Babystep6

From OSDev Wiki

Entering Protected mode is actually done by switching a single bit in a special control register (cr0). (All the other stuff, like A20Line, tasks, IDT, call gates, etc. is additional stuff.)

However, before switching to pmode, you have to use the LGDT instruction to load another special register (gdtr) with the location of a table of data structures called descriptors that tell the process how to access memory.

Babystep6

Tutorial

Previous	**Next**
Babystep5	Babystep7


```
We're arguing about whether GDT could be set up after switching to pmode in this thread
--PypeClicker
```

Descriptors

Overview of bytes in the descriptor:

```
+0 +1 +2 +3  +4 +5 +6 +7
l0 l1 b0 b1  b2 TT Fl b3
```

Descriptor bytes arranged from lowest memory location to highest:

0	`0x00` lowest byte of Limit
1	`0x00` next byte of Limit
2	`0x00` lowest byte of Base Addr
3	`0x00` next byte of Base Addr
4	`0x00` third byte of Base Addr
5	`0x00` = (bits) 0 - 00 - 0 - 0000 = P - DPL - S - Type
6	`0x00` = (bits) 0 - 0 - 0 - 0 - 0000 = G - D/B - R - AVL - Size
7	`0x00` fourth and highest byte of Base Addr

Bits in Type (byte #5)

P

Present (1 bit) = 1 means segment is in memory (accessing a non-present segment will raise an exception)

DPL
: Descriptor Privilege Level (2 bits) = 0 is most privileged and 3 is least.

S
: System (1 bit) = must be 0 in descriptors for Task State Segments (TSS), Interrupt Gate, Trap Gate, Task Gate, Call Gates. Otherwise, for code/data/stack segment descriptors, it will be 1.

Type
: Type (4 bits) = interpretation of these depends on whether S (above) is set or not. For S=0, the interpretation will be covered in specific instances of gates etc.

Type bit 3
: If S=1, then if high bit is 1, it's a code segment, otherwise it's a data segment.

Type bit 2
: The next highest bit depends on the highest bit. If code segment, this next bit indicates whether the segment is 'Conforming' or not. This allows programs somewhere else that are LESS privileged to access this segment, then this segment conforms to the privilege level of the calling program. If it's a data segment, this bit specifies "Expand (up or down)" for when the segment is used as a stack. Expand-up (bit=0) is your normal stack behavior. Expand-down is used to prevent problems in stacks that are resized.

Type bit 1
: The subsequent bit specifies permission to Read/Write. For data segments, 0 means read-only and 1 is r/w. For code segments, 0 means you can't read from it (e.g. using MOV) and 1 means you can.

Type bit 0
: The lowest bit means that the segment has been accessed already (1) or not.

Bits in Flags (byte #6)

G
: Granularity (1 bit) = segment Size specified in bytes (0) or 4K pages (1)

D/B
: Default (code seg) / Big (data seg) = (1 bit) In a code segment (see "Type" above), this bits says default operand/address size is 32-bit (1) or 16-bit (0). For a data segment, it means stack pointer is 32-bit (1) or 16-bit (0). Also means something for expand-down stacks (see "Type" above), but we don't care.

R
: Reserved (1 bit) = belongs to the Intel of the future.

AVL
: Available (1 bit) = For your use. Go crazy.

Size
: Top Nibble of Size (4 bits) = The size of the segment is 20-bits. This is the final four. Whether it means the highest possible segment size is 1 meg or 4 Gigs depends on Granularity above.

See Also

- Global Descriptor Table
- Segmentation

Category: Babystep
