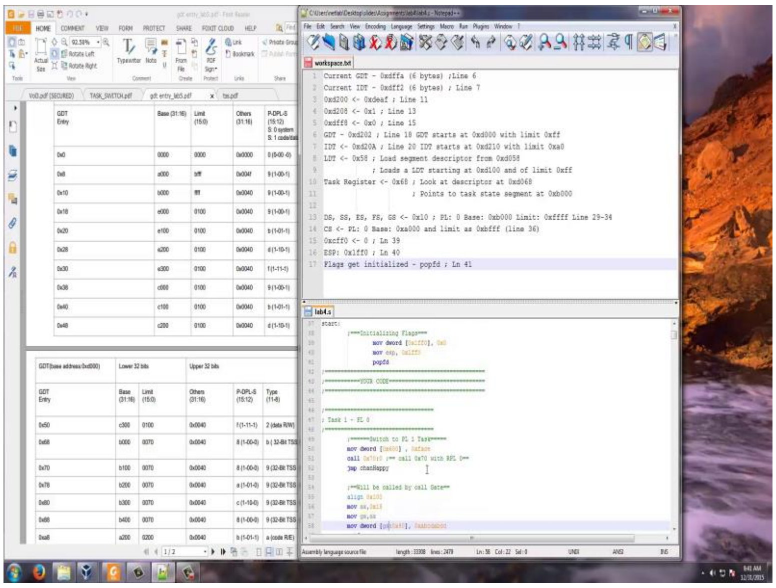
Information Security – II
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 32
Lab4 Part 2 - Week 6

(Refer Slide Time: 00:09)



GDT Entry	Base (31:16)	Limit (15:0)	Others (31:16)	P-DPL-S (15:12) S: 0 system S: 1 code/data
0x0	0000	0000	0x0000	0 (0-00 -0)
0x8	a000	bfff	0x004f	9 (1-00-1)
0x10	b000	ffff	0x0040	9 (1-00-1)
0x18	e000	0100	0x0040	9 (1-00-1)
0x20	e100	0100	0x0040	b (1-01-1)
0x28	e200	0100	0x0040	d (1-10-1)
0x30	e300	0100	0x0040	f (1-11-1)
0x38	c000	0100	0x0040	9 (1-00-1)
0x40	c100	0100	0x0040	b (1-01-1)
0x48	c200	0100	0x0040	d (1-10-1)

GDT (base address:0xd000)	Lower 32 bits		Upper 32 bits		
GDT Entry	Base (31:16)	Limit (15:0)	Others (31:16)	P-DPL-S (15:12)	Type (11-8)
0x50	c300	0100	0x0040	f (1-11-1)	2 (data R/W)
0x68	b000	0070	0x0040	8 (1-00-0)	b (32-Bit TSS
0x70	b100	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS
0x78	b200	0070	0x0040	a (1-01-0)	9 (32-Bit TSS
0x80	b300	0070	0x0040	c (1-10-0)	9 (32-Bit TSS
0x88	b400	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS
0xa8	a200	0200	0x0040	b (1-01-1)	a (code R/E)

workspace.txt

```
Current GDT - 0xdffa (6 bytes) ;Line 6
Current IDT - 0xdff2 (6 bytes) ; Line 7
0xd200 <- 0xdeaf ; Line 11
0xd208 <- 0x1 ; Line 13
0xdff8 <- 0x0 ; Line 15
GDT - 0xd202 ; Line 18 GDT starts at 0xd000 with limit 0xff
IDT <- 0xd20A ; Line 20 IDT starts at 0xd210 with limit 0xa0
LDT <- 0x58 ; Load segment descriptor from 0xd058
            ; Loads a LDT starting at 0xd100 and of limit 0xff
Task Register <- 0x68 ; Look at descriptor at 0xd068
                      ; Points to task state segment at 0xb000

DS, SS, ES, FS, GS <- 0x10 ; PL: 0 Base: 0xb000 Limit: 0xffff Line 29-34
CS <- PL: 0 Base: 0xa000 and limit as 0xbfff (line 36)
0xcff0 <- 0 ; Ln 39
ESP: 0x1ff0 ; Ln 40
Flags get initialized - popfd ; Ln 41
```

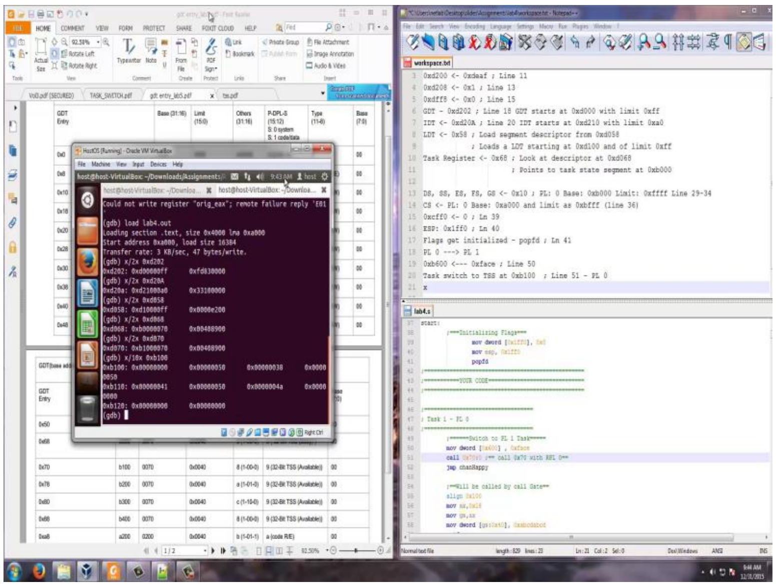
lab4.s

```
37 start:
38         ;===Initializing Flags===
39                 mov dword [0x1ff0], 0x0
40                 mov esp, 0x1ff0
41                 popfd
42 ;==========================================
43 ;==========YOUR CODE=======================
44 ;==========================================
45
46 ;================================
47 ; Task 1 - PL 0
48 ;================================
49         ;======Switch to PL 1 Task=====
50         mov dword [0xb600] , 0xface
51         call 0x70:0 ;== call 0x70 with RPL 0==
52         jmp chanHappy
53
54         ;==Will be called by call Gate==
55         align 0x100
56         mov ax,0x13
57         mov ds,ax
58         mov dword [ds:0x40], 0xabcdabcd
```

Assembly language source file length : 33308 lines : 2479 Ln : 58 Col : 22 Sel : 0 UNIX ANSI INS

What we do in our code is as follows. Let us go and do step by step, now from PL0, I have to move to PL1. So, before going from PL0 to PL1, just to say that I have reached there, I go and move to 0xb600, why b600, because ds is b000. So, b600 I move to 0xb600, the value phases. Now, I go and say, this is line number 51 of your code, I go and say, call 70 colon 0. So, what is 70? 70 here, is a task stage segment that starts at b100 and limit 70. Now, the line 52, this is a task switch to tss at 0xb100 and that happens in line number 50, previous one this happens at line number 51, the code as you see here, this called 0x70.

