

How to boot a PC

Andrei Pitis

Hardware

- Processor
 - protected mode
- Memory
 - Segmentation
- I/O subsystems
 - Interrupt controller (8259)
 - Timer (8253/8254)
 - Keyboard
 - Display

Processor - x86

- Registers:
 - AX, BX, CX, DX
 - SI, DI
 - CS, DS, ES, SS
 - SP, BP, IP
 - Flags
 - GDTR, LDTR, IDTR

Protected mode

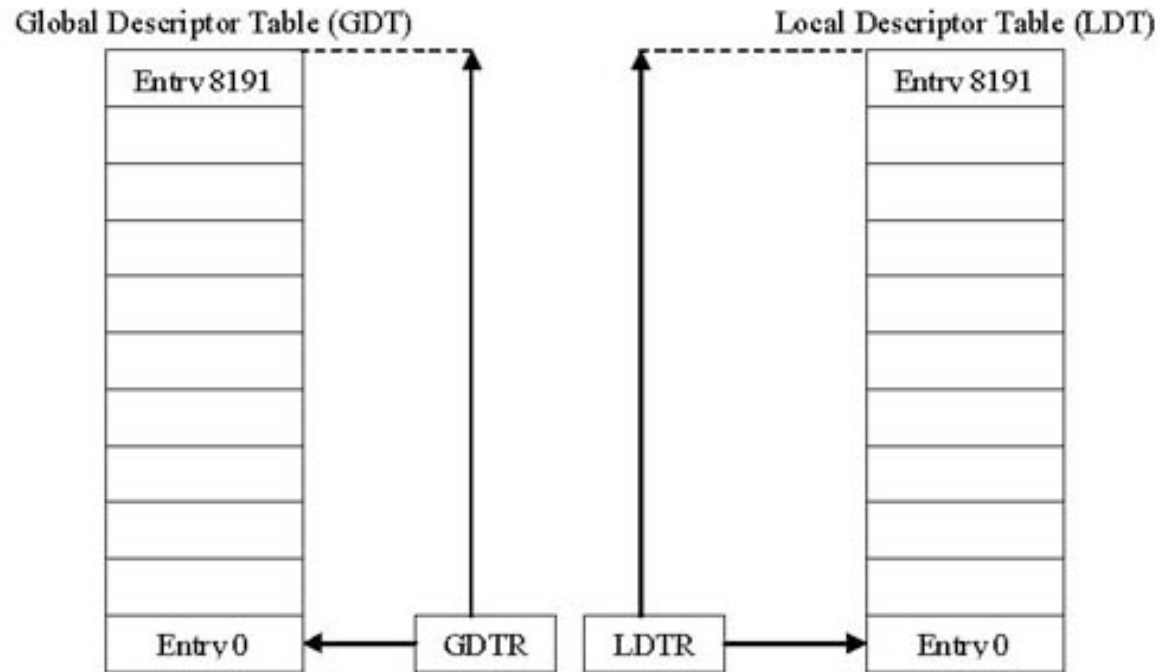
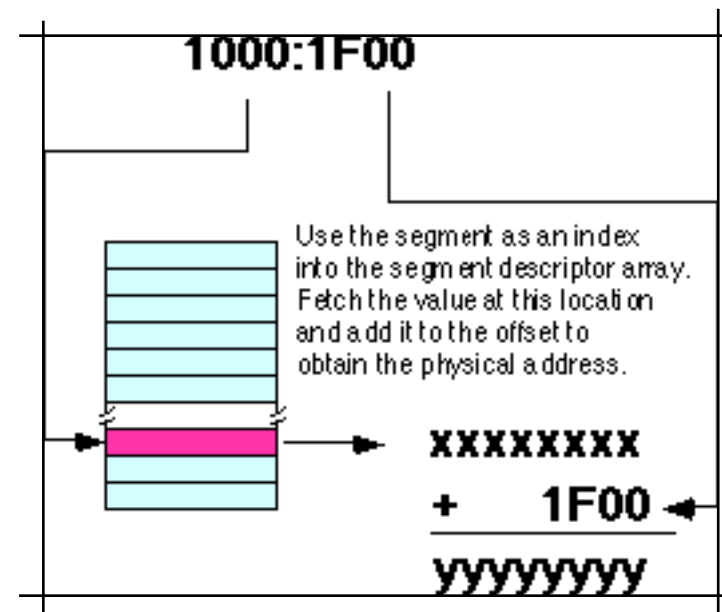
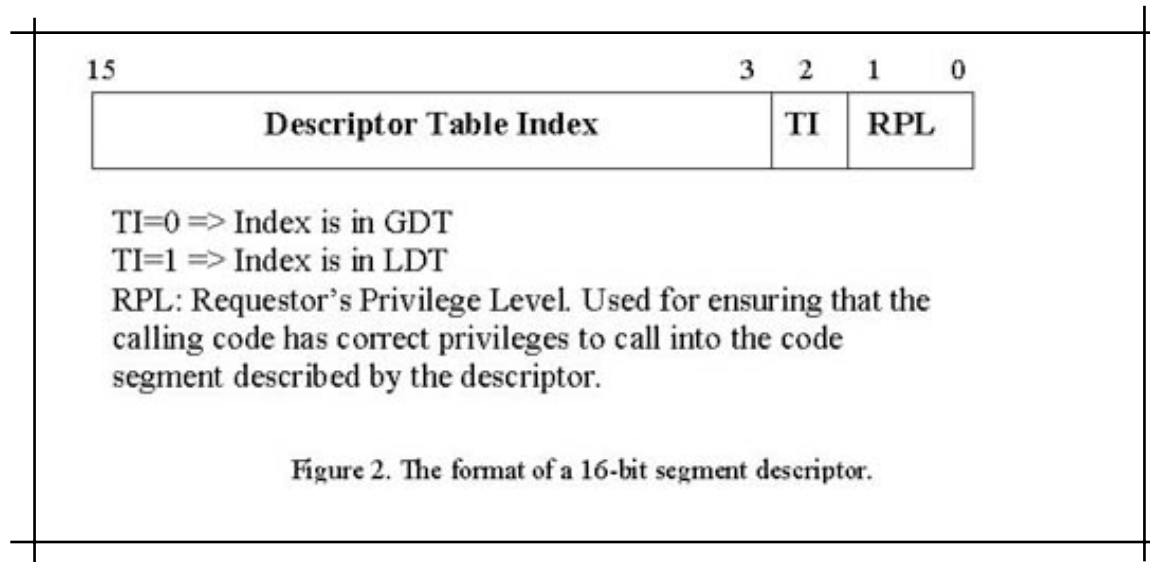


