**Прерывания от внешних устройств в системе x86.**

## I. Эволюция контроллеров прерываний

Рассмотрим механизмы доставки прерываний от внешних устройств процессору в системе x86 и попытаться ответить на вопросы:

1) что такое PIC и для чего он нужен?
2) что такое APIC и для чего он нужен? Для чего нужны LAPIC и I/O APIC?
3) что такое MSI? В чём отличия MSI и MSI-X?
4) как с этим связаны таблицы $PIR, MPtable, ACPI?

**Введение**

В систему прерываний входит три типа прерываний:

- Системные вызовы (System calls)
- Исключения или ловушки (Traps)
- Аппаратные прерывания (Interrupts)

При глубоком изучении можно сказать, что ОС управляется событиями (англ. OS is event driven), т.е. код операционной системы выполняется только при возникновении прерывания, системного вызова или исключения.

Системные вызовы и исключения являются синхронными событиями, так как возникают при выполнении программного кода.

Аппаратные прерывания или просто прерывания (часто их называют внешними) – это асинхронное событие, возникающее вне зависимости от каких-либо действий, выполняемых в системе.

*Рассмотрим аппаратные прерывания более подробно.*

### Аппаратные прерывания

Схема обработки аппаратных прерываний — это принципиально архитектурно зависимое действие, связанное с непосредственным взаимодействием с контроллером прерываний. Но схема в основных чертах остаётся неизменной, независимо от архитектуры. Это связано со спецификой решаемой прерываниями задачи информирования процессора о асинхронном событии, возникающем в системе.

Аппаратные прерываний возникают от внешних устройств, являются в системе асинхронными событиями, которые возникают независимо от какой-либо выполняемой в системе работы, и их принято делить на следующие группы:

- Прерывание от системного таймера, которое возникает в системе периодически.
- Прерывания от устройств ввода-вывода. Возникают по инициативе устройства, когда устройству нужно сообщить процессору о завершении операции ввода-вывода.
- Прерывания от действий оператора, например, в ОС Windows при нажатии клавишей ctrl_alt_del для вызова task manager.

Аппаратные прерывания освобождают процессор от необходимости опрашивать внешние устройства с целью определения их готовности передать запрошенные процессом данные. Но требуют от системы выполнения последовательности действий по их обслуживанию. Пока данные не готовы процессор может выполнять какую-то другую работу. Но, когда поступает сигнал прерывания, процессор должен переключиться на его обслуживание.

1. PIC

В персональных компьютерах на базе процессоров Intel обработка аппаратных прерываний поддерживается аппаратно. Первой была микросхема Intel 8259 PIC, которая имела 8 входных линий (IRQ0-7), и одну выходную линию INTR (или просто INT) (Рис.1).

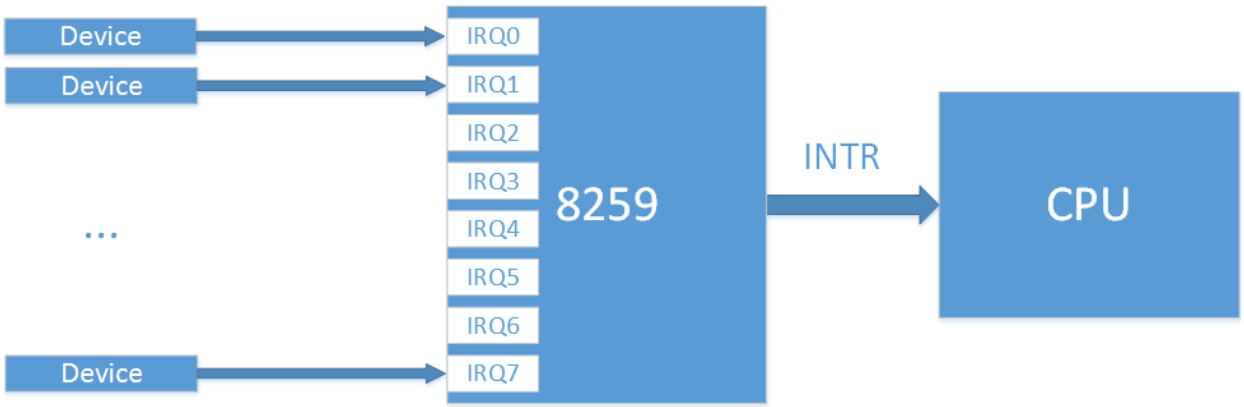


Рис.1

Сигналы прерывания от устройств ввода-вывода поступают на входы IRQ (Interrupt Request), а контроллер прерывания формирует сигнал прерывания, который по шине управления (линии INTR) поступает на соответствующую ножку (pin) процессора. Сигнал прерывания будет передан процессору, если он не замаскирован, т.е. его обработка разрешена. Для увеличения числа обрабатываемых прерываний контроллеры стали подключать в виде каскада: ведущий (master) и ведомый (slave) контроллеры (всего 15 линий IRQ, одна линия используется для каскадного соединения) (рис.2).

