

The K Project

Memory Management

LSE Team

EPITA

mars 10, 2017

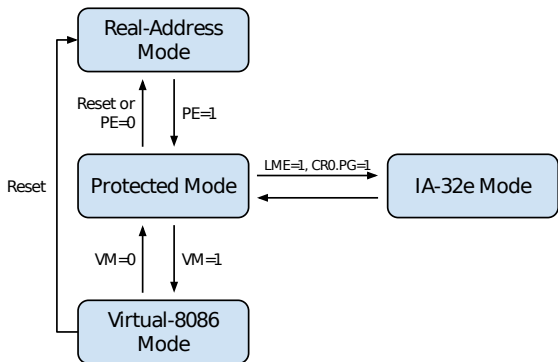


Figure:

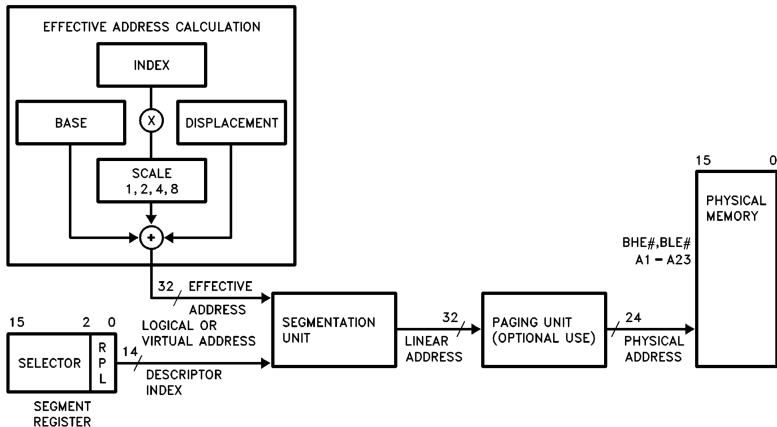


Figure:

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

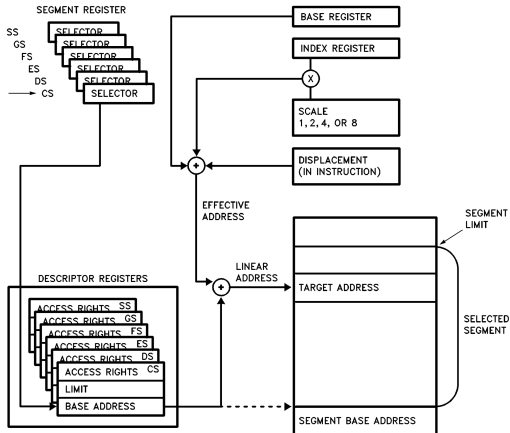


Figure:

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

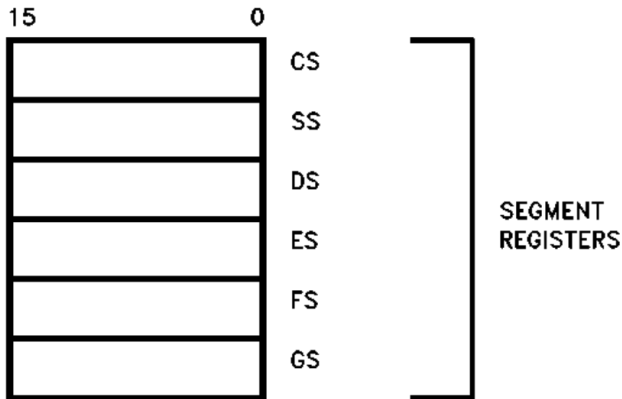


Figure:

- The GDT (Global Descriptor Table) is an array that contains information about every segment.
- Segment Descriptor:
 - base address
 - limit (segment size)
 - segment type (code or data)
 - access rights

First entry must be null. (Don't forget to memset your GDT memory area before setting it)

The K Project

LSE Team

Protected
Mode

Overview

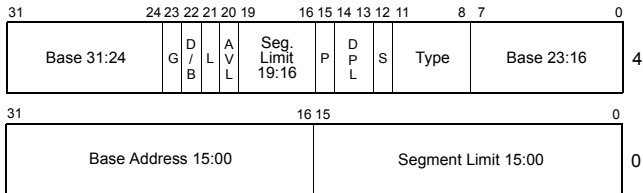
GDT

Segment
selectors
reloading

Protected
mode

Conclusion

- Null segment
- Kernel Code segment
- Kernel Data segment
- Userland Code segment
- Userland Data segment
- Task State Segment (don't worry about that right now)



- L — 64-bit code segment (IA-32e mode only)
- AVL — Available for use by system software
- BASE — Segment base address
- D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
- DPL — Descriptor privilege level
- G — Granularity
- LIMIT — Segment Limit
- P — Segment present
- S — Descriptor type (0 = system; 1 = code or data)
- TYPE — Segment type