Figure 1. The GDTR and LDTR hold the location and size of the descriptor tables.

```
descriptor struc
limit          dw    0
low            dw    0
highb         db    0
rights        db    0
              dw    0
descriptor ends
```

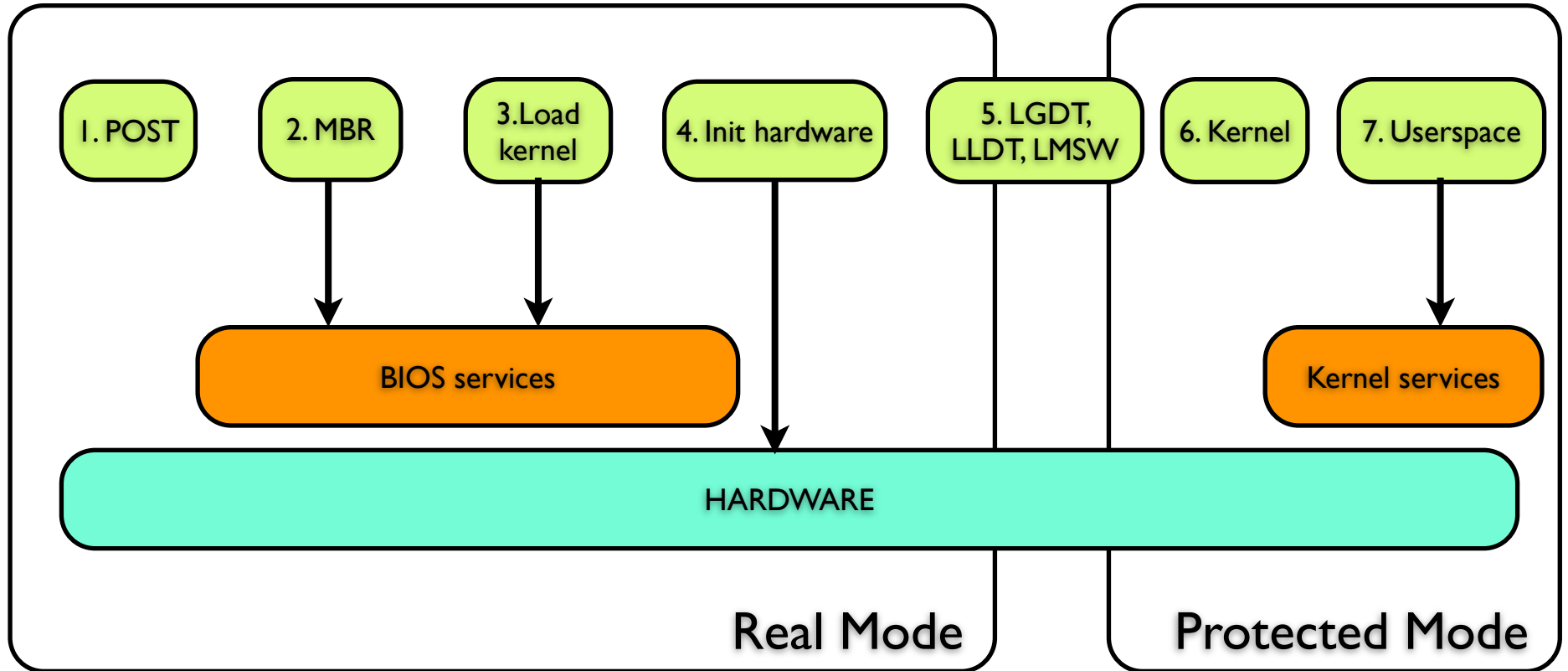
Segmentation



Boot process

- POST
- search boot device: floppy, HD, etc
- Load MBR at 0000:7C00
- jump at that address