О возникновении прерывания CPU сигнализирует только ведущий (master). Если возникло прерывание на линиях 8-15, второй PIC (slave) сигнализирует о прерывании мастеру по линии IRQ 2, и тот уже в свою очередь сигнализирует CPU. Это каскадное прерывание отнимает одну из 16

линий, но в итоге даёт 15 доступных прерываний для устройств.

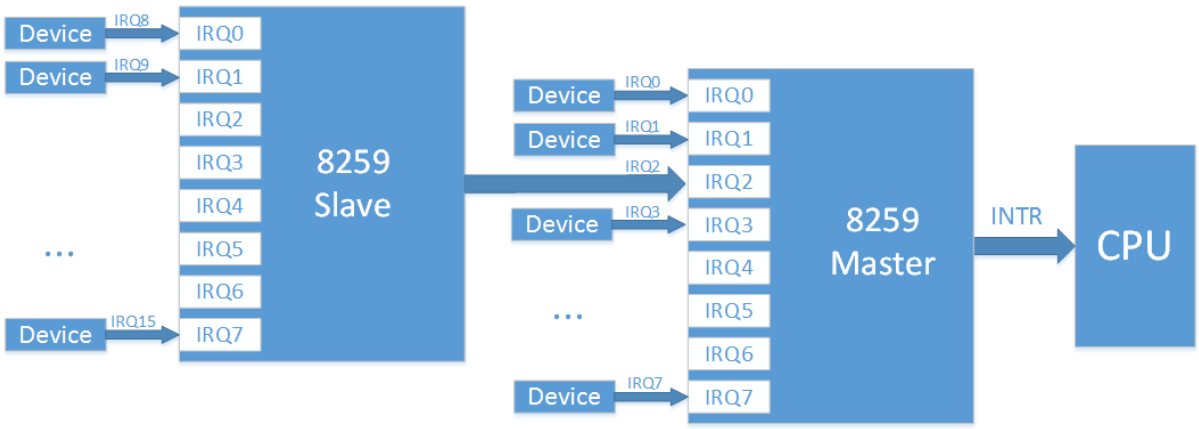


Рис.2

Этого было достаточно для систем с шиной ISA.

Обычно раскладка прерываний под устройства была более менее стандартная, например:

IRQ 0 — system timer
IRQ 1 — keyboard controller
IRQ 2 — cascade (прерывание от slave контроллера)
IRQ 3 — serial port COM2
IRQ 4 — serial port COM1
IRQ 5 — parallel port 2 and 3 or sound card
IRQ 6 — floppy controller
IRQ 7 — parallel port 1
IRQ 8 — RTC timer
IRQ 9 — ACPI
IRQ 10 — open/SCSI/NIC
IRQ 11 — open/SCSI/NIC
IRQ 12 — mouse controller
IRQ 13 — math co-processor
IRQ 14 — ATA channel 1
IRQ 15 — ATA channel 2

Конфигурация и работа с микросхемами 8259 осуществляется через I/O порты:

| Чип | Регистр | I/O port |
|---|---|---|

| | | |
|---|---|---|
| Master PIC | Command | 0x0020 |
| Master PIC | Data | 0x0021 |
| Slave PIC | Command | 0x00A0 |
| Slave PIC | Data | 0x00A1 |

На смену шине ISA пришла шина PCI. И количество устройств, требующих подключения, стало больше. Кроме того, в отличие от статической шины ISA шина PCI позволяла добавляться устройства в систему динамически. В данной шине прерывания могут быть разделяемыми (то есть к одной линии IRQ можно подсоединить несколько устройств). В итоге чтобы решить проблему нехватки линий IRQ, прерывания от PCI устройств стали группировать в линии PIRQ (Programmable Interrupt Request) (рис.3).

