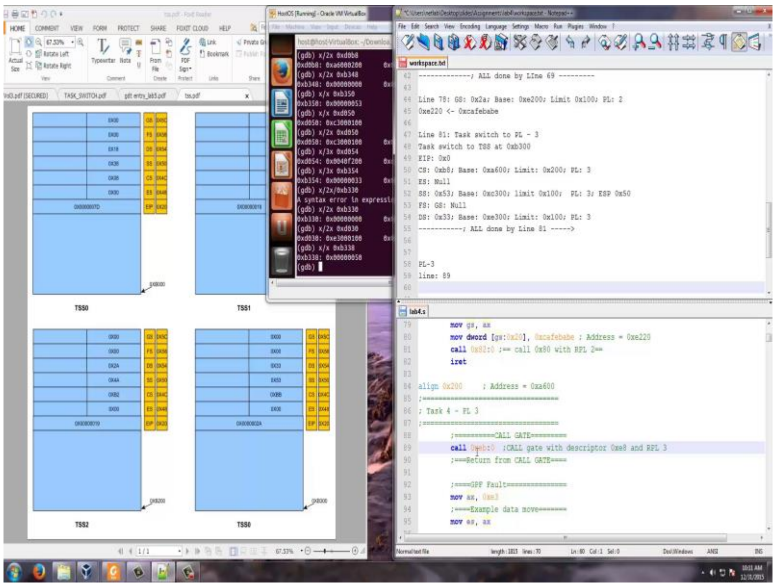
Information Security - II
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 33
Lab4 Part 3 - Week 6

(Refer Slide Time: 00:09)



```
(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)
```

workspace.txt

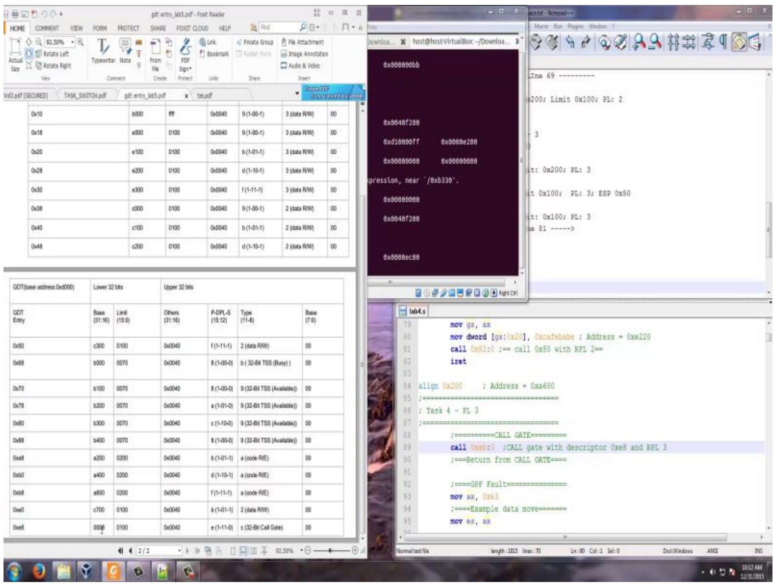
```
-------------; ALL done by LIne 69 ---------

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
```

lab4.s

```
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
```

What is line 89? Line 89 is again a call to 0xep. So, this is 0xep8, it is actually the descriptor so let us go and see what is there in x slash 2x0xd0e8 that is what you wanted. This is a call gate as you see here type this is a system descriptor call gate of privilege 3. This descriptor privilege is 3 and it points to a code segment which is 8 and offset there which is 100. So I need to go to the code segment pointing by 8 and start executing from an offset 100 from the code segment. And this is what (Refer Time: 01:18) so just get in this thing what is 0xe8. Actually a 32 bit call gate, this is not the base, this is actually the code segment selector 8 and then offset is 100.