(Refer Slide Time: 02:05)



```
Could not write register "orig_eax"; remote failure reply 'E01
.
(gdb) load lab4.out
Loading section .text, size 0x4000 lma 0xa000
Start address 0xa000, load size 16384
Transfer rate: 3 KB/sec, 47 bytes/write.
(gdb) x/2x 0xd202
0xd202: 0xd00000ff      0xfd830000
(gdb) x/2x 0xd20A
0xd20a: 0xd21000a0      0x33100000
(gdb) x/2x 0xd058
0xd058: 0xd10000ff      0x0000e200
(gdb) x/2x 0xd068
0xd068: 0xb0000070      0x00408900
(gdb) x/2x 0xd070
0xd070: 0xb1000070      0x00408900
(gdb) x/10x 0xb100
0xb100: 0x00000000      0x00000050      0x00000038      0x0000
0050
0xb110: 0x00000041      0x00000050      0x0000004a      0x0000
0000
0xb120: 0x00000000      0x00000000
(gdb) 
```

GDT Entry	Base (31:16)	Limit (15:0)	Others (31:16)	P-DPL-S (15:12) S: 0 system S: 1 code/data	Type (11-8)	Base (7:0)
0x70	b100	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS (Available))	00
0x78	b200	0070	0x0040	a (1-01-0)	9 (32-Bit TSS (Available))	00
0x80	b300	0070	0x0040	c (1-10-0)	9 (32-Bit TSS (Available))	00
0x88	b400	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS (Available))	00
0xa8	a200	0200	0x0040	b (1-01-1)	a (code R/E)	00

workspace.txt

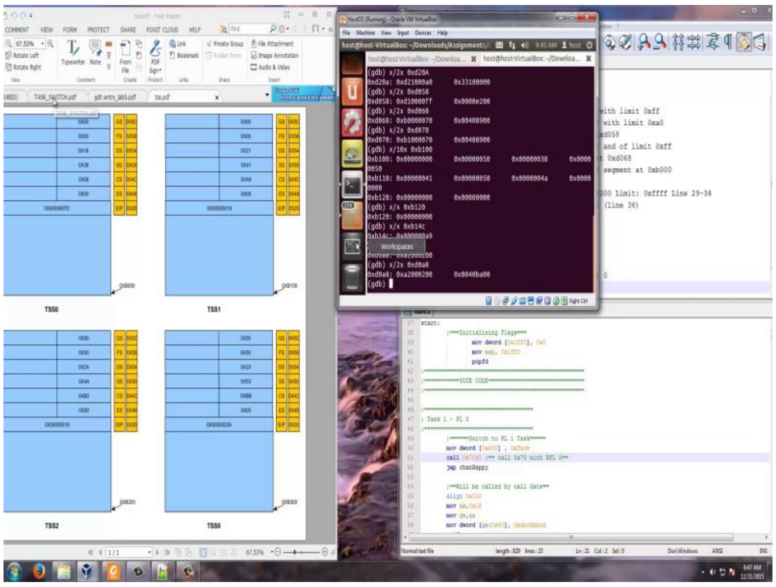
```
0xd200 <- 0xdeaf ; Line 11
0xd208 <- 0x1 ; Line 13
0xdff8 <- 0x0 ; Line 15
GDT - 0xd202 ; Line 18 GDT starts at 0xd000 with limit 0xff
IDT <- 0xd20A ; Line 20 IDT starts at 0xd210 with limit 0xa0
LDT <- 0x58 ; Load segment descriptor from 0xd058
            ; Loads a LDT starting at 0xd100 and of limit 0xff
Task Register <- 0x68 ; Look at descriptor at 0xd068
                      ; Points to task state segment at 0xb000

DS, SS, ES, FS, GS <- 0x10 ; PL: 0 Base: 0xb000 Limit: 0xffff Line 29-34
CS <- PL: 0 Base: 0xa000 and limit as 0xbfff (line 36)
0xcff0 <- 0 ; Ln 39
ESP: 0x1ff0 ; Ln 40
Flags get initialized - popfd ; Ln 41
PL 0 ---> PL 1
0xb600 <--- 0xface ; Line 50
Task switch to TSS at 0xb100  ; Line 51 - PL 0
x
```

lab4.s

```
37 start:
38         ;===Initializing Flags===
39                 mov dword [0x1ff0], 0x0
40                 mov esp, 0x1ff0
41                 popfd
42 ;==============================================
43 ;==========YOUR CODE===========================
44 ;==============================================
45 
46 ;================================
47 ; Task 1 - PL 0
48 ;================================
49         ;======Switch to PL 1 Task=====
50         mov dword [0xb600] , 0xface
51         call 0x70:0 ;== call 0x70 with RPL 0==
52         jmp chanHappy
53 
54         ;==Will be called by call Gate==
55         align 0x100
56         mov ax,0x18
57         mov gs,ax
58         mov dword [gs:0x40], 0xabcdabcd
```

Now, let us go back and see what is there in this. We have described here some of the things that we need to see, is that at 70, it is at 0xd070. This is the 17th descriptor because your GDT starts at d000; I am loading 0x70 and line number 51. So, 0 this is again calls task stage segment with privilege one and B070 is a privilege one and it is having b100 as the task stage segment. This is a privilege 0 code can call this pointing at to a task stage segment starting at b100. Now, let us go and see at b100, s slash 10x0xb100, now, this is how the task stage segment is going to look like.

(Refer Slide Time: 03:57)