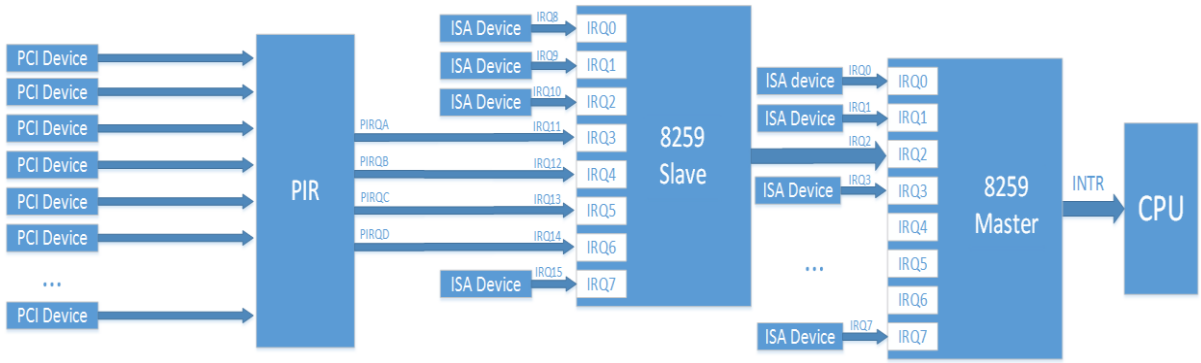


Рис.3

*Допустим у нас 4 линии прерываний свободно на PIC контроллере, а PCI устройств 20. Они объединяются по 5 устройств на линию PIRQx и подключаем линии PIRQx к контроллеру. При возникновении прерывания на линии PIRQx процессору будет опрашивать все устройства, подключённые к данной линии для того, чтобы определить от какого именно устройства пришло прерывание. Устройство осуществляющее связывание линий прерываний PCI в линии PIRQ часто называют PIR router.*

*Информацию о роутинге прерываний на PIC контроллере BIOS передавал ОС с помощью таблицы $PIR и с помощью заполнения регистров 3Ch (INT_LN Interrupt Line (R/W)) и 3Dh (INT_PN Interrupt Pin (RO)) конфигурационного пространства PCI для каждой функции. Спецификация о таблице $PIR раньше была на сайте Microsoft, но сейчас её там уже*

*нет. Содержимое строк таблицы $PIR можно понять из PCI BIOS Specification.*

2. APIC

С появлением многопроцессорных систем PIC был заменен APIC (Advanced PIC), чтобы сбалансировать нагрузку по обработке прерываний на процессоры. Для каждого процессора добавляется специальный контроллер LAPIC (Local APIC) и для маршрутизации прерываний от устройств добавляется контроллер I/O APIC. Все эти контроллеры объединяются в общую шину с названием APIC (новые системы сейчас уже соединяются по стандартной системной шине).

Другими словами, контроллер выполнен в виде двух раздельных устройств. Одна часть контроллера размещена прямо в ядре процессора (так называемый Local APIC, или LAPIC), другая (I/O APIC) — на материнской плате. Когда прерывание от устройства приходит на вывод I/O APIC, контроллер направляет прерывание в LAPIC одного из процессоров. Наличие I/O APIC позволяет сбалансировано распределять прерывания от внешних устройств между процессорами.