- usually the code in the MBR looks for the active partition, loads its first sector also at 7C00 and jumps there
- we will instead load sectors from the disk and jump

Boot sequence



Bootstrap

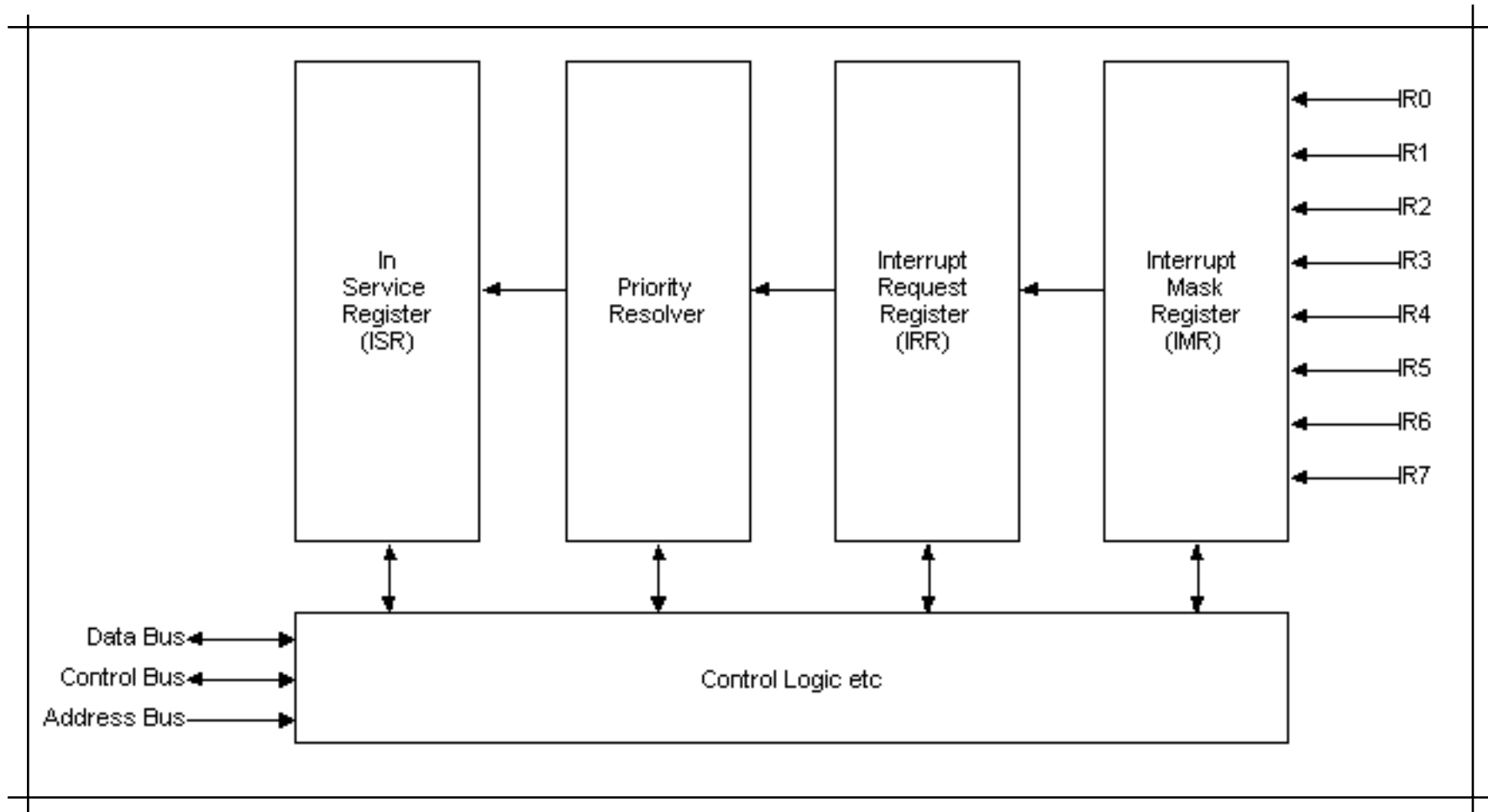
- ```
start: cli

 xor ax,ax
 mov ss,ax
 mov sp,7C00h
 mov si,sp
 mov ds,ax
 mov es,ax
 sti
 cld
 mov di,600h
 mov cx,100h
 repnz movsw
 db 0EAh
 dw 61Dh,0 ; jmp far ptr 0000:061D
```

