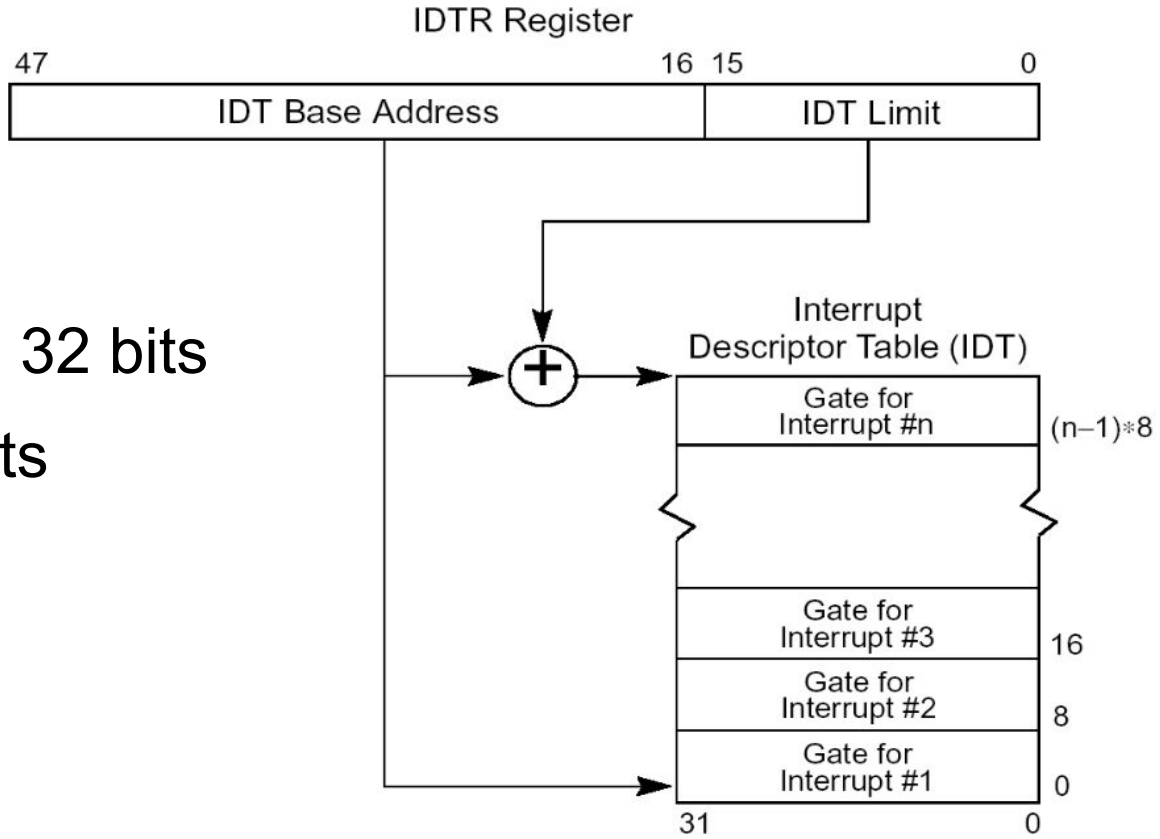
IDT

Cómo cargar la IDT

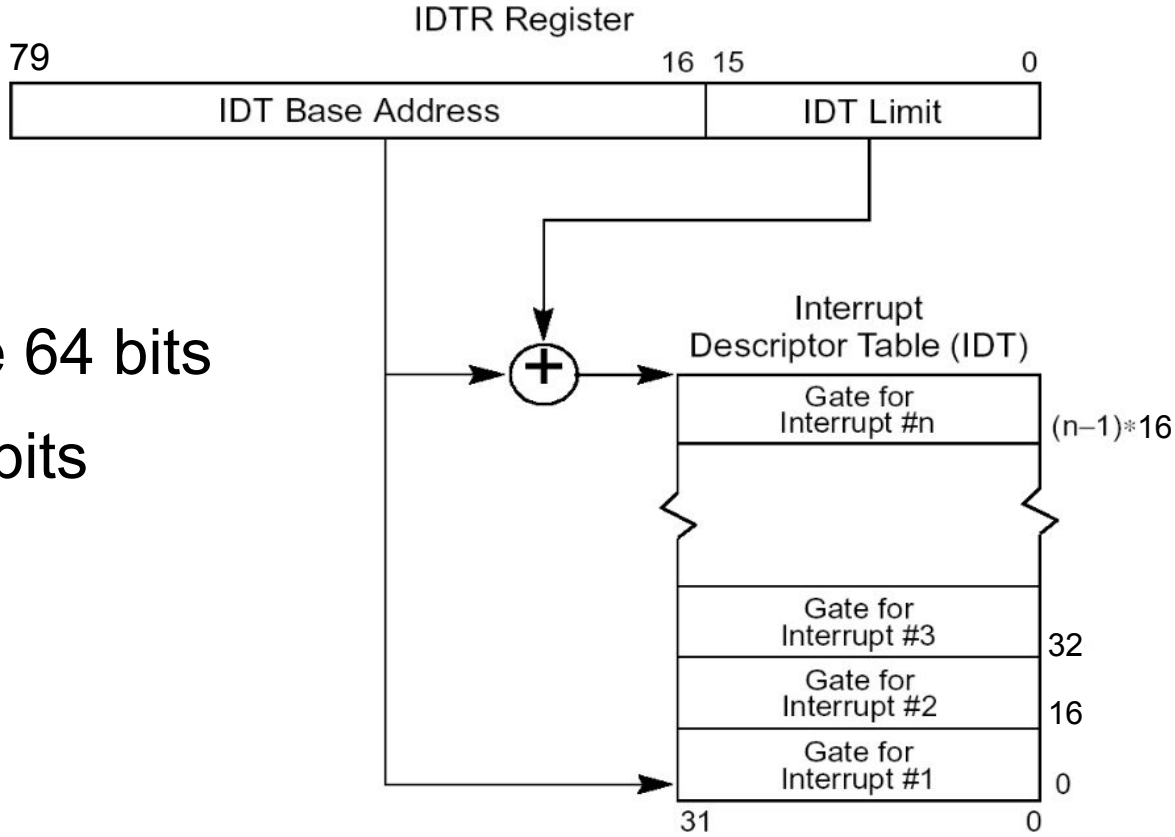
Interrupciones en 32 bits

- IDT Base Address de 32 bits
- Descriptores de 64 bits
- IDTR se carga con la instrucción LIDT