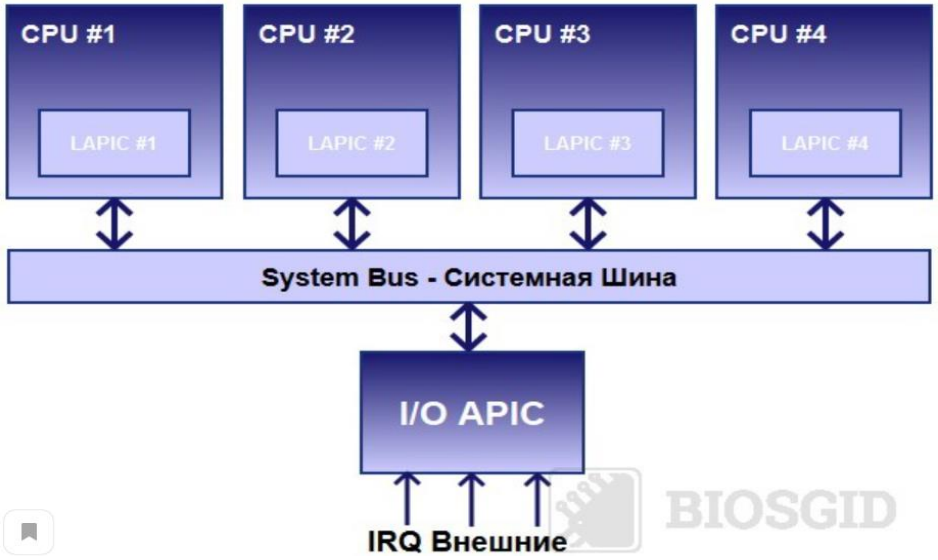


Рис.4

I/O APIC 82093AA содержала 24 входных вывода, а архитектура APIC могла поддерживать до 16 CPU. Для поддержки совместимости со старыми системами, прерывания 0~15 отвели под старые прерывания ISA. А прерывания от PCI устройств стали выводить на линии IRQ 16-23.

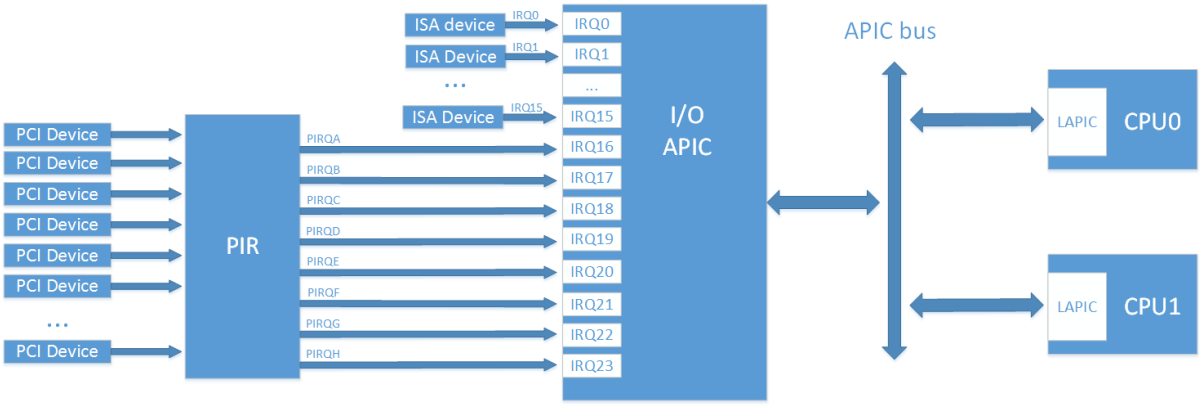


Рис.5

Программирование I/O APIC и LAPIC осуществляется через MMIO. Регистры LAPIC расположены обычно по адресу 0xFEE00000, регистры I/O APIC по адресу 0xFEC00000. Хотя в принципе все эти адреса возможно переконфигурировать.

*В дальнейшем архитектура APIC получила модернизацию и новый вариант получил название xAPIC (x — extended). Сохранена обратная совместимость с предыдущим вариантом. Количество возможных CPU в системе увеличилось до 256.*

*Следующий виток развития архитектуры получил название x2APIC. Количество возможных CPU в системе увеличилось до 2^32. Контроллеры могут работать в режиме совместимости с xAPIC, а могут в новом режиме x2APIC, где программирование LAPIC осуществляется не через MMIO, а через MSR регистры (что гораздо быстрее). Судя по этой ссылке для работы этого режима необходима поддержка IOMMU.*

*Следует заметить, что в системе может быть несколько контроллеров I/O APIC. Наприме, один на 24 прерывания в южном мосту, другой на 32 в северном. В контексте I/O APIC прерывания часто обозначаются GSI (Global System Interrupt). Есть ли в CPU встроенный LAPIC и какой именно архитектуры можно понять по бит-флагам в CPUID. Чтобы система могла обнаружить LAPIC и I/O APIC, BIOS должен представить информацию о них системе либо через таблицу MPtable (старый метод), либо через таблицу ACPI (таблицу MADT в данном случае). Помимо общей информации, и в MPtable и в ACPI (на этот раз в таблице DSDT) должна содержаться информация о роутинге прерываний, то есть информация о том, какое устройство сидит на какой линии прерываний (аналог таблицы $PIR).*

