - __vdso_clock_gettime
  - __vdso_getcpu
  - __vdso_gettimeofday
  - __vdso_time

Другие способы инициации системных вызовов

- sysenter/sysexit - для IA-32
- syscall и vsyscall - для AMD x86-64