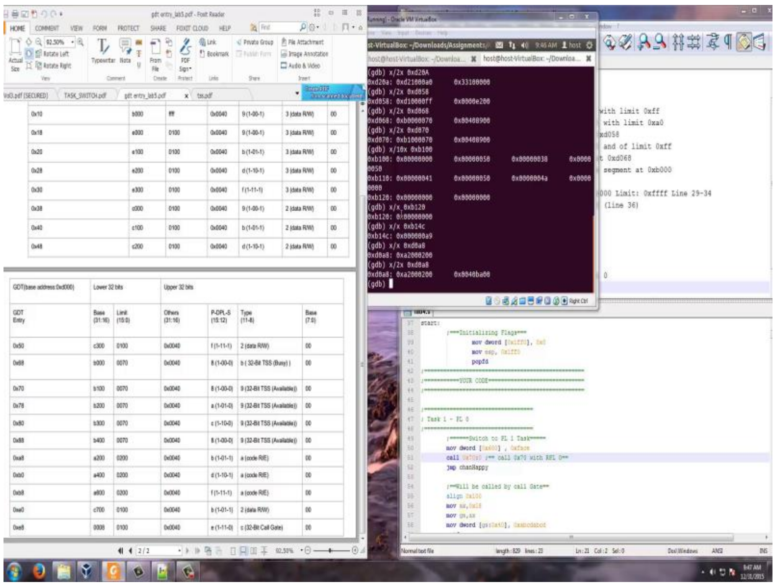


```
(gdb) x/2x 0xd20A
0xd20a: 0xd21000a0      0x33100000
(gdb) x/2x 0xd058
0xd058: 0xd10000ff      0x0000e200
(gdb) x/2x 0xd068
0xd068: 0xb0000070      0x00400900
(gdb) x/2x 0xd070
0xd070: 0xb1000070      0x00400900
(gdb) x/10x 0xb100
0xb100: 0x00000000      0x00000050      0x00000038      0x0000
0050
0xb110: 0x00000041      0x00000050      0x0000004a      0x0000
0000
0xb120: 0x00000000      0x00000000
(gdb) x/x 0xb120
0xb120: 0x00000000
(gdb) x/x 0xb14c
0xb14c: 0x000000a9
[illegible]
0xd0a8: 0xa2000200
(gdb) x/2x 0xd0a8
0xd0a8: 0xa2000200      0x0040ba00
(gdb)
```

```
start:
        ;***Initializing Flags***
                mov dword [0xfff0], 0x0
                mov esp, 0xfff0
                popfd
;=================================================
;===========YOUR CODE=============================
;=================================================

;=================================
; Task 1 - PL 0
;=================================
        ;======Switch to PL 1 Task=====
        mov dword [0x600] , 0xface
        call 0x70:0 ;** call 0x70 with RPL 0**
        jmp chanHappy

        ;**Will be called by call Gate**
        align 0x100
        mov ax,0x18
        mov gs,ax
        mov dword [gs:0x40], 0xabcdabcd
```

Let us see, how this is going to fair. Let us very quickly look into what is there at b100. Now, your e i p should be at b120 because when I start executing I should start executing with the e i p, here this is how the task which happens. So, s slash 0xb120, it starts at 00 offset; this is after what you see on the left hand side is after the execution. But before execution, the initial stage, your e i p is 0 and then s slash x0xb14c will basically give you code segment a9, a9 is the privilege one code segment. Let us go and see where should it be a9, a9 should be at s slash x0xd0a8 s slash 2x. Please note that, this is a code segment that starts at 200 of a size and this is of privilege one. The interpretation of these 4 bytes, 8 bytes, this is at descriptor at d 0.

(Refer Slide Time: 06:10)



0x10	b000	fff	0x0040	9 (1-00-1)	3 (data R/W)	00
0x18	e000	0100	0x0040	9 (1-00-1)	3 (data R/W)	00
0x20	e100	0100	0x0040	b (1-01-1)	3 (data R/W)	00
0x28	e200	0100	0x0040	d (1-10-1)	3 (data R/W)	00
0x30	e300	0100	0x0040	f (1-11-1)	3 (data R/W)	00
0x38	c000	0100	0x0040	9 (1-00-1)	2 (data R/W)	00
0x40	c100	0100	0x0040	b (1-01-1)	2 (data R/W)	00
0x48	c200	0100	0x0040	d (1-10-1)	2 (data R/W)	00

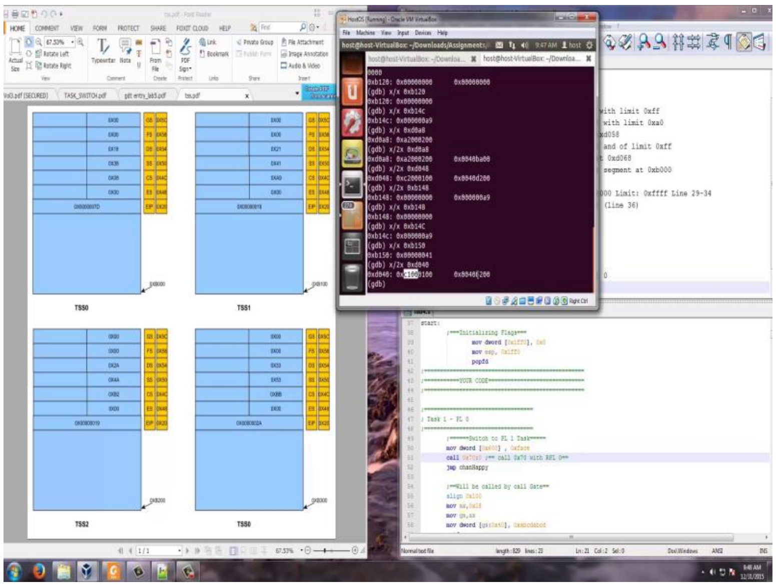
GDT(base address 0xd000)	Lower 32 bits		Upper 32 bits			
GDT Entry	Base (31:16)	Limit (15:0)	Others (31:16)	P-DPL-S (15:12)	Type (11-8)	Base (7:0)
0x50	c300	0100	0x0040	f (1-11-1)	2 (data R/W)	00
0x68	b000	0070	0x0040	8 (1-00-0)	b (32-Bit TSS (Busy))	00
0x70	b100	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS (Available))	00
0x78	b200	0070	0x0040	a (1-01-0)	9 (32-Bit TSS (Available))	00
0x80	b300	0070	0x0040	c (1-10-0)	9 (32-Bit TSS (Available))	00
0x88	b400	0070	0x0040	8 (1-00-0)	9 (32-Bit TSS (Available))	00
0xa8	a200	0200	0x0040	b (1-01-1)	a (code R/E)	00
0xb0	a400	0200	0x0040	d (1-10-1)	a (code R/E)	00
0xb8	a800	0200	0x0040	f (1-11-1)	a (code R/E)	00
0xe0	c700	0100	0x0040	b (1-01-1)	2 (data R/W)	00
0xe8	0008	0100	0x0040	e (1-11-0)	c (32-Bit Call Gate)	00

```
(gdb) x/2x 0xd20A
0xd20a: 0xd21000a0      0x33100000
(gdb) x/2x 0xd058
0xd058: 0xd10000ff      0x0000e200
(gdb) x/2x 0xd068
0xd068: 0xb0000070      0x00408900
(gdb) x/2x 0xd070
0xd070: 0xb1000070      0x00408900
(gdb) x/10x 0xb100
0xb100: 0x00000000      0x00000050      0x00000038      0x0000
0050
0xb110: 0x00000041      0x00000050      0x0000004a      0x0000
0000
0xb120: 0x00000000      0x00000000
(gdb) x/x 0xb120
0xb120: 0x00000000
(gdb) x/x 0xb14c
0xb14c: 0x000000a9
(gdb) x/x 0xd0a8
0xd0a8: 0xa2000200
(gdb) x/2x 0xd0a8
0xd0a8: 0xa2000200      0x0040ba00
(gdb)
```

```
start:
        ;***Initializing Flags***
                mov dword [0xfff0], 0x0
                mov esp, 0xfff0
                popfd
;*************************************************
;**********YOUR CODE******************************
;*************************************************

;*********************************
; Task 1 - PL 0
;*********************************
        ;******Switch to PL 1 Task*****
        mov dword [0x600] , 0xface
        call 0x70:0 ;** call 0x70 with RPL 0**
        jmp chanHappy

        ;**Will be called by call Gate**
        align 0x100
        mov ax,0x18
        mov gs,ax
        mov dword [gs:0x40], 0xabcdabcd
```

Let us just see here, where is a8? Please note that a8 starts at a200 size 200 and this is the code. So, I start executing at a200 with than offset of 0000, please note that e i p to start with 0000, I start executing a200 with privilege one using a privilege one code segment. So, my current privilege level actually becomes 1.