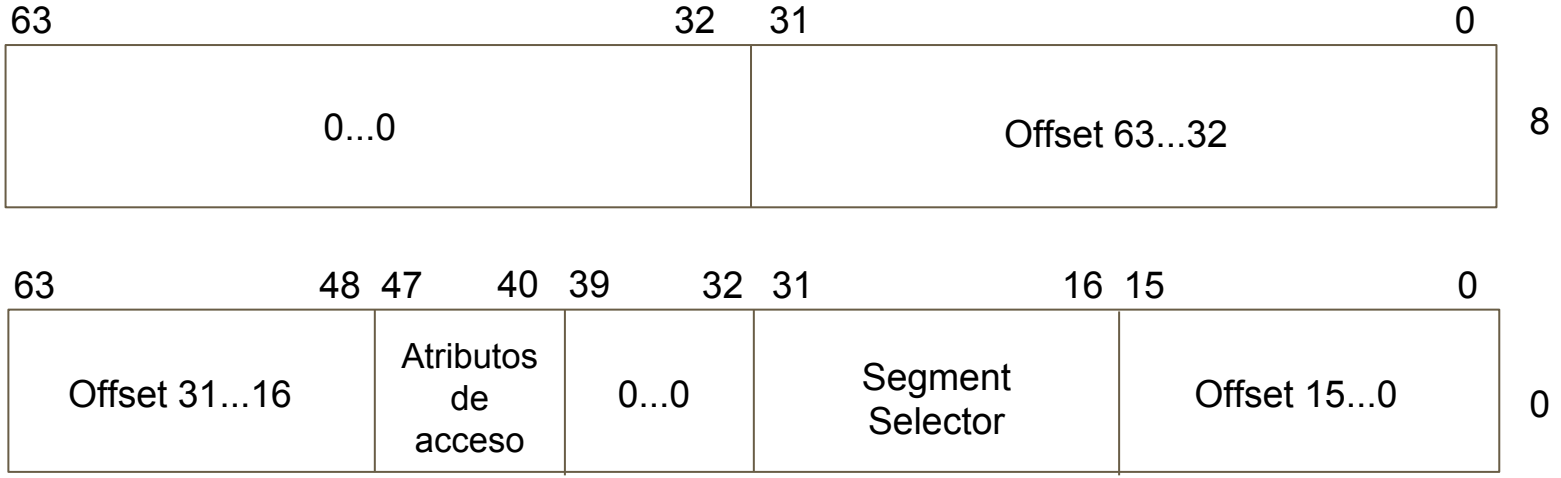


Interrupciones en 64 bits

- IDT Base Address de 64 bits
- Descriptores de 128 bits
- IDTR se carga con la instrucción LIDT



Descriptores en 64 bits



- Los offsets representan el puntero a la función a invocar cuando se produce la interrupción: la rutina de atención de interrupción
- Dicha función pertenece al segmento de código (CS) que fue cargado en la entrada 1 de la GDT por Pure 64. Por lo tanto, el Segment Selector vale 0x08 = 0000 0000 0000 1000 b

Cargar la IDT

1. Cargar el registro IDTR con la instrucción LIDT
2. Definir un *struct* con el formato de un descriptor
3. Definir un puntero a la IDT. El valor del puntero vendría a ser la IDT Base Address
4. Cargar los descriptores deseados en la IDT en la posición correspondiente. Por ejemplo, la interrupción 80h corresponde al descriptor en la posición 80h de la IDT
    a. Si la interrupción es de hardware se debe enviar al PIC el EOI (End of Interrupt) una vez manejada la interrupción. Para esto se envía 20h al 20h del mapa I/O. Es necesario también habilitar el IRQ correspondiente mediante la máscara del PIC.

Cargar el registro IDTR

De esto ya se encargó Pure 64. Los valores del IDT Base Address y IDT Limit se encuentran en *Bootloader/Pure64/src/sysvar.asm*. Estos son:

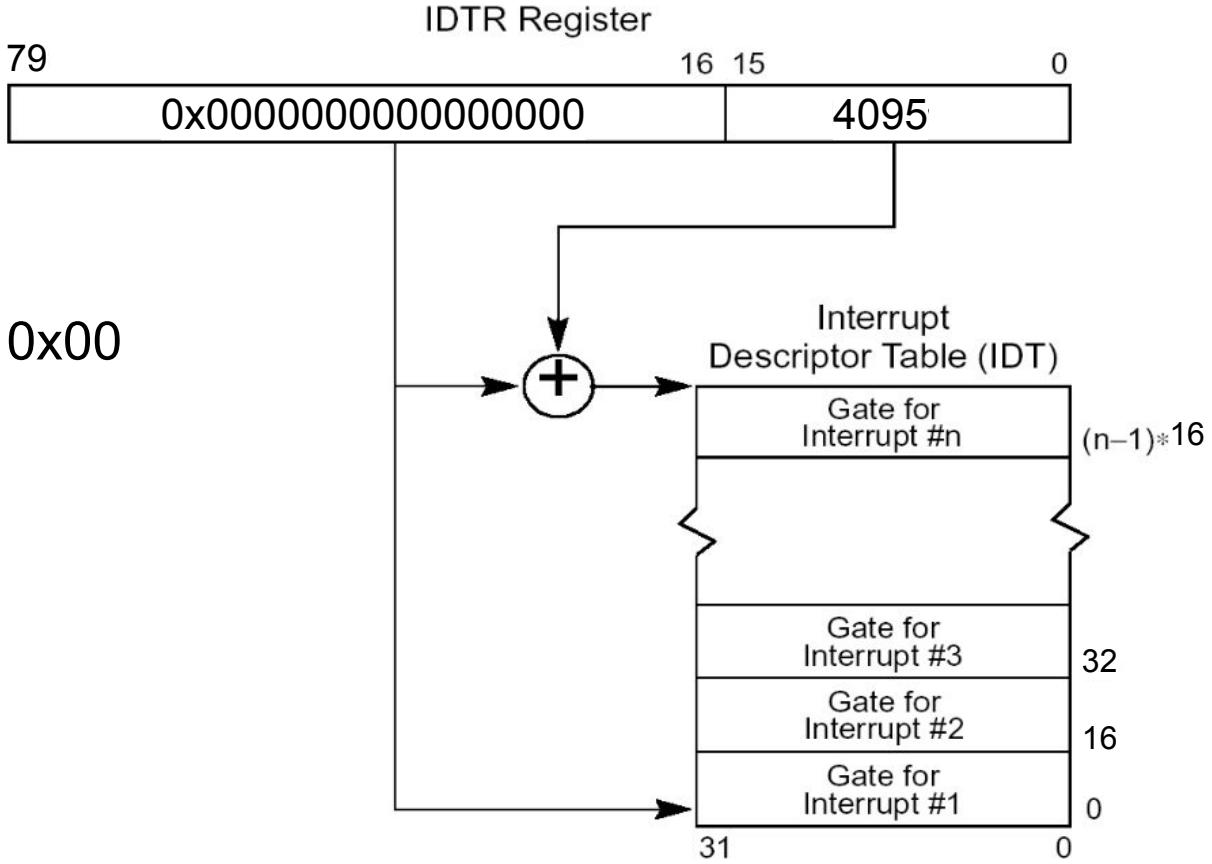
- IDT Base Address = 0x00
- IDT Limit = 4095

Por lo tanto:

- La IDT se encuentra en la posición 0x00 de memoria
- Ocupa hasta 4096 bytes, entonces puede tener hasta 256 descriptores, pues 4096 / 16 = 256