(Refer Slide Time: 01:21)



0x10	b000	ffff	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	a600	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

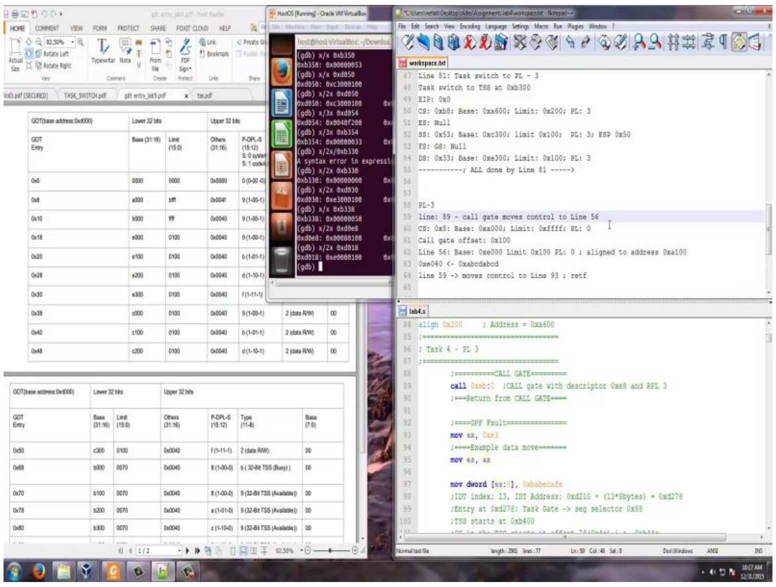
```
0x000000bb



0x0040f200

0xd10000ff      0x0000e200

0x00000000      0x00000000

pression, near `/0xb330'.

0x00000000

0x0040f200



0x0000ec00
```

```
ine 69 ---------

200; Limit 0x100; PL: 2


 3


t: 0x200; PL: 3

t 0x100;  PL: 3; ESP 0x50

t: 0x100; PL: 3
e 81 ----->
```

```
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
```

What is the code segment 8 here? Let us go and see the code segment 8 is actually pointing to a000 and I should start executing at a100 here. The call gate essentially says go and start executing at a100. What is a100? Please note that, this is the align 0x100 so this is 1100.

(Refer Slide Time: 02:29)



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) S: 0 system S: 1 code/		
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)	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

```
(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) x/x 0xb338
0xb338: 0x00000058
(gdb) x/2x 0xd0e8
0xd0e8: 0x00080100      0x
(gdb) x/2x 0xd018
0xd018: 0xe0000100      0x
(gdb) 
```

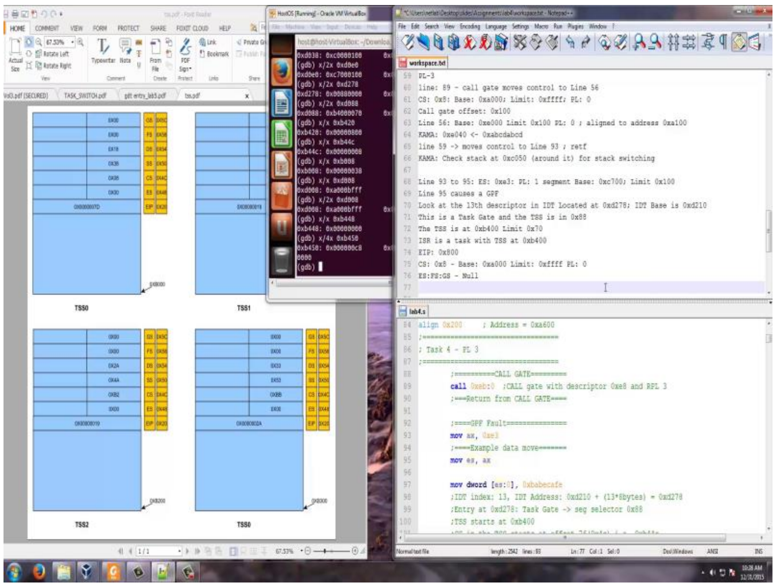
```
Line 81: Task switch to PL - 3
Task switch to TSS at 0xb300
EIP: 0x0
CS: 0xb8; Base: 0xa600; Limit: 0x200; PL: 3
ES: Null
SS: 0x53; Base: 0xc300; limit 0x100;  PL: 3; ESP 0x50
FS: GS: Null
DS: 0x33; Base: 0xe300; Limit: 0x100; PL: 3
-----------; ALL done by Line 81 ----->



PL-3
line: 89 - call gate moves control to Line 56
CS: 0x8: Base: 0xa000; Limit: 0xffff; PL: 0
Call gate offset: 0x100
Line 56: Base: 0xe000 Limit 0x100 PL: 0 ; aligned to address 0xa100
0xe040 <- 0xabcdabcd
line 59 -> moves control to Line 93 ; retf
```

```
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
96
97          mov dword [es:0], 0xbabecafe
98          ;IDT index: 13, IDT Address: 0xd210 + (13*8bytes) = 0xd278
99          ;Entry at 0xd278: Task Gate -> seg selector 0x88
100         ;TSS starts at 0xb400
```

