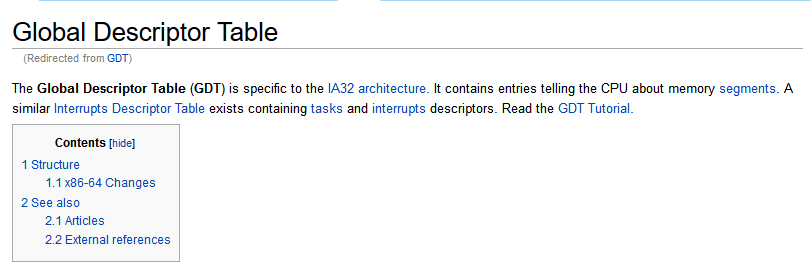
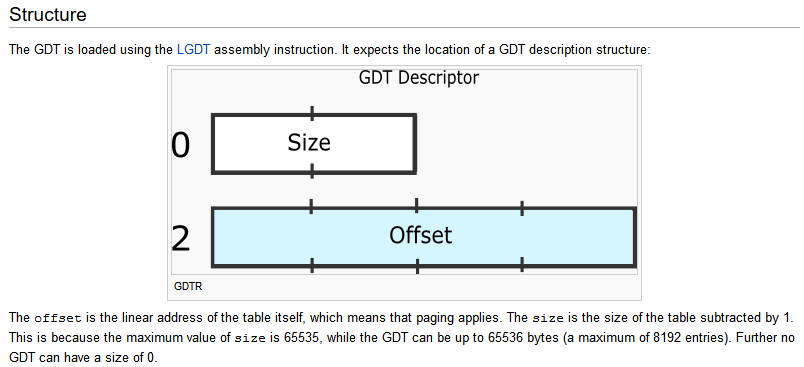
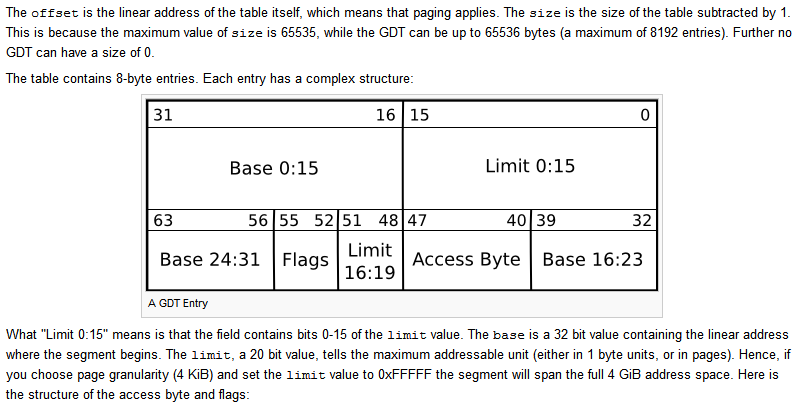
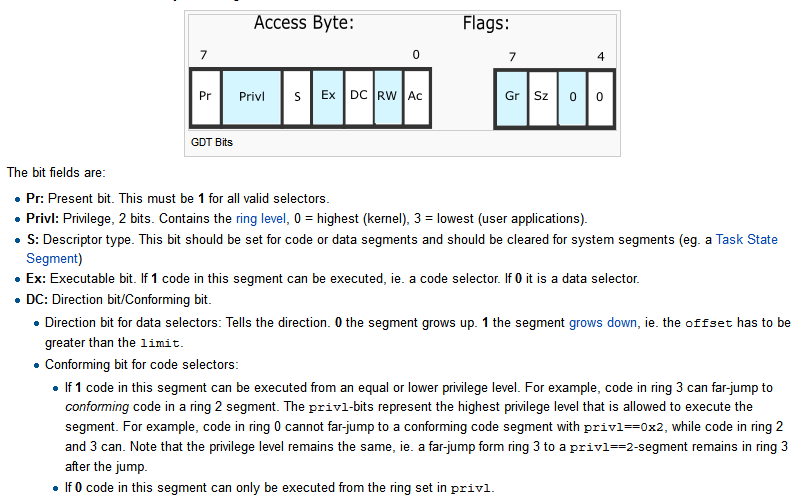
REF:

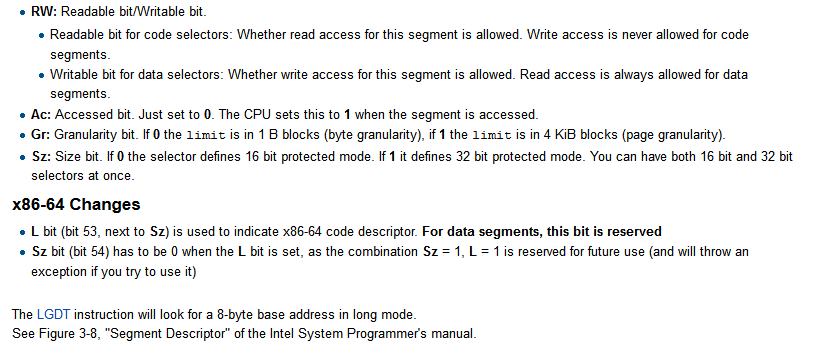
<https://wiki.osdev.org/GDT>

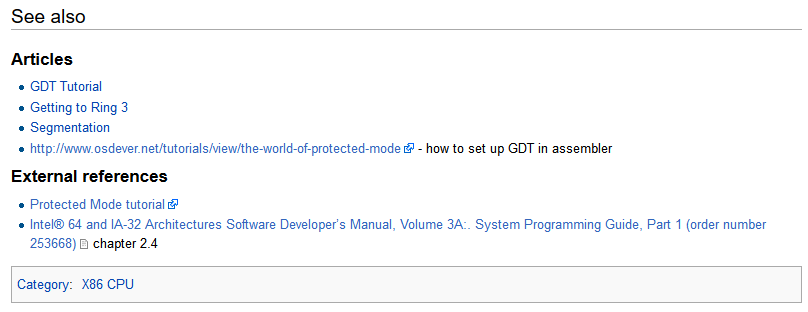












REF:

https://stackoverflow.com/questions/9137947/assembler-jump-in-protected-mode-with-gdt

# [Assembler jump in Protected Mode with GDT](https://stackoverflow.com/questions/9137947/assembler-jump-in-protected-mode-with-gdt)

Asked 9 years, 1 month ago

Active [1 year, 11 months ago](https://stackoverflow.com/questions/9137947/assembler-jump-in-protected-mode-with-gdt?lastactivity)

Viewed 9k times

6

I am currently playing around with x86 Assember in order to sharpen my low-level programming skills. Currently, I am facing a little problem with the addressing scheme in 32-Bit Protected Mode.

The situation is the following:

I have a Program loaded at 0x7e0 which switches the CPU to Protected Mode and jumps to the according label in the code:

[...]

code to switch CPU in Protected Mode

[...]

jmp ProtectedMode

[...]

bits 32

ProtectedMode:

.halt:

hlt

jmp .halt

This works absolutely fine so far. The "jmp ProtectedMode" works without a explicit far jump to clear the prefetch queue - since this program is loaded with offset 0 (org 0 at the beginning) - causing the code segment pointing to the right location.

My current problem now is, that within the "ProtectedMode" label I want to jump to an other program which is loaded at 0x8000 (I checked this with a memory dump, the loading function did work properly and the program is loaded correctly to 0x8000).

Since the CPU is now in ProtectedMode and not RealMode anymore, the addressing schema is different. ProtectedMode uses descriptor selectors to lookup a base address and a limit in a descriptor table to add the given offset and retrieve the physical address (as I understood). Therefore, it was necessary to install a GDT before entering ProtectedMode.

Mine is looking like the following:

%ifndef \_\_GDT\_INC\_INCLUDED\_\_

%define \_\_GDT\_INC\_INCLUDED\_\_

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Global Descriptor Table (GDT) \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NULL\_DESC:

dd 0 ; null descriptor

dd 0

CODE\_DESC:

dw 0xFFFF ; limit low

dw 0 ; base low

db 0 ; base middle

db 10011010b ; access

db 11001111b ; granularity

db 0 ; base high

DATA\_DESC:

dw 0xFFFF ; data descriptor

dw 0 ; limit low

db 0 ; base low

db 10010010b ; access

db 11001111b ; granularity

db 0 ; base high

gdtr:

Limit dw 24 ; length of GDT

Base dd NULL\_DESC ; base of GDT

%endif ;\_\_GDT\_INC\_INCLUDED\_\_

and is loaded to the GDT register via

lgdt [gdtr]

What I did not understand so far is, how do I now jump to the physical address 0x8000 in ProtectedMode using the GDT?

My first thoughts were to select the Code Descriptor (CODE\_DESC) which should point to 0x7e00 (were the current program is loaded) and use the offset that is necessary to get to 0x8000 (512 bytes), resulting in the jump instruction:

jmp CODE\_DESC:0x200

But this does not work.

jmp 0x7e0:0x200

does not work either...

Do you have any idea what I am missing here? Maybe I did not understand something essential within the 32-Bit ProtectedMode addressing scheme and the usage of the GDT.