*О таблице MPTable можно почитать в официальной спецификации. Раньше спецификация была на сайте Intel, а сейчас её можно найти только в архиве. Спецификация ACPI сейчас расположена на сайте UEFI (текущая версия 6.2). Следует отметить, что с помощью ACPI можно указать роутинг прерываний и для систем без APIC (вместо использования таблицы $PIR).*

## 3. MSI

На смену шины PCI пришёл PCI express, в котором линии прерываний было решено убрать. Чтобы сохранить совместимость, сигналы о возникновении прерываний (INTx#) эмулируются отдельными видами сообщений. В этой схеме логическое сложение линий прерываний, которое раньше производилось физическим соединением проводов, легло на плечи PCI

мостов. Однако поддержка legacy INTx прерываний — это лишь поддержка обратной совместимости с шиной PCI. На деле PCI express предложил новый метод доставки сообщений о прерываниях — MSI (Message Signaled Interrupts). В этом методе для сигнализации о прерывании устройство просто производит запись в MMIO область отведённую под LAPIC процессора.


Рис. 6

Если раньше на одно PCI устройство (то есть на все его функции) выделялось всего 4 прерывания, то сейчас сейчас стало возможным адресовать до 32 прерываний.

В случае с MSI нет никакого разделения (sharing) для линий прерываний. Каждое прерывание соответствует своему устройству.

Прерывания MSI решают также ещё одну проблему. Допустим устройство проводит memory-write транзакцию, и хочет сообщить о её завершении через прерывание. Но write транзакция может быть задержана на шине в процессе передачи (о чём устройство никак не знает), и сигнал о прерывании придёт до процессора раньше. Таким образом CPU будет читать ещё невалидные данные. В случае если используется MSI, информация об MSI передаётся также как и данные, и раньше прийти просто не сможет.

Следует заметить, что прерывания MSI не могут работать без LAPIC, но использование MSI может заменить нам I/O APIC (упрощение дизайна).

В последствии данный метод получил расширение MSI-X. Теперь каждое устройство может иметь до 2048 прерываний. И стало возможным указывать индивидуально каждому прерыванию на каком процессоре оно должно выполняться. Это может быть очень полезно для высоконагруженных устройств, например сетевых карт.

Для поддержки MSI не требуется никаких дополнительных таблиц BIOS. Но устройство должно сообщить о поддержке MSI в одной из Capability в своём PCI Config, а драйвер устройства должен поддерживать работу с MSI.
Для устройств и драйверов, поддерживающих прерывания MSI/MSI-X, используются именно они. Остальные прерывания «роутятся» (англ. *routing* - маршрутизация) через I/O APIC.
Упрощённая схема роутинга прерываний показана на рис.7.