So, when your call gate executes in line 89, it swifts control to line 56, call gate move control to line 56, it is calling this. And what you do here? Here you mov ax 0x18, so what is there in 0x18?. So line 56. What is there in slash 2x, you know it is d018 mov e000, so this is moving into GS base is 0xe000 and limit is 0x100 and you know this is 4092. So this is privilege 0. I am trying to do a privilege 0 activity here. I am in privilege level 3 and just by using call gate I am now executing a privilege level 0 code.

Actually here your CS is pointing to 8 which is base is 0xa000 and e limit is whatever f f and e l is 0, call gate offset is 0x100. So I start executing from a100 which is line 56, please note that this is align to address a100. Now from privilege 3 I have started executing a privilege 0 code that is what we are saying. Now what type we are doing is? In e040, so this GS colon, I am moving a b c d, a b c d. So, 0xe040 basically gets a b c d, a b c d. And then I do a far written because it is a far call. I do a far written, when I do a far written basically I come to, I go back so this is line number 59. Line 59 mov control to line number 93 (Refer Time: 06:03). So, at this point also we have been using the stack pointer 0 for doing this call.

(Refer Slide Time: 06:28)



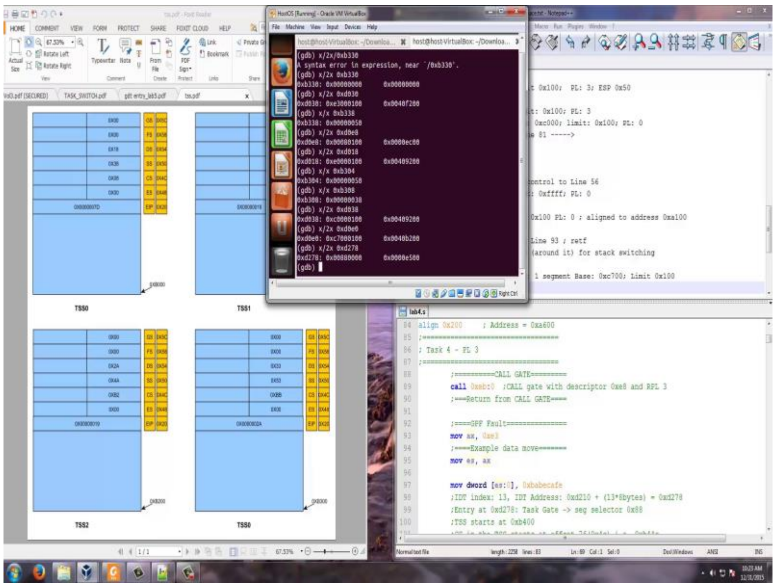
```
0xd038: 0xc0000100
(gdb) x/2x 0xd0e0
0xd0e0: 0xc7000100
(gdb) x/2x 0xd278
0xd278: 0x00880000
(gdb) x/2x 0xd088
0xd088: 0xb4000070
(gdb) x/x 0xb420
0xb420: 0x00000800
(gdb) x/x 0xb44c
0xb44c: 0x00000008
(gdb) x/x 0xb008
0xb008: 0x00000038
(gdb) x/x 0xd008
0xd008: 0xa000bfff
(gdb) x/2x 0xd008
0xd008: 0xa000bfff
(gdb) x/x 0xb448
0xb448: 0x00000000
(gdb) x/4x 0xb450
0xb450: 0x000000c8
0000
(gdb)
```

```
PL-3
line: 89 - call gate moves control to Line 56