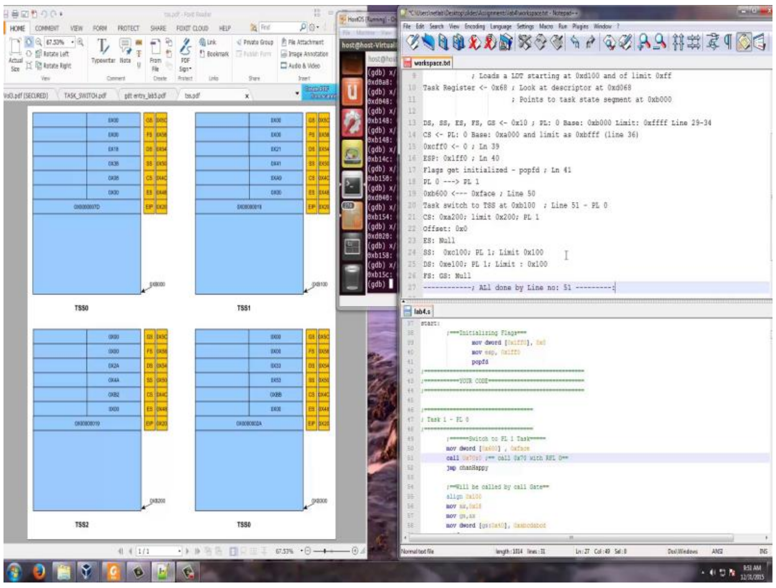
(Refer Slide Time: 06:52)



```
0000
0xb120: 0x00000000      0x00000000
(gdb) x/x 0xb120
0xb120: 0x00000000
(gdb) x/x 0xb14c
0xb14c: 0x000000a9
(gdb) x/x 0xd0a8
0xd0a8: 0xa2000200
(gdb) x/2x 0xd0a8
0xd0a8: 0xa2000200      0x0040ba00
(gdb) x/2x 0xd048
0xd048: 0xc2000100      0x0040d200
(gdb) x/2x 0xb148
0xb148: 0x00000000      0x000000a9
(gdb) x/x 0xb148
0xb148: 0x00000000
(gdb) x/x 0xb14C
0xb14c: 0x000000a9
(gdb) x/x 0xb150
0xb150: 0x00000041
(gdb) x/2x 0xd040
0xd040: 0xc1000100      0x0040b200
(gdb)
```

Now, we should also see, what is loaded at the remaining things. For example, what is loaded at ex, let us go and see b0b148 is a null, this a9b148 is again b148 is a null descriptor and then. So, I need only this b148 is a null descriptor. Then b14c is a 9 that is csb150 is 41. Let us go and see s slash 2x0xd040, it is a segment. So, the stack actually starts at the c100 and base 100 and is again of privilege level one.

(Refer Slide Time: 08:11)

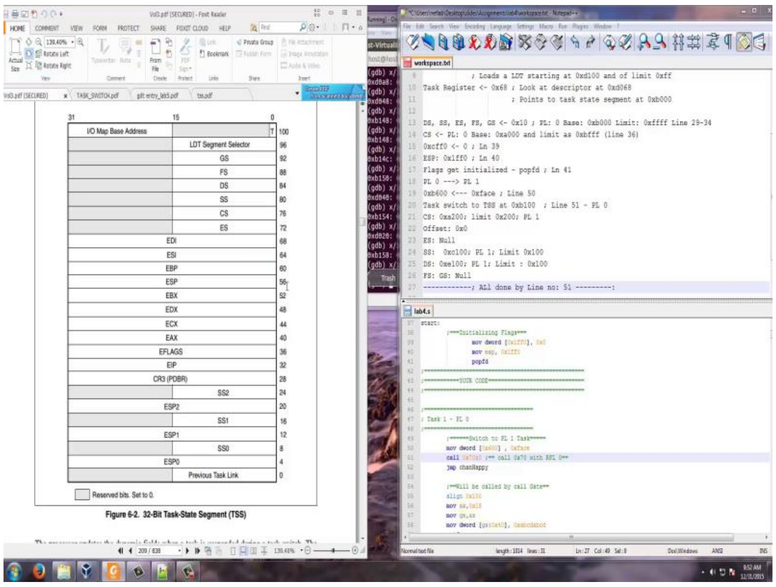


```
                ; Loads a LDT starting at 0xd100 and of limit 0xff
Task Register <- 0x68 ; Look at descriptor at 0xd068
                        ; Points to task state segment at 0xb000

DS, SS, ES, FS, GS <- 0x10 ; PL: 0 Base: 0xb000 Limit: 0xffff Line 29-34
CS <- PL: 0 Base: 0xa000 and limit as 0xbfff (line 36)
0xcff0 <- 0 ; Ln 39
ESP: 0x1ff0 ; Ln 40
Flags get initialized - popfd ; Ln 41
PL 0 ---> PL 1
0xb600 <--- 0xface ; Line 50
Task switch to TSS at 0xb100  ; Line 51 - PL 0
CS: 0xa200; limit 0x200; PL 1
Offset: 0x0
ES: Null
SS:  0xc100; PL 1; Limit 0x100
DS: 0xe100; PL 1; Limit : 0x100
FS: GS: Null
------------; ALl done by Line no: 51 ---------;
```

```
start:
        ;***Initializing Flags***
                mov dword [0x1ff0], 0x0
                mov esp, 0x1ff0
                popfd
;*********************************************
;***********YOUR CODE*************************
;*********************************************

;********************************
; Task 1 - PL 0
;********************************
        ;******Switch to PL 1 Task*****
        mov dword [0xb600] , 0xface
        call 0x70:0 ;** call 0x70 with RPL 0**
        jmp chanHappy

        ;**Will be called by call Gate**
        align 0x100
        mov ax,0x18
        mov gs,ax
        mov dword [gs:0x40], 0xabcdabcd
```

So, here let us go and see, I have code segment starting at 200 limits in hexadecimal. My starting offset is 0, my es is null descriptor, my ss is the segment, c100 is code segment is privilege one, ss is starting at c100 and PL 1, its limit is 100. Now, let us see what is ds, b154 is ds and that is 21. So, let us see what is there in s slash x20. This is again a privilege length segment starting at e100. So, ds starting at 0xe100 and privilege one and the limit is also 100. This is what happens to ds and let us see what is in fs, which is just 58 and 5c, this gs is also 0. So, fs and gs are null, these are d1 by your single task switching. These all things are d1 by line number 51 and what is your ESP value?