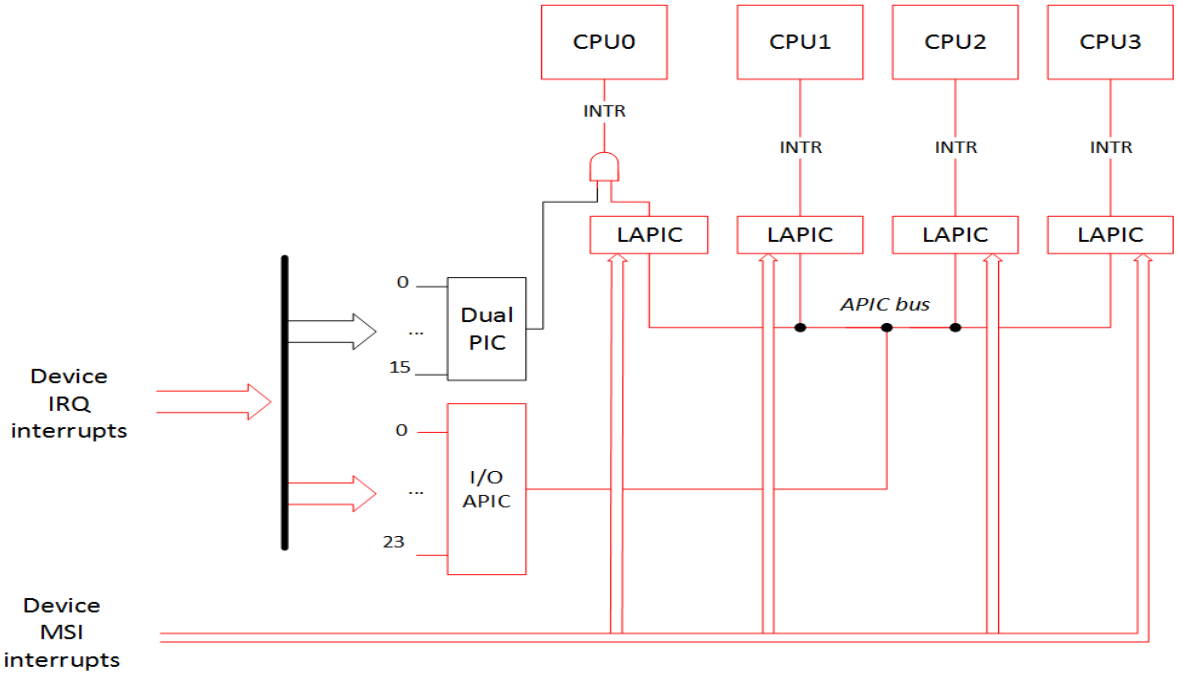


Рис.7

В многопроцессорных и многоядерных системах устройства получают возможность самостоятельно выбирать процессор/ядро для обработки конкретного прерывания, причем это делается полностью на уровне аппаратуры без программной поддержки. Это позволяет оптимизировать работу системы путем размещения большей части структур драйвера устройства и связанного с ним ПО, например, сетевых протоколов, в кэше конкретного процессора или же в его «ближней» NUMA-памяти[1]

**II. Адресация обработчиков прерываний в защищенном режиме**

Схема, представленная на рис.3, может рассматриваться как абстракция, позволяющая лаконично продемонстрировать адресацию обработчиков прерываний в защищенном режиме. Операции, выполняемые на единственном процессоре, рассматриваются только как тривиальный, вырожденный случай SMP (SMP - **S**ymmetric **M**ulti**P**rocessing — симметричное мультипроцессирование).

[1] NUMA – Non-Uniform Memory Access – «неравномерный доступ к памяти» или Non-Uniform Memory Architecture – архитектура с неравномерной памятью – схема реализации оперативной памяти, которая используется в мультипроцессорных системах, когда время доступа к памяти определяется ее расположением относительно процессора. Практически все архитектуры процессоров используют кэши. В NUMA поддержка когерентности кэша приводит к тому, что более один кэш может хранить содержимое одной и той же ячейки памяти. Системы с когерентностью кэша называются ccNUMA. Kernel Linux начиная с версии 2.5 содержит базовые NUMA. Версия 3.8 поддерживает новую NUMA и позволяет более эффективно развивать NUMA политику в следующих версиях.