real\_start:

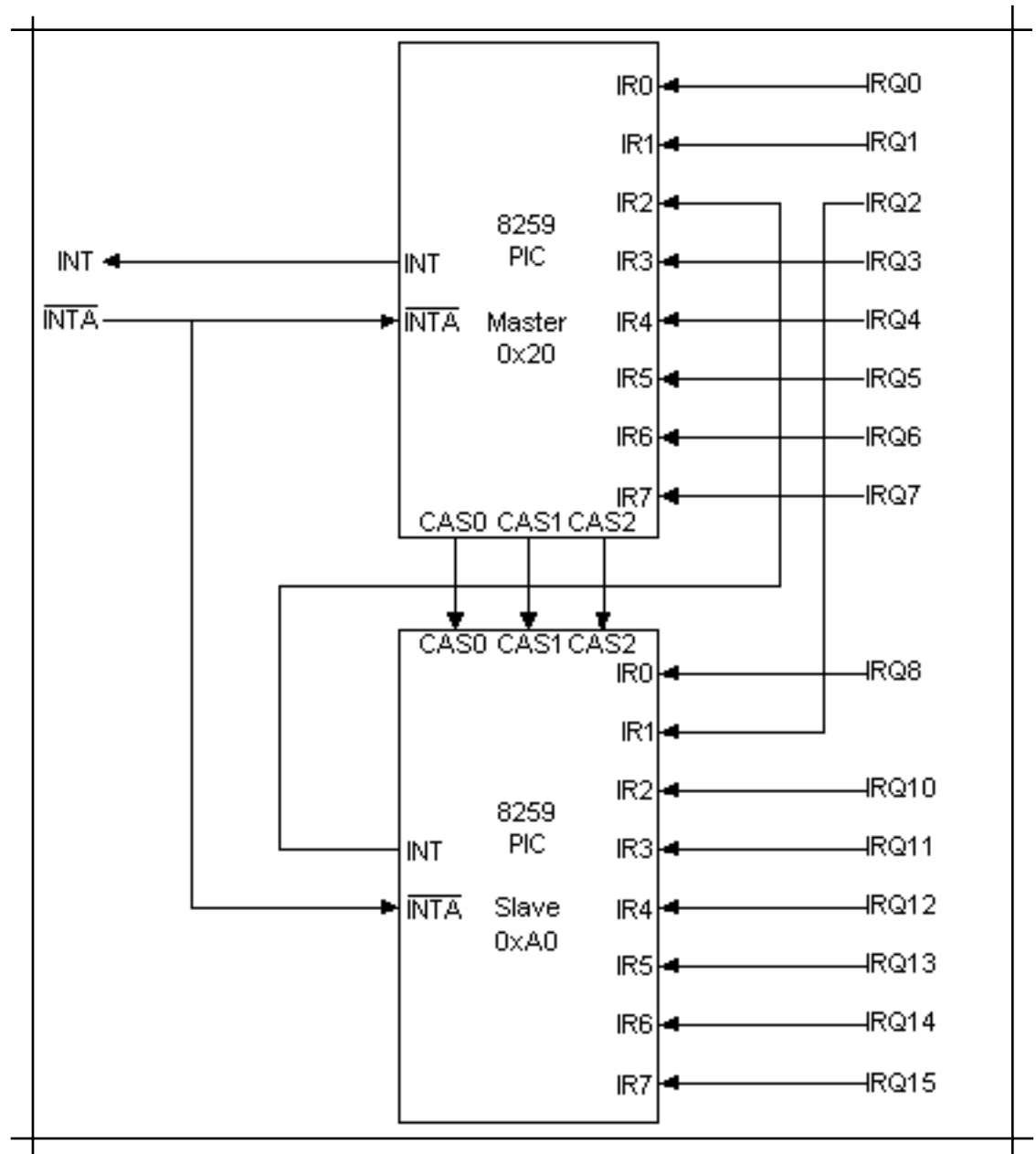


# PIC - 8259



# 8259 - cascade

- The original 8086 had only one PIC
- Needed more devices - cascaded a second PIC



# Interrupts real mode

- 00-01 - Exception handlers
- 02 - NMI
- 03-07 - Exception handlers
- 08-0F - IRQ0 - IRQ7
- 10-6F - Software interrupts
- 70-77 - IRQ8 - IRQ15
- 78-FF - Software interrupts