**[EDIT] Complete code:**

bits 16

org 0 ; loaded with offset 0000 (phys addr: 0x7e00)

jmp Start

Start:

xor ax, ax

mov ax, cs

mov ds, ax ; update data segment

cli ; clear interrupts

lgdt [gdtr] ; load GDT from GDTR (see gdt\_32.inc)

call OpenA20Gate ; open the A20 gate

call EnablePMode ; jumps to ProtectedMode

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Opens A20 Gate \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

OpenA20Gate:

in al, 0x93 ; switch A20 gate via fast A20 port 92

or al, 2 ; set A20 Gate bit 1

and al, ~1 ; clear INIT\_NOW bit

out 0x92, al

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Enables Protected Mode \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EnablePMode:

mov eax, cr0

or eax, 1

mov cr0, eax

jmp ProtectedMode ; this works (jumps to label and halts)

;jmp (CODE\_DESC-NULL\_DESC):ProtectedMode ; => does not work

;jmp 08h:ProtectedMode , => does not work

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* data fields \*

;\* &includes \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%include "gdt\_32.inc"

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Protected Mode \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bits 32

ProtectedMode:

;here I want to jump to physical addr 0x8000 (elf64 asm program)

.halt:

hlt

jmp .halt

## 2 Answers

14

There are multiple problems in the code.

First, your GDTR.Base contains the offset of the GDT from the beginning of the code since your code is compiled to begin at address 0 (because of org 0). The base address must be the physical address, not a relative address. IOW, if you keep this org 0, you must add CS\*16 (=0x7e00) to Base.

Second, because of that same org 0, the 32-bit offsets in your code (after bits 32 and ProtectedMode:) aren't equal to physical addresses they correspond to, they're 0x7e00 less than the physical addresses. OTOH, the segments defined in your GDT start at physical address 0 (because the base portions of the GDT entries are 0's) and not at 0x7e00. This means that when you try to use these segments with your code/data, you'll be missing the addresses by 0x7e00. If you want to keep org 0, the base addresses in the GDT must be set to 0x7e00.

Or you can change org 0 to org 0x7e00 and then the bases in the GDT should be 0. And you won't need to adjust GDTR.Base by 0x7e00, 0 will do.

This should work:

bits 16

org 0x7e00 ; loaded at phys addr 0x7e00

; control must be transferred with jmp 0:0x7e00

xor ax, ax

mov ds, ax ; update data segment

cli ; clear interrupts

lgdt [gdtr] ; load GDT from GDTR (see gdt\_32.inc)

call OpenA20Gate ; open the A20 gate

call EnablePMode ; jumps to ProtectedMode

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Opens A20 Gate \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

OpenA20Gate:

in al, 0x93 ; switch A20 gate via fast A20 port 92

or al, 2 ; set A20 Gate bit 1

and al, ~1 ; clear INIT\_NOW bit

out 0x92, al

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Enables Protected Mode \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EnablePMode:

mov eax, cr0

or eax, 1

mov cr0, eax

jmp (CODE\_DESC - NULL\_DESC) : ProtectedMode

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* data fields \*

;\* &includes \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;%include "gdt\_32.inc"

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Global Descriptor Table (GDT) \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NULL\_DESC:

dd 0 ; null descriptor

dd 0

CODE\_DESC:

dw 0xFFFF ; limit low

dw 0 ; base low

db 0 ; base middle

db 10011010b ; access

db 11001111b ; granularity

db 0 ; base high

DATA\_DESC:

dw 0xFFFF ; limit low

dw 0 ; base low

db 0 ; base middle

db 10010010b ; access

db 11001111b ; granularity

db 0 ; base high

gdtr:

Limit dw gdtr - NULL\_DESC - 1 ; length of GDT

Base dd NULL\_DESC ; base of GDT

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Protected Mode \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bits 32

ProtectedMode:

mov ax, DATA\_DESC - NULL\_DESC

mov ds, ax ; update data segment

.halt:

hlt

jmp .halt

Note that a segment limit equals the segment size minus

1.

A few more points... Load all segment registers with valid selectors or 0. Also, set up the stack. If you have garbage there (or old values from real mode), when you start playing with interrupts/exceptions, you'll have more crashes.

Finally, I don't know what elf64 is, but you will have to take care of the org thing for other modules and make sure all generated addresses correspond to load addresses. And if you intend to enable 64-bit mode, there's a ton of work to do. I'd advise not to rush into 64-bit mode yet since you're tripping over relatively simple stuff.

3

A couple things. First, your current code does not technically enter protected mode. You enter protected mode by loading cs with a descriptor from the GDT. Since you can't directly set the cs register, the easiest way to do this is using a far jump. Replace your current jump with:

jmp (CODE\_DESC-NULL\_DESC):ProtectedMode

Second, the base for your code segment is 0, not 0x7e00. If you look at the four bytes labeled with the word "base", they are all 0. You have two options. Either change your GDT to have a base of 0x7e00, or add directives to change the load address for all protected mode code for a base of 0.

Once you have done both of these things, you can jump to your program using a normal jump instruction. If you choose to leave your GDT as it is, you would use the full address:

jmp 0x8000

If you choose to change the base of your code segment, you will need to use the address relative to that.

jmp 0x200

[More information about the GDT](http://wiki.osdev.org/GDT)  
[More information about entering protected mode](http://wiki.osdev.org/Journey_To_The_Protected_Land)