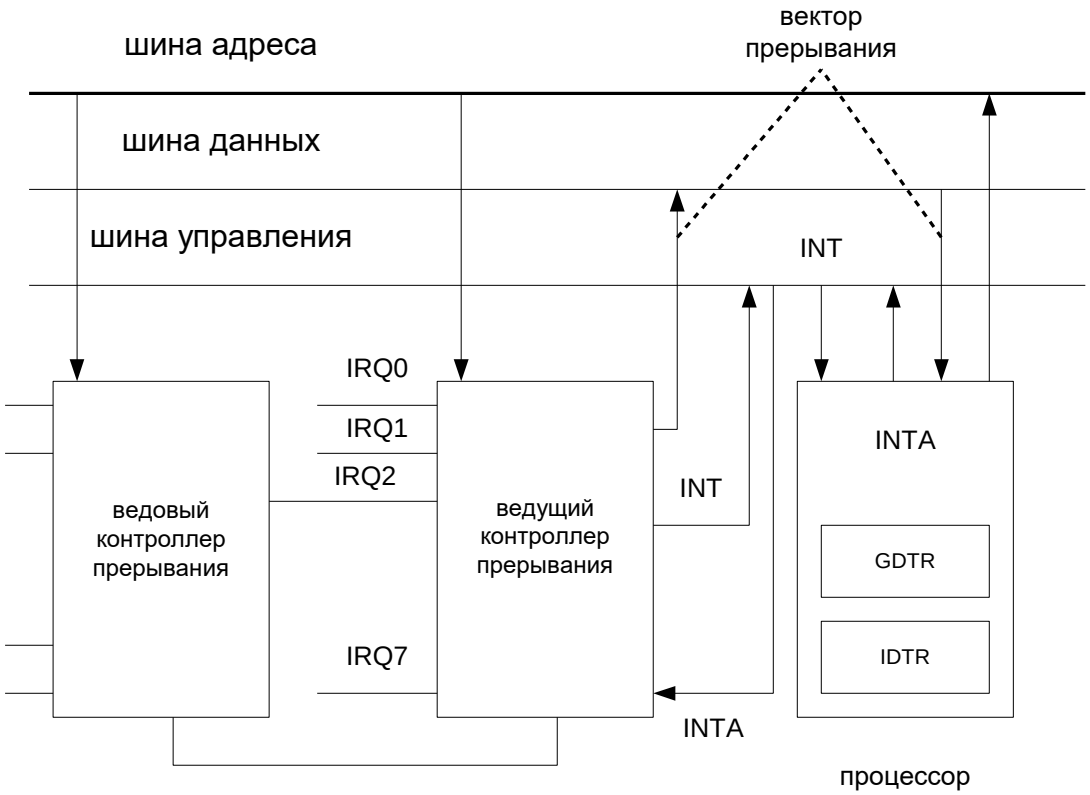


Рис.8

Регистр GDTR содержит начальный адрес таблицы GDT – Global Descriptor Table – таблицы глобальных дескрипторов, которая в защищенном режиме содержит дескрипторы сегментов физической памяти.

Регистр IDTR содержит начальный адрес таблицы дескрипторов прерываний, которая содержит дескрипторы шлюзов прерываний, ловушек и задач.

Дескриптор глобальной таблицы дескрипторов имеет размер 8 байт и имеет следующие поля (рис.4):

1. Размер – limit – 2 байта
2. Начальный адрес сегмента – младшая часть – base_low – 2 байта
3. Начальный адрес сегмента – средняя часть – base_midle – 1 байт
4. Байт атрибутов – младший байт атрибутов – attr1
5. Байт атрибутов – старший байт атрибутов – attr2
6. Начальный адрес сегмента –старшая часть – base_hight – 1 байт

Таким образом, начальный адрес сегмента составляет 32 разряда в защищенном режиме.

Биты атрибутов 1 имеют следующее назначение:

Бит 0 - A – бит доступа к сегменту; устанавливается аппаратно при обращении к сегменту;

Следующие три бита определяют тип сегмента:

Бит 1 – для сегмента кода, если бит сброшен, то чтение из сегмента запрещено, но это не относиться к выборке команд, если установлен, то чтение разрешено;
для сегмента данных, если бит сброшен, то модификация данных запрещена, если установлен, то модификация данных разрешена.
Бит 2 – для сегмента кода это – бит подчинения: если 0, то сегмент является подчиненным, если 1, то сегмент кода - обычный;

- для сегментов данных и стека: если 0, то это сегмент данных, если 1, то это - сегмент стека;

Бит 3 – бит предназначения: если он равен 0, то это – сегмент данных или стека, если равен 1, то это – сегмент данных.
Бит 4 - S - указывает, является ли дескриптор системным (0 - дескриптор системный).
Биты 5-6 - DPL – descriptor privilege level - привилегии дескриптора – определяет уровень привилегий (0 - ядро системы, 3 - пользовательский). В архитектуре x86 поддерживается 4 кольца защиты: от 0 до 3.
Бит 7 - P - present - признак присутствия сегмента в памяти (сегмент может быть выгружен менеджером виртуальной памяти).

Биты байта атрибутов 2 имеют следующие значения:

Биты 0-3 - limit – размер и с учетом двух первых байтов дескриптора – 0-го и 1-го – размер сегмента ограничен 20 битами ($2^{20}$=1024 Кб или 1Мб)
Бит 4 - AVL – "For software use, not used by hardware".
Бит 5 - 0 – зарезервирован, а в 64-разрядных системах это - флаг L, который теперь служит признаком 64-разрядности сегмента. Если он установлен, флаг D/B должен быть сброшен.
Бит 6 - D/B - флаг, указывающий разрядность сегмента: 0 – сегмент 16-разрядный, 1 – сегмент 32 разрядный (этот флаг ещё называют BIG).
Бит 7 - G - бит гранулярности (Granularity) - указывает, в чём измеряется лимит (0 - в байтах, 1 - в страницах по 4 килобайта).