# Ints - protected mode

- IDT - Interrupt Descriptor Table

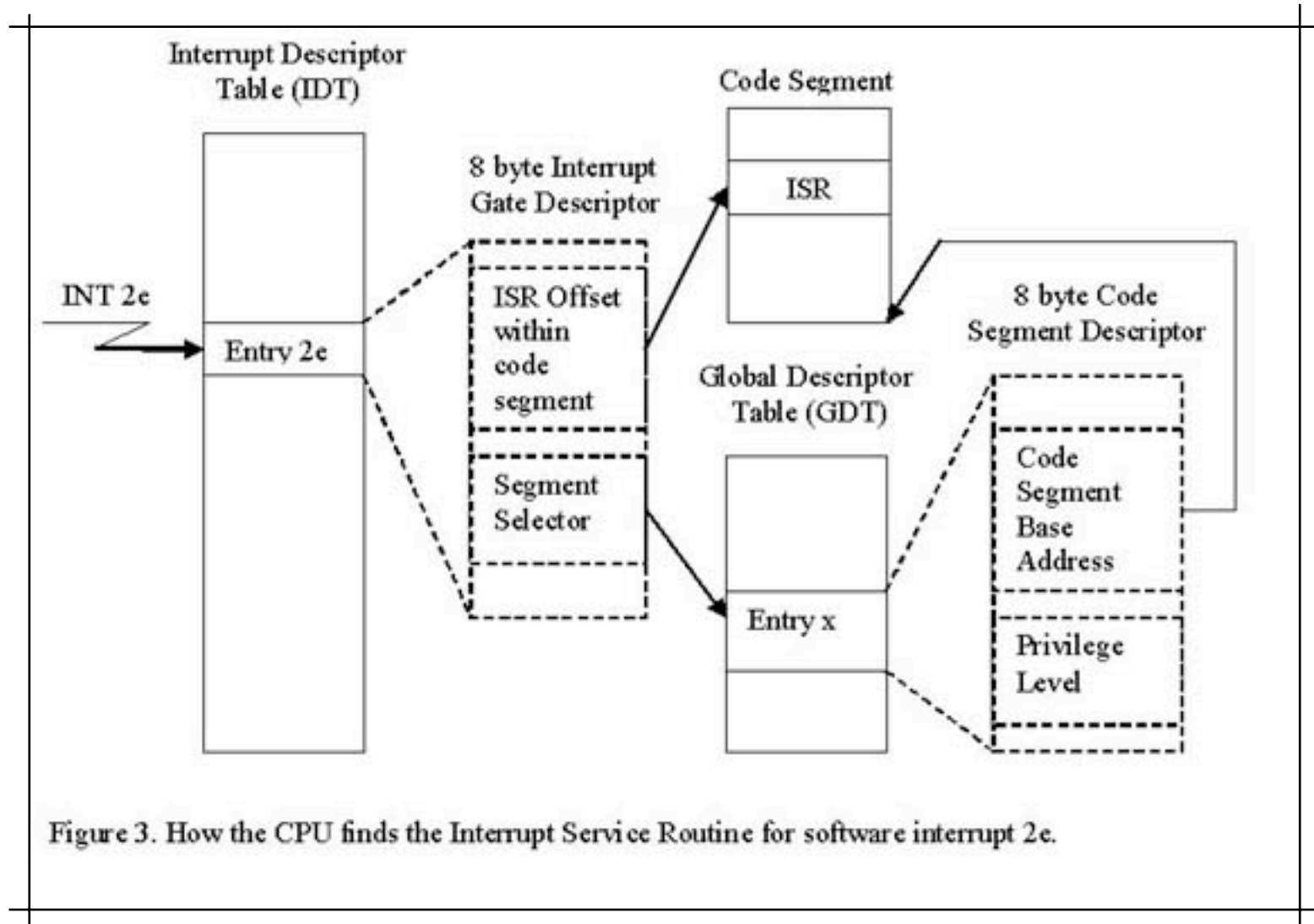


Figure 3. How the CPU finds the Interrupt Service Routine for software interrupt 2e.

# init\_8259

- `init_8259`      `proc`   `near`

```
mov al,11h ; ICW1: edge, call address interval 8, cascaded, sed ICW4
out 20h,al
out 0A0h,al

mov al,20h ; ICW2: first 8 IRQs start at 20h after the reserved exceptions