(Refer Slide Time: 11:03)

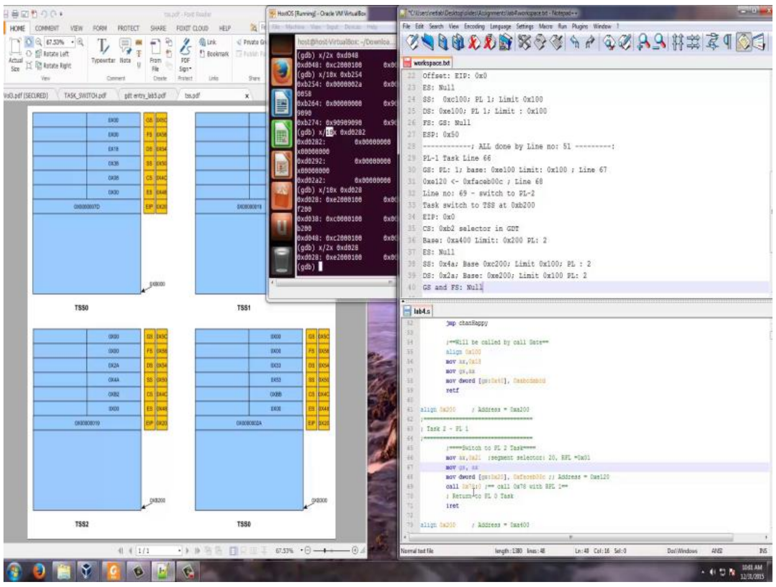


```
        ; Loads a LDT starting at 0xd100 and of limit 0xff
Task Register <- 0x68 ; Look at descriptor at 0xd068
                ; Points to task state segment at 0xb000

DS, SS, ES, FS, GS <- 0x10 ; PL: 0 Base: 0xb000 Limit: 0xffff Line 29-34
CS <- PL: 0 Base: 0xa000 and limit as 0xbfff (line 36)
0xcff0 <- 0 ; Ln 39
ESP: 0x1ff0 ; Ln 40
Flags get initialized - popfd ; Ln 41
PL 0 ---> PL 1
0xb600 <--- 0xface ; Line 50
Task switch to TSS at 0xb100  ; Line 51 - PL 0
CS: 0xa200; limit 0x200; PL 1
Offset: 0x0
ES: Null
SS:  0xc100; PL 1; Limit 0x100
DS: 0xe100; PL 1; Limit : 0x100
FS: GS: Null
------------; ALl done by Line no: 51 ---------:
```

```
start:
        ;***Initializing Flags***
                mov dword [0x1ff0], 0x0
                mov esp, 0x1ff0
                popfd
;************************************************
;**********YOUR CODE*****************************
;************************************************
```

So, let us go and see where ESP is? ESP is at 56 in this 30 30 8 0x38, your ESP value is s slash x0xb138, which ESP is 0850. So, that is where it will start, your stack segment is a currently initialized to c100. So, c 150 is the point, where the stack will start. So, that is where, d1 there to start with. Now, the task switch will happen.

(Refer Slide Time: 12:00)

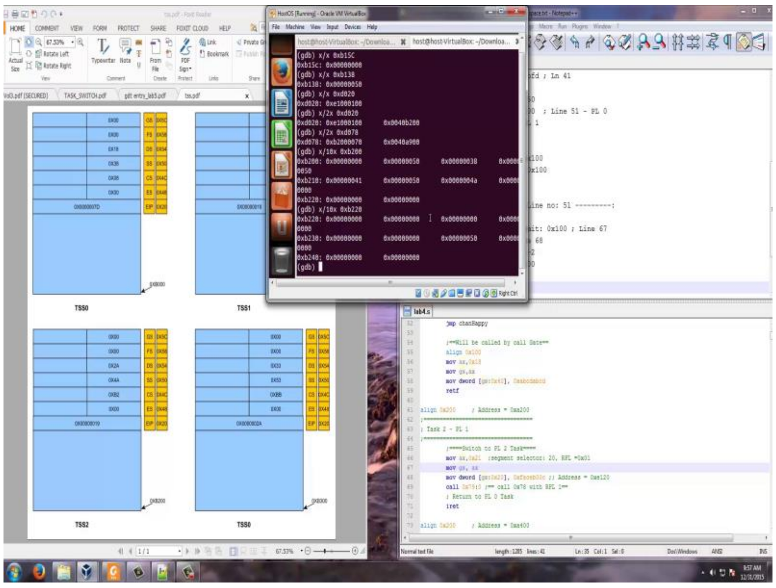


```
Offset: EIP: 0x0
ES: Null
SS:  0xc100; PL 1; Limit 0x100
DS: 0xe100; PL 1; Limit : 0x100
FS: GS: Null
ESP: 0x50
------------; ALL done by Line no: 51 ---------:
PL-1 Task Line 66
GS: PL: 1; base: 0xe100 Limit: 0x100 ; Line 67
0xe120 <- 0xfaceb00c ; Line 68
Line no: 69 - switch to PL-2
Task switch to TSS at 0xb200
EIP: 0x0
CS: 0xb2 selector in GDT
Base: 0xa400 Limit: 0x200 PL: 2
ES: Null
SS: 0x4a; Base 0xc200; Limit 0x100; PL : 2
DS: 0x2a; Base: 0xe200; Limit 0x100 PL: 2
GS and FS: Null
```

```
        jmp chanBappy

        ;**Will be called by call Gate**
        align 0x100
        mov ax,0x18
        mov gs,ax
        mov dword [gs:0x40], 0xabcdabcd
        retf
```

Once, this task switch that is the call 0x70 happens, it should now start executing at a200 because the offset this is basically the e i p set as 0. So, it should start executing at a200. Let us go and see where a200 is? So, I do and others point align 200. So, it comes to a200, it starts executing this. At this point what I am doing, this is PL1 task, this is line 66. So, I move ax comma 21 and what is ax comma 21.