CS: 0x8: Base: 0xa000; Limit: 0xffff; PL: 0
Call gate offset: 0x100
Line 56: Base: 0xe000 Limit 0x100 PL: 0 ; aligned to address 0xa100
KAMA: 0xe040 <- 0xabcdabcd
line 59 -> moves control to Line 93 ; retf
KAMA: Check stack at 0xc050 (around it) for stack switching

Line 93 to 95: ES: 0xe3: PL: 1 segment Base: 0xc700; Limit 0x100
Line 95 causes a GPF
Look at the 13th descriptor in IDT Located at 0xd278; IDT Base is 0xd210
This is a Task Gate and the TSS is in 0x88
The TSS is at 0xb400 Limit 0x70
ISR is a task with TSS at 0xb400
EIP: 0x800
CS: 0x8 - Base: 0xa000 Limit: 0xffff PL: 0
ES:FS:GS - Null
```

```
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
96
97          mov dword [es:0], 0xbabecafe
98          ;IDT index: 13, IDT Address: 0xd210 + (13*8bytes) = 0xd278
99          ;Entry at 0xd278: Task Gate -> seg selector 0x88
100         ;TSS starts at 0xb400
```

So when you are doing this call, when I am in a PL 3 segment I need a stack of PL 0. This is I am just trying to explain you the task switching. I need to have PL 0 and where will I find a PL 0. So let us go here, your ESP 0 and SS 0. At 4 and 8 I am going to find what is ESP 0 and SS 0, so let us just see this. At PL 3 your SS, ESP 0 is p304 is 50, ESP 0 is 0x50 and your b308 04. Your stack segment SS 0 is 0x38 and what is0x38?

(Refer Slide Time: 07:46)



```
(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) x/x 0xb338
0xb338: 0x00000050
(gdb) x/2x 0xd0e8
0xd0e8: 0x00080100      0x0000ec00
(gdb) x/2x 0xd018
0xd018: 0xe0000100      0x00409200
(gdb) x/x 0xb304
0xb304: 0x00000050
(gdb) x/x 0xb308
0xb308: 0x00000038
(gdb) x/2x 0xd038
0xd038: 0xc0000100      0x00409200
(gdb) x/2x 0xd0e0
0xd0e0: 0xc7000100      0x0040b200
(gdb) x/2x 0xd278
0xd278: 0x00880000      0x0000e500
(gdb)
```