out 21h,al

mov al,28h ; ICW2: next 8 IRQs are immediately after
out 0A1h,al

mov al,4 ; ICW3: IRQ2 is connected to a slave
out 21h,al

mov al,2
out 0A1h,al ; ICW3: Slave ID 2

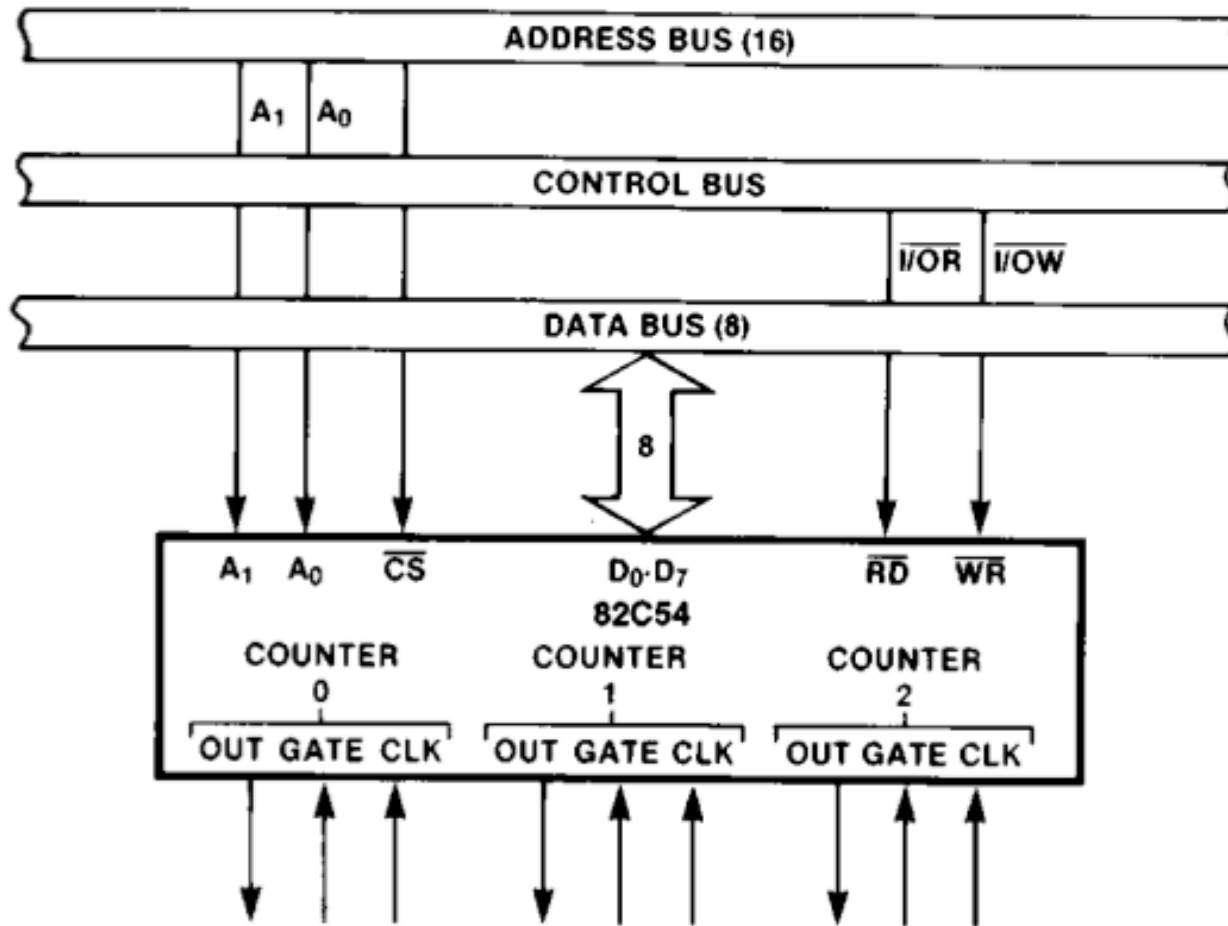
mov al,1 ; ICW4: 8086 Mode
out 021h,al
out 0A1h,al

mov al,0FFh
out 0A1h,al ; OCW1: Mask all ints on PIC2
mov al,0FCh ; OCW1: mask all ints but timer and keyboard on PIC1
out 021h,al

ret

endp
```