(Refer Slide Time: 13:01)

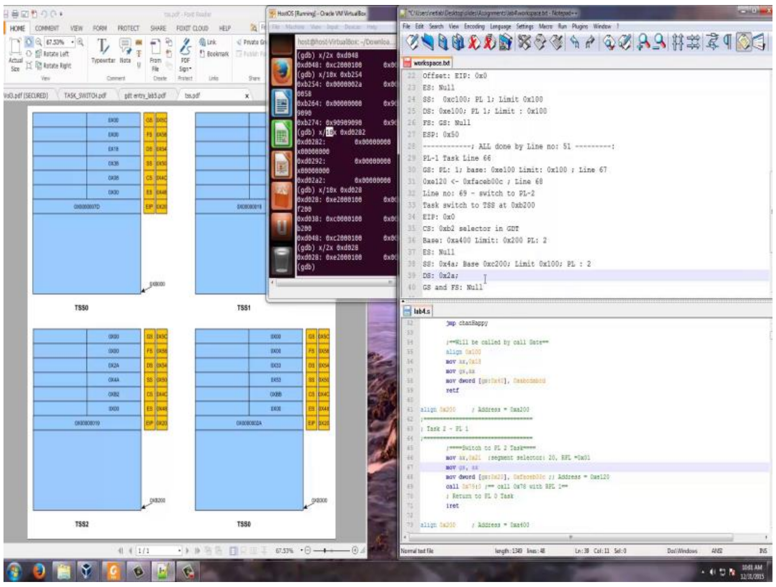


```
(gdb) x/x 0xb15C
0xb15c: 0x00000000
(gdb) x/x 0xb138
0xb138: 0x00000050
(gdb) x/x 0xd020
0xd020: 0xe1000100
(gdb) x/2x 0xd020
0xd020: 0xe1000100      0x0040b200
(gdb) x/2x 0xd078
0xd078: 0xb2000078      0x0040a900
(gdb) x/10x 0xb200
0xb200: 0x00000000      0x00000050      0x00000038      0x0000
0050
0xb210: 0x00000041      0x00000058      0x0000004a      0x0000
0000
0xb220: 0x00000000      0x00000000
(gdb) x/10x 0xb220
0xb220: 0x00000000      0x00000000      0x00000000      0x0000
0000
0xb230: 0x00000000      0x00000000      0x00000058      0x0000
0000
0xb240: 0x00000000      0x00000000
(gdb)
```

```
52          jmp chanHappy
53
54          ;==Will be called by call Gate==
55          align 0x100
56          mov ax,0x18
57          mov gs,ax
58          mov dword [gs:0x40], 0xabcdabcd
59          retf
60
61  align 0x200       ; Address = 0xa200
62  ;===============================
63  ; Task 2 - PL 1
64  ;===============================
65          ;====Switch to PL 2 Task====
66          mov ax,0x21   ;segment selector: 20, RPL =0x01
67          mov gs, ax
68          mov dword [gs:0x20], 0xfaceb00c ;; Address = 0xe120
69          call 0x79:0 ;== call 0x78 with RPL 1==
70          ; Return to PL 0 Task
71          iret
72
73  align 0x200       ; Address = 0xa400
```

Let us go to the GDB and find out 0xd020, I moving to the gs e000100 to gs PL 1 segment. This is what I am trying to move, please note that this is a PL 1 data segment, type 2 starting at e000100 with base 100, PL 1 base e000100 limit 100. This is what I do at line 57. This is your 20th descriptor 0x20. Please note e000100, base is 100, on the left hand side and is a data read write. Now, I move gs colon 0x20 face, I moving to address e120 basically gets 0x face. This is line 68, just to show that we are coming into these routines and I am able to load, before gs was nothing, now gs has become this now. I do a call line number 69. I do a switch two, privilege two by doing a call. So, this call is looking at 0x79, what is 0x79? That is 0878 s slash 2x0xb078, this is another tss segment 40a9 in privilege one and starting at b200 and 0070. So, this is cis to PL 2 again. This is task switch into tss at 0xb200. So, let us go and see what is there in b200. So, s slash 10x0xb200 and there you see that I should first go and see where is e i p s slash b220.

(Refer Slide Time: 16:33)



```
(gdb) x/2x 0xd048
0xd048: 0xc2000100      0x0
(gdb) x/10x 0xb254
0xb254: 0x0000002a      0x0
0058
0xb264: 0x00000000      0x9
9090
0xb274: 0x90909090      0x9
(gdb) x/10x 0xd0282
0xd0282:        0x00000000
x00000000
0xd0292:        0x00000000
x00000000
0xd02a2:        0x00000000
(gdb) x/10x 0xd028
0xd028: 0xe2000100      0x0
f200
0xd038: 0xc0000100      0x0
b200
0xd048: 0xc2000100      0x0
(gdb) x/2x 0xd028
0xd028: 0xe2000100      0x0
(gdb)
```

```
Offset: EIP: 0x0
ES: Null
SS:  0xc100; PL 1; Limit 0x100
DS: 0xe100; PL 1; Limit : 0x100
FS: GS: Null
ESP: 0x50
------------; ALL done by Line no: 51 ---------:
PL-1 Task Line 66
GS: PL: 1; base: 0xe100 Limit: 0x100 ; Line 67
0xe120 <- 0xfaceb00c ; Line 68
Line no: 69 - switch to PL-2
Task switch to TSS at 0xb200
EIP: 0x0
CS: 0xb2 selector in GDT
Base: 0xa400 Limit: 0x200 PL: 2
ES: Null
SS: 0x4a; Base 0xc200; Limit 0x100; PL : 2
DS: 0x2a;
GS and FS: Null
```

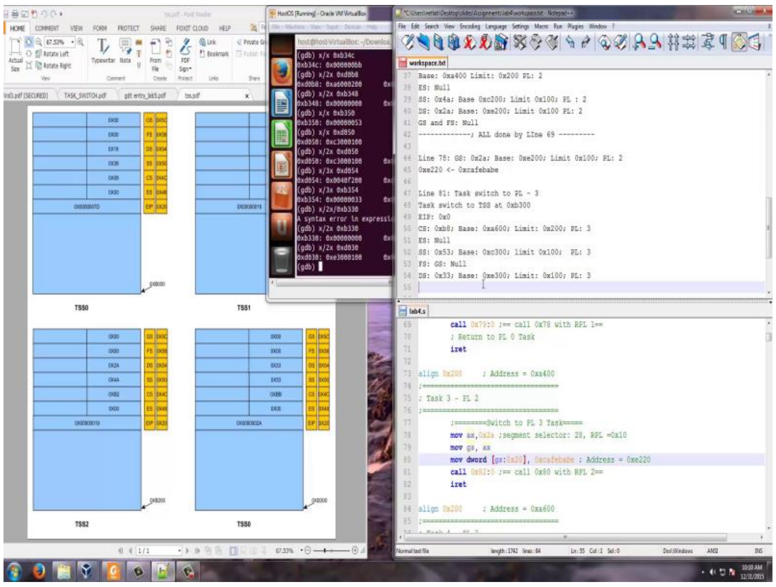
```
52          jmp chanHappy
53
54          ;**Will be called by call Gate**
55          align 0x100
56          mov ax,0x18
57          mov gs,ax
58          mov dword [gs:0x40], 0xabcdabcd
59          retf
60
61  align 0x200     ; Address = 0xa200
62  ;**********************************
63  ; Task 2 - PL 1
64  ;**********************************
65          ;****Switch to PL 2 Task*****
66          mov ax,0x21  ;segment selector: 20, RPL =0x01
67          mov gs, ax
68          mov dword [gs:0x20], 0xfaceb00c ;; Address = 0xe120
69          call 0x79:0 ;** call 0x78 with RPL 1**
70          ; Return to PL 0 Task
71          iret
72
73  align 0x200     ; Address = 0xa400
```