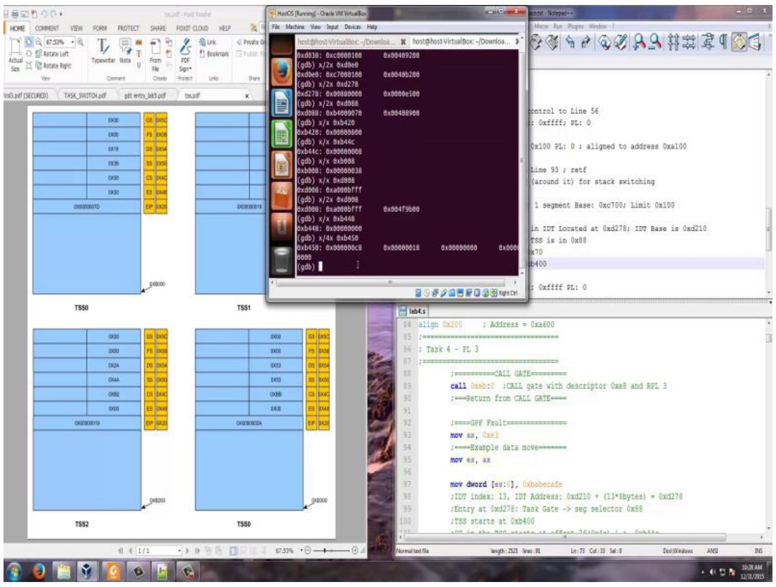
```
t 0x100;  PL: 3; ESP 0x50

t: 0x100; PL: 3
 0xc000; limit: 0x100; PL: 0
s 81 ----->

ontrol to Line 56
: 0xffff; PL: 0

0x100 PL: 0 ; aligned to address 0xa100

Line 93 ; retf
(around it) for stack switching

 1 segment Base: 0xc700; Limit 0x100
```

```
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
96
97          mov dword [es:0], 0xbabecafe
98          ;IDT index: 13, IDT Address: 0xd210 + (13*8bytes) = 0xd278
99          ;Entry at 0xd278: Task Gate -> seg selector 0x88
100         ;TSS starts at 0xb400
```

Again let us go back to the GDT 0xd038 and c000 base value. This is the stack with base as 0xc000 limit 0x100 and here 0. So when I do this call gate all that I need for written etcetera would have essentially been done on this a particular stack. Now, I can check stack at 0xc000 right c050, 0c050 actually around it for stack switching, because I am going from a PL 3 code to a PL 0 code. Similarly, also please note that the current stack for this it is c300 and this stack is at c050 for the current stag of the PL 3 processes is at c300 and your stack is at c050. It is not using the stack, it is trying to use the stack we will check this also. I will just put from now on some comma here which is essentially means I have to check these things, yes this is to prove that I have executed.

Coming back I will be now trying to execute line 93, I am back at line 93. And in line 93 I am moving line 93 to 95 I am moving into EX, ES the thing 0x is e3 0xe3 is e0. So let us see what is S slash 2x0xd0e0? Please note that this is a privilege 1 segment with c700 as base and 100. There is a PL 1 segment with base as 0xc700 and limit as 0x100. This obviously creates the general protection fault. What is a general protection fault? It is number is 13. Now 13 means, it should be stored in 13 into 8 which is 104 which in hexadecimal is 68. Where is my IDT? IDT is at d210. So I will go and find out x slash 2x0xd210 plus 68 is d278. This says that this is 0088, so this is the 0x88th descriptor and it is e5, if you see the type it is a system. This is a task switch. Again this trying to do a task switch using a task state segment descriptor, which is this a task gate. So, essentially it is trying to do a task switch and using a descriptor at 088. Let us just summarize it here, so line 95 causes a GPF, so look at d 13th descriptor in IDT located at xd278. IDT base derives d210. This shows this is a task gate and the TSS is in 0x. Let us see where is x slash 2x0xd088 and this tells me this is a DSS the TSS is at 0xb400 and limit is 0x, and note that this has a 32 bit descriptor. Now what happens here, I go and see a b400 is the task gate, now I have to go and see what is there in b400.

(Refer Slide Time: 14:21)



```
0xd038: 0xc0000100      0x00409200
(gdb) x/2x 0xd0e0
0xd0e0: 0xc7000100      0x0040b200
(gdb) x/2x 0xd278
0xd278: 0x00880000      0x0000e500
(gdb) x/2x 0xd088
0xd088: 0xb4000070      0x00408900
(gdb) x/x 0xb420
0xb420: 0x00000800
(gdb) x/x 0xb44c
0xb44c: 0x00000008
(gdb) x/x 0xb008
0xb008: 0x00000038
(gdb) x/x 0xd008
0xd008: 0xa000bfff
(gdb) x/2x 0xd008
0xd008: 0xa000bfff      0x004f9b00
(gdb) x/x 0xb448
0xb448: 0x00000000
(gdb) x/4x 0xb450
0xb450: 0x000000c8      0x00000018      0x00000000      0x0000
0000
(gdb)
```