# Timer - 8253



231244-7

# Timer - 8253

## Control Word Format

$A_1, A_0 = 11$   $\overline{CS} = 0$   $\overline{RD} = 1$   $\overline{WR} = 0$

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| SC1            | SC0            | RW1            | RW0            | M2             | M1             | M0             | BCD            |

### SC — Select Counter:

| SC1 | SC0 |                                            |
|-----|-----|--------------------------------------------|
| 0   | 0   | Select Counter 0                           |
| 0   | 1   | Select Counter 1                           |
| 1   | 0   | Select Counter 2                           |
| 1   | 1   | Read-Back Command<br>(See Read Operations) |

### M — MODE:

| M2 | M1 | M0 |        |
|----|----|----|--------|
| 0  | 0  | 0  | Mode 0 |
| 0  | 0  | 1  | Mode 1 |
| X  | 1  | 0  | Mode 2 |
| X  | 1  | 1  | Mode 3 |
| 1  | 0  | 0  | Mode 4 |
| 1  | 0  | 1  | Mode 5 |

### RW — Read/Write:

| RW1 | RW0 |                                                                      |
|-----|-----|----------------------------------------------------------------------|
| 0   | 0   | Counter Latch Command (see Read Operations)                          |
| 0   | 1   | Read/Write least significant byte only.                              |
| 1   | 0   | Read/Write most significant byte only.                               |
| 1   | 1   | Read/Write least significant byte first, then most significant byte. |

### BCD:

|   |                                                |
|---|------------------------------------------------|
| 0 | Binary Counter 16-bits                         |
| 1 | Binary Coded Decimal (BCD) Counter (4 Decades) |

**NOTE:** Don't care bits (X) should be 0 to insure compatibility with future Intel products.

# Timer - 8253

- ```
init_8253    proc  near

    mov     al,36h ; LSB then MSB, mode 011
    out     43h,al
    mov     ax,11932 ; clock freq is 1193181.8181..Hz, 1/3 NTSC color subcarrier freq
    jmp     $ + 2
    jmp     $ + 2
    out     40h,al
    jmp     $ + 2
    jmp     $ + 2
    mov     al,ah
    out     40h,al
    rets

    endp
```


Scheduler

- clk proc far ; Task scheduler (100 Hz)

```
push ax
push bx
push ds
mov ax,10h
mov ds,ax
```

```
add task,18h
cmp task,88h
jne jmpfar
mov task,40h
jmp jmpfar
```

```
jmpfar: mov bx,task
and GDT[bx].rights,0FDh
```

```
mov al,20h
out 20h,al
```

```
pop ds
pop bx
pop ax
```

```
task db 0EAh                    ; jmp TSSi (i = 1, 2, 3)
dw 0
dw 40h
```

```
iret
```

Next

- Use the source, Luke!