So, e i p is 000. So, your e i p offset is 0x0 and we need to see what is the code segment? The code segment is b24c. Of the code segment b2, the cs go to segment descriptor 0xb2 it is nothing, but 0xb0. So, sx slash 2x0xd0b0 is the segment selector. This is the code segment that starts at 0xa400 and of size 200. Please note that it is privilege level is two. So, 0xb2 selector in GDT which is nothing but base is 0xa400 and limit is 0x200 and privilege level is two and note that your es segment is null because your s slash xb248 is 00.

Let us see what is b250? It points to the segment 4a. So, ss point to the segment 4a, 4a is 48 actually privilege two. So, f slash 2xp048, this is a stack starting at c000200 and this is also a privilege two. This is base 0xc200 and 0x100 and limit is 0x100 and privilege level is two. Now, I have to see for the remaining things. So, s slash some 10x, I can do 0xd0d0 b2 b2500 b250. We have seen b254, there you see that ds is pointing to 2a and your fs and gs are null. Your ds is pointing to 2a and your gs and fs are actually, what is 2a? That is nothing but the GDT and sees what this d028? Actually, a will be 8 this is only in the GDT. So, this is a data segments starting at e000200 with base as 2e000200 and limit as 100 and p is 2. So, all these things happen because of this particular call 0x79 colon 0. This is happening at line number 69. So this is all d1 by line number 69.

Let us just try now, this should start now. So, this call should start now executing at a000 400 and it should execute at PL 2 because you are now cis to PL 2.

(Refer Slide Time: 21:10)



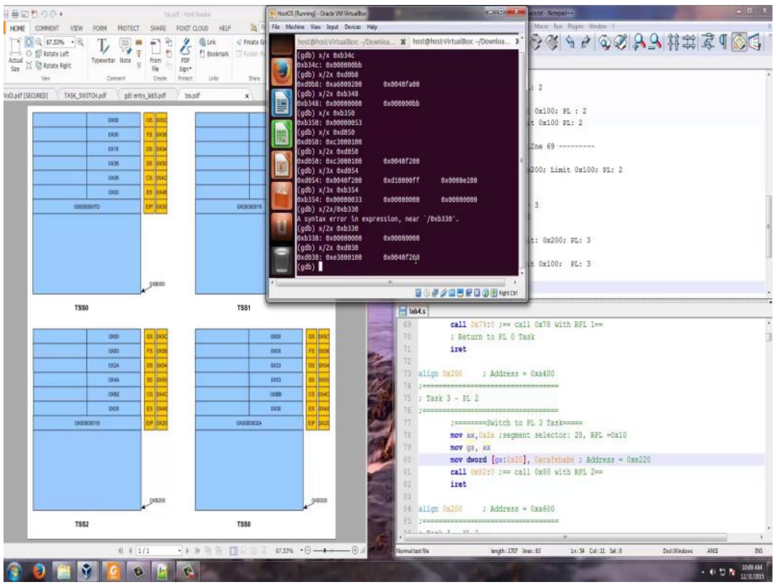
```
(gdb) x/x 0xb34c
0xb34c: 0x000000bb
(gdb) x/2x 0xd0b8
0xd0b8: 0xa6000200      0x
(gdb) x/2x 0xb348
0xb348: 0x00000000      0x
(gdb) x/x 0xb350
0xb350: 0x00000053
(gdb) x/x 0xd050
0xd050: 0xc3000100
(gdb) x/2x 0xd050
0xd050: 0xc3000100      0x
(gdb) x/3x 0xd054
0xd054: 0x0040f200      0x
(gdb) x/3x 0xb354
0xb354: 0x00000033      0x
(gdb) x/2x/0xb330
A syntax error in expressi
(gdb) x/2x 0xb330
0xb330: 0x00000000      0x
(gdb) x/2x 0xd030
0xd030: 0xe3000100      0x
(gdb)
```

```
37 Base: 0xa400 Limit: 0x200 PL: 2
38 ES: Null
39 SS: 0x4a; Base 0xc200; Limit 0x100; PL : 2
40 DS: 0x2a; Base: 0xe200; Limit 0x100 PL: 2
41 GS and FS: Null
42 ------------; ALL done by LIne 69 ---------
43
44 Line 78: GS: 0x2a; Base: 0xe200; Limit 0x100; PL: 2
45 0xe220 <- 0xcafebabe
46
47 Line 81: Task switch to PL - 3
48 Task switch to TSS at 0xb300
49 EIP: 0x0
50 CS: 0xb8; Base: 0xa600; Limit: 0x200; PL: 3
51 ES: Null
52 SS: 0x53; Base: 0xc300; limit 0x100;  PL: 3
53 FS: GS: Null
54 DS: 0x33; Base: 0xe300; Limit: 0x100; PL: 3
55
```

```
69          call 0x79:0 ;== call 0x78 with RPL 1==
70          ; Return to PL 0 Task
71          iret
72
73 align 0x200      ; Address = 0xa400
74 ;==================================
75 ; Task 3 - PL 2
76 ;==================================
77          ;========Switch to PL 3 Task=====
78          mov ax,0x2a ;segment selector: 28, RPL =0x10
79          mov gs, ax
80          mov dword [gs:0x20], 0xcafebabe ; Address = 0xe220
81          call 0x82:0 ;== call 0x80 with RPL 2==
82          iret
83
84 align 0x200      ; Address = 0xa600
85 ;==================================
```

This is a000200, now this is a000400. So, your program will start executing at this part now. What we will do now, we move to line 78 and there we move ax to 8, again the same thing. So, your gs is loaded with 0x2a, which is this 28. So, your gs will get base as e000200 and e000200 limit as 0x100 and PL 2 that is what your line 78 does. Now I am moving d word gs colon 20. So, your 0xe220 should basically get 0xf. I will try and just go and mark some of the things that even 32, this is what your PL 2 does and now it goes and calls 0x82. So, in line number 81, I am doing task switch to PL 3. So, I am calling 0x82 call 0x82. So, what is 0x82, that is let us go and see that s slash 2x.