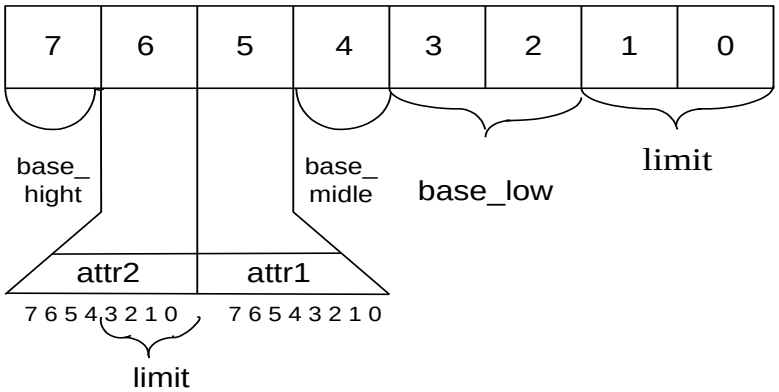


Рис.9 Дескриптор таблицы GDT

Дескриптор таблицы дескрипторов прерываний имеет следующий формат: размер дескриптора 8 байт, которые распределяются следующим образом (рис. 10):

- Первые два байта и последние два байта – смещение – 32 бита.
- Байты 2 и 3 – селектор – 16 бит – используется для доступа к дескриптору сегмента, содержащего обработчик прерывания, в таблице GDT (смещение). Получив из дескриптора сегмента его начальный адрес и добавив смещение из дескриптора прерывания, получается адрес точки входа в обработчик прерывания.

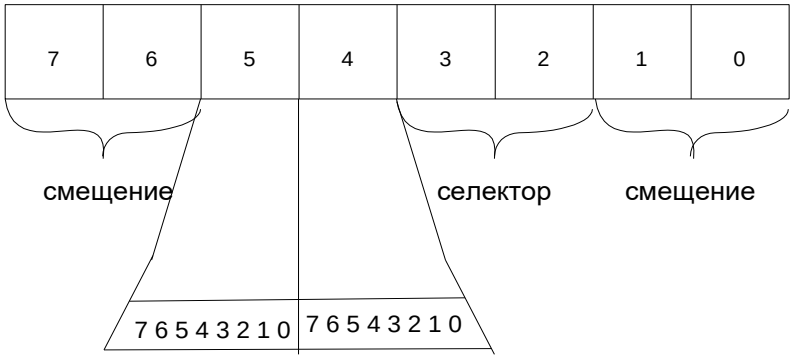


Рис. 10 Дескриптор таблицы IDT

- Байт 4 – резерв
- Байт 5 – байт атрибутов:

Биты 0, 1, 2, 3 определяют тип дескриптора

| № | Значение типа | Назначение |
|---|---|---|
| 1 | 0 | Не определено |
| 2 | 1 | Свободный сегмент задачи (TSS) 80286 |
| 3 | 2 | LDT |
| 4 | 3 | Занятый сегмент состояния задачи (TSS) 80286 |
| 5 | 4 | Шлюзы вызова 80286 |
| 6 | 5 | Шлюз задачи (Task Gate) |
| 7 | 6 | Шлюз прерываний 80286 (Interrupt Gate) |
| 8 | 7 | Шлюз ловушки 80286 (Trap Gate) |
| 9 | 8 | Не определено |
| 10 | 9 | Свободный сегмент состояния задачи (TSS) 80386, 80486 и старше |
| 11 | Ah | Не определено |

| | | |
|---|---|---|
| 12 | Bh | Занятый сегмент состояния задачи (TSS) 80386, 80486 и старше |
| 13 | Ch | Шлюз вызова 80386, 80486 и старше |
| 14 | Dh | Не определено |
| 15 | Eh-1110 | Шлюз прерываний (Interrupt Gate) 80386, 80486 и старше |
| 16 | Fh-1111 | Шлюз ловушки (Trap Gate) 80386, 80486 и старше |

4 бит – 0 – не используется
Биты 5-6 – DPL - Уровень привилегий сегмента (**DPL** англ. - **Descriptor** Privilege Level) соответствует значению поля **DPL** в дескрипторе сегмента. Текущий уровень привилегий (CPL англ. ... **DPL** вызываемого сегмента не должен быть численно больше CPL;
**DPL** шлюза не должен быть численно меньше **DPL,** указанного в нём сегмента;
В шлюзе должен быть указан только сегмент кода или TSS (если это шлюз задачи).
Бит 7 – P – present – бит присутствия, если 0 – неиспользуемый вектор.

Для доступа к дескриптору прерывания в качестве смещения в таблице IDT используется вектор прерывания (рис. 3).