Figure:

Type Field					Descriptor Type	Description
Decimal	11	10 E	9 W	8 A		
0	0	0	0	0	Data	Read-Only
1	0	0	0	1	Data	Read-Only, accessed
2	0	0	1	0	Data	Read/Write
3	0	0	1	1	Data	Read/Write, accessed
4	0	1	0	0	Data	Read-Only, expand-down
5	0	1	0	1	Data	Read-Only, expand-down, accessed
6	0	1	1	0	Data	Read/Write, expand-down
7	0	1	1	1	Data	Read/Write, expand-down, accessed
		C	R	A		
8	1	0	0	0	Code	Execute-Only
9	1	0	0	1	Code	Execute-Only, accessed
10	1	0	1	0	Code	Execute/Read
11	1	0	1	1	Code	Execute/Read, accessed
12	1	1	0	0	Code	Execute-Only, conforming
13	1	1	0	1	Code	Execute-Only, conforming, accessed
14	1	1	1	0	Code	Execute/Read, conforming
15	1	1	1	1	Code	Execute/Read, conforming, accessed

Figure:

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloadingProtected
mode

Conclusion

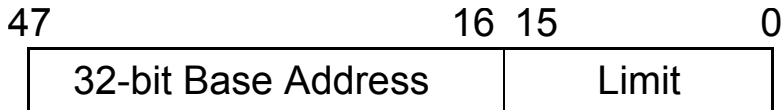


Figure:

```
gdt_r gdtr;
```

```
gdtr.base = gdt;           /* gdt base address */  
gdtr.limit = sizeof(gdt) - 1 /* gdt size - 1 */
```

```
asm volatile("lgdt %0\n"  
             : /* no output */  
             : "m" (gdtr)  
             : "memory");
```

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

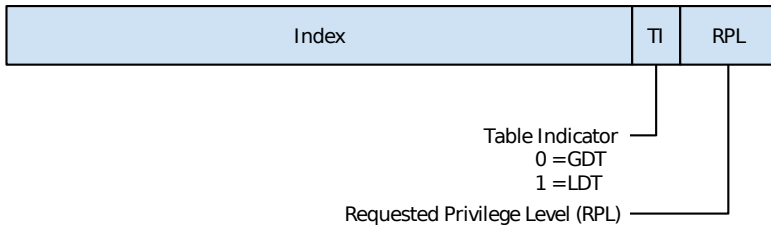


Figure:

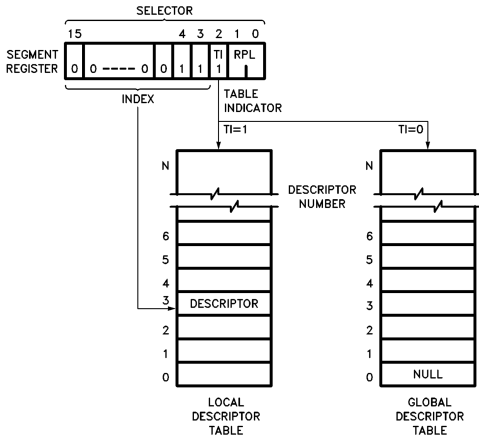


Figure:

Switching to Protected Mode

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

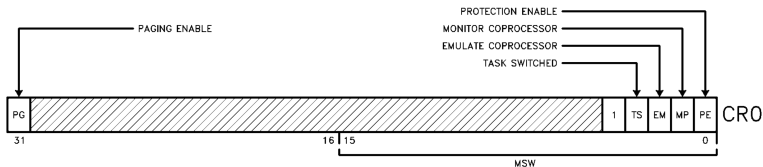


Figure:

You have to use an auxilliary register to set cr0:

```
movl $0x42, %cr0 ; It _won't_ assemble
```

```
movl $0x42, %eax
```

```
movl %eax, %cr0 ; OK
```

Likewise, setting data in segment register:

```
movw $0x42, %ax
```

```
movw %ax, %ds
```

```
movw %ax, %fs
```

```
movw %ax, %gs
```

```
movw %ax, %ss
```


Methode 1

```
pushl $0x42      ; push %cs on the stack
pushl $1f         ; push %eip on the stack
lret              ; far return
1:                ; After the lret you will get
                  ; here, with cs set to 0x42
```

Methode 2

```
ljmp $0x42, %1    ; long jump
1:
```

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

- Build GDT
- Load GDT
- Set PE flag in CR0
- Reload segment selectors

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

- Use packed struct and bitfields
- Write a GDT Pretty Printer

The K Project

LSE Team

Protected
Mode

Overview

GDT

Segment
selectors
reloading

Protected
mode

Conclusion

- #k (irc.rezosup.org)
- epita.cours.k
- k[at]lse.epita.fr
- naam[at]lse.epita.fr
- nurelin[at]lse.epita.fr