(Refer Slide Time: 24:09)



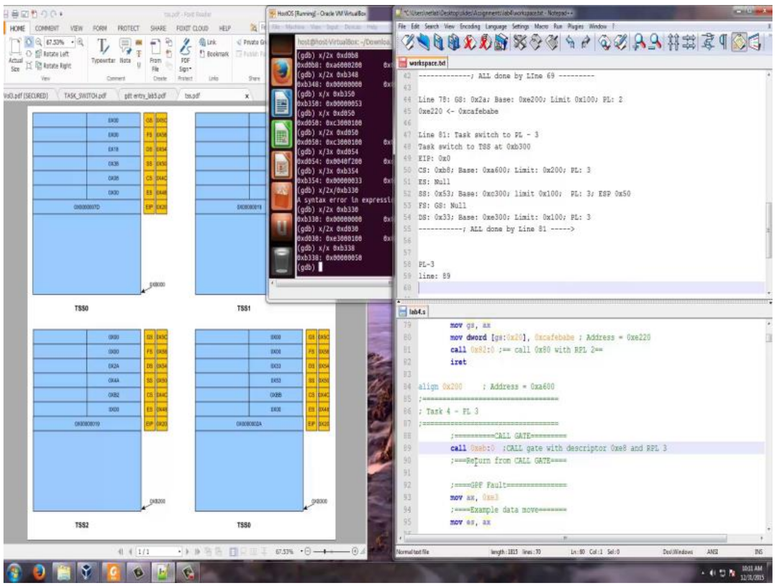
```
(gdb) x/x 0xb34c
0xb34c: 0x000000bb
(gdb) x/2x 0xd0b8
0xd0b8: 0xa6000200      0x0040fa00
(gdb) x/2x 0xb348
0xb348: 0x00000000      0x000000bb
(gdb) x/x 0xb350
0xb350: 0x00000053
(gdb) x/x 0xd050
0xd050: 0xc3000100
(gdb) x/2x 0xd050
0xd050: 0xc3000100      0x0040f200
(gdb) x/3x 0xd054
0xd054: 0x0040f200      0xd10000ff      0x0000e200
(gdb) x/3x 0xb354
0xb354: 0x00000033      0x00000000      0x00000000
(gdb) x/2x/0xb330
A syntax error in expression, near `/0xb330'.
(gdb) x/2x 0xb330
0xb330: 0x00000000      0x00000000
(gdb) x/2x 0xd030
0xd030: 0xe3000100      0x0040f200
(gdb) 
```

```
        call 0x79:0 ;== call 0x78 with RPL 1==
        ; Return to PL 0 Task
        iret

align 0x200     ; Address = 0xa400
;================================
; Task 3 - PL 2
;================================
        ;========Switch to PL 3 Task=====
        mov ax,0x2a ;segment selector: 28, RPL =0x10
        mov gs, ax
        mov dword [gs:0x20], 0xcafebabe ; Address = 0xe220
        call 0x82:0 ;== call 0x80 with RPL 2==
        iret

align 0x200     ; Address = 0xa600
;================================
```

Let us go and see what is there in d080 that is the segment. So, this is again a task stage segment starting at b000 300 and this is again a task switch to tss at 0xb100 and so let us go and see what is the e i p, there s slash x0xb32s b320. So, this is your e i p is 0. So, what is happening because of this task switch, your e i p is 0 and let us see what are all the other things there and like what is your cs? That will be b34, cacs uses a segment bb. So, cs now a segments bb is 0xb8 right, with privilege c. Let us see what is there in that segment in the GDT, GDT it starts at d000. So, d0b8, this is a code segment of privilege level three starting at a000600. This base is 0xa600 and limit is 0x200 and what else the privilege level is three. So, when I switch to this particular task I am now going to execute at privilege level three.

Now, let us go and see what is there in ab348 that is a null descriptor. So, your es is null now, where stack s slash xb350. This is a stack, it is pointing to 53, stack 53 means 50. So, what is the 50 segment there, s slash x0xdd000 is GDT stack, d050. So, this is a stack at c300, base is 0xc300, limit is 0x100 and what is happening, if you see that it is privilege level. If this is a privilege level three code. Now let us also see what is happening to other things. So, let us see the remaining three things b0b354. So, your code segment is a 30 your fs and gs are null. So, please note that your fs and gs both are null and your data segment is at 0x33, which is actually 0x30. So, let us go and find out, what is there in d030. So, this is a data segment starting at e000 300 base and this is privilege three so. Now, we will go and see that this base is 0xe000 300, limit is 0x100 and privilege is three and all this was d1 till line 81.

(Refer Slide Time: 29:29)



```
(gdb) x/2x 0xd0b8
0xd0b8: 0xa6000200
(gdb) x/2x 0xb348
0xb348: 0x00000000
(gdb) x/x 0xb350
0xb350: 0x00000053
(gdb) x/x 0xd050
0xd050: 0xc3000100
(gdb) x/2x 0xd050
0xd050: 0xc3000100
(gdb) x/3x 0xd054
0xd054: 0x0040f200
(gdb) x/3x 0xb354
0xb354: 0x00000033
(gdb) x/2x/0xb330
A syntax error in expressi
(gdb) x/2x 0xb330
0xb330: 0x00000000
(gdb) x/2x 0xd030
0xd030: 0xe3000100
(gdb) x/x 0xb338
0xb338: 0x00000050
(gdb)
```

```
-------------; ALL done by LIne 69 ---------

Line 78: GS: 0x2a; Base: 0xe200; Limit 0x100; PL: 2
0xe220 <- 0xcafebabe

Line 81: Task switch to PL - 3
Task switch to TSS at 0xb300
EIP: 0x0
CS: 0xb8; Base: 0xa600; Limit: 0x200; PL: 3
ES: Null
SS: 0x53; Base: 0xc300; limit 0x100;  PL: 3; ESP 0x50
FS: GS: Null
DS: 0x33; Base: 0xe300; Limit: 0x100; PL: 3
-----------; ALL done by Line 81 ----->


PL-3
line: 89
```

```
79          mov gs, ax
80          mov dword [gs:0x20], 0xcafebabe ; Address = 0xe220
81          call 0x82:0 ;== call 0x80 with RPL 2==
82          iret
83
84  align 0x200      ; Address = 0xa600
85  ;===================================
86  ; Task 4 - PL 3
87  ;===================================
88          ;==========CALL GATE=========
89          call 0xeb:0  ;CALL gate with descriptor 0xe8 and RPL 3
90          ;===Return from CALL GATE====
91
92          ;====GPF Fault===============
93          mov ax, 0xe3
94          ;====Example data move=======
95          mov es, ax
```

We should also go and find out what is the stack segment? ESP as you see here is 38. So, in the tss, b338 will give you the ESP s slash x0xb338 and that gives you. So, ESP is set to 50. Along with the stack segment, your ESP is also set to 50 that is where you are coming here and that is what you are in line 81 now. When I do this call, it should start executing at a000600 plus 0, it should start executing a000600. So, essentially this align will take to a000600 and so the first thing that I will be executing is the PL 3 task is I will be executing line 89.