Cargar el registro IDTR

- IDT Base Address = 0x00
- IDT Limit = 4095
- n = 256 descriptores

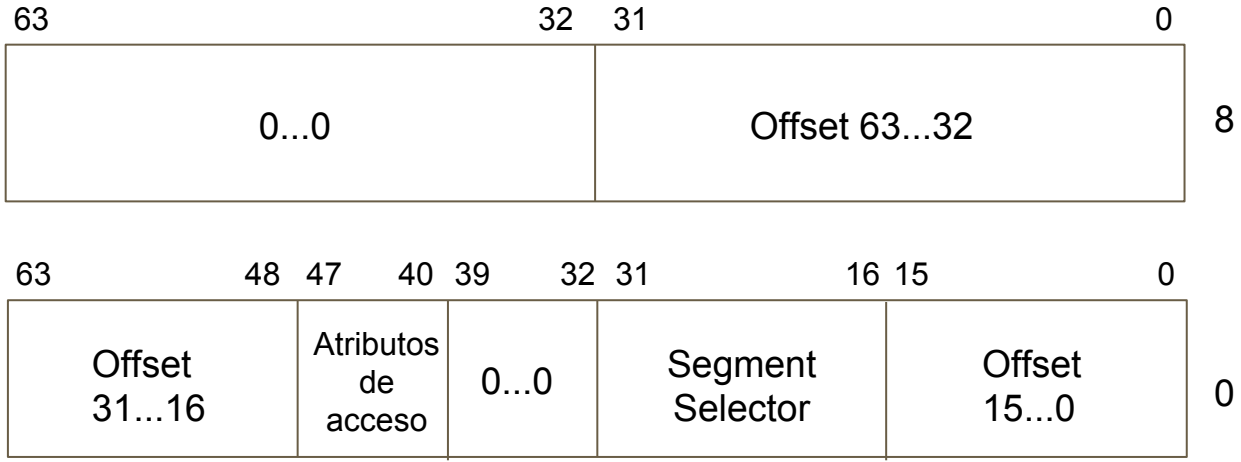


Definir un *struct* con el formato de un descriptor

```
#pragma pack(push)
#pragma pack (1)

typedef struct {
  uint16_t offset_l;
  uint16_t selector;
  uint8_t  cero;
  uint8_t  access;
  uint16_t offset_m;
  uint32_t offset_h;
  uint32_t other_cero;
} DESCR_INT;

#pragma pack(pop)
```



idtLoader.c

Definir un puntero a la IDT

La IDT estaba cargada en la posición 0x00

```
DESCR_INT * idt = (DESCR_INT *) 0;
```

idtLoader.c

Cargar descriptores

Definimos una función que recibe el número de la interrupción y el puntero a la rutina de atención de interrupción

```
void setup_IDT_entry (int index, uint64_t offset) {
  idt[index].offset_l = offset & 0xFFFF;
  idt[index].selector = 0x08;
  idt[index].cero = 0;
  idt[index].access = ACS_INT;
  idt[index].offset_m = (offset >> 16) & 0xFFFF;
  idt[index].offset_h = (offset >> 32) & 0xFFFFFFFF;
  idt[index].other_cero = 0;
}
```

idtLoader.c

Cargar descriptores

La interrupción IRQ0 del timer tick corresponde a la 0x20. La rutina de atención de interrupción es la función *_irq00Handler*. Se debe invocar a `load_idt` en el comienzo del main de Kernel/kernel.c

```
void load_idt() {
  _cli();

  setup_IDT_entry (0x20, (uint64_t) &_irq00Handler);

  // Interrupción de timer tick habilitada
  picMasterMask(0xFE);
  picSlaveMask(0xFF);
  _sti();
}
```

idtLoader.c

Rutina de atención

```
_irq00Handler:
     irqHandlerMaster 0

%macro irqHandlerMaster 1
     pushState
     mov rdi, %1 ; pasaje de parámetro
     call irqDispatcher

     ; signal pic EOI (End of Interrupt)
     mov al, 20h
     out 20h, al

     popState
     iretq
%endmacro
```

Llama a *irqDispatcher* con el número del IRQ para que se encargue de manejar la interrupción. Luego envía el EOI.

interrupts.asm

Rutina de atención

```
void irqDispatcher(uint64_t irq) {
     switch (irq) {
          case 0:
               int_20();
               break;
     }
     return;
}

void int_20() {
     timer_handler();
}
```

irqDispatcher.c

```
static unsigned long ticks = 0;

void timer_handler() {
     ticks++;
}
```

time.c

Cada vez que interrumpe el timer tick (cada 55 ms) aumenta en 1 la variable *`ticks`*.

Nota: cambiar el `switch` por un arreglo de punteros a función

Resumiendo

El código necesario para cargar su propia interrupción sólo debería involucrar:

1. Definir la rutina de atención de interrupción
2. Cargar la interrupción dentro de *`load_idt`* invocando a *`setup_IDT_entry`* pasandole el número de la interrupción y el puntero a la rutina de atención de interrupción
3. Si la interrupción es de hardware, dicha rutina debe enviar el EOI y deben habilitar el IRQ con la máscara correspondiente *(revisar teórica* ARQ03 - Interrupciones*)*

Pueden hallar información más detallada en http://wiki.osdev.org/Interrupt_Descriptor_Table