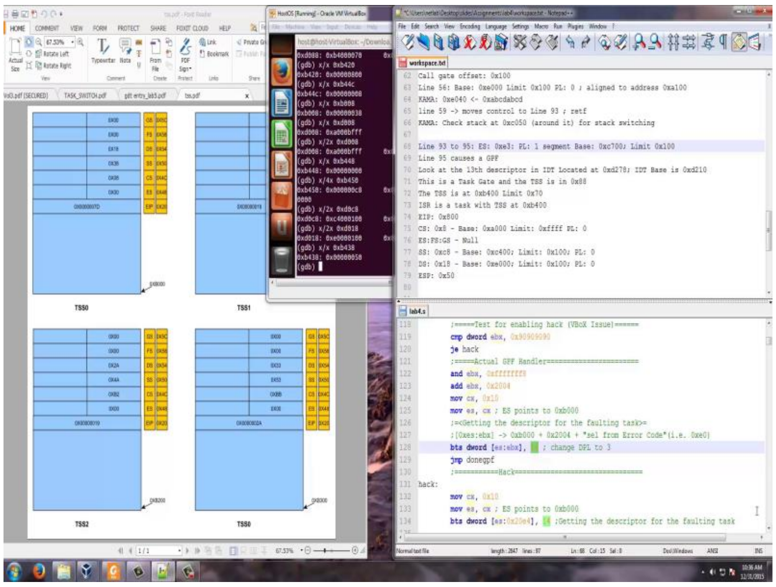
```
control to Line 56
: 0xffff; PL: 0

0x100 PL: 0 ; aligned to address 0xa100

Line 93 ; retf
(around it) for stack switching

 1 segment Base: 0xc700; Limit 0x100

in IDT Located at 0xd278; IDT Base is 0xd210
TSS is in 0x88
x70
b400

: 0xffff PL: 0
```

```
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
96
97          mov dword [es:0], 0xbabecafe
98          ;IDT index: 13, IDT Address: 0xd210 + (13*8bytes) = 0xd278
99          ;Entry at 0xd278: Task Gate -> seg selector 0x88
100         ;TSS starts at 0xb400
```

We have to see what is there in b400, so let us go and see. The same thing that we could see here, I should say that the interrupt service between isr is a task with PSS at 0xb400. So let us first go and see where it will start executing, where it will be as x s slash x0xv420 is where it should start executing. It starts executing at 800. Now, we will see e i p is 0x800 and your CS which is b44c should give you a c s s slash b44c and it is 8. What is 8 here? So, CS is again 0x8 so this is nothing but your base is 0xa000 limit is f f f f andPL is 0. How find out? Just go to x slash x0xd0x sorry s slash x0xd008. This is a000 and limit is quit (Refer Time: 16:26) fpfff and this is a PL level. Now the intreup service routine will start executing at a800. And what is the stack x? Let us see the other segments s slash x b448 this is your ES, ES is null, FS and GS are null. So, we just see ES, FF and GS are all null. Let us also see what is there in SS. SS is b450 which is c8. Your SS is 0xc8 and let us see, what does that mean. That is again d0c8. This c400 is a stack is start at base is c400 and limit is 100 and this is a privilege 0 stack. Your DS is also very important. Your DS is 18, DS is 0x18.

What is that base for this? S slash 2x0xd018 it is e000 and base is 0xe000, limit is 100 and privilege level is 0. So, now the task state thing will start working. It should start executing at a800 that is the exception. You have seen several aliens here, so you start with a000 and you put an align 100 here this is a100, you put align 200 it is a200 then another 200 a400 another 200 a600 and another 200 say a800. So your code will start executing here. I am popping EBX from the stack which stack you should be the 0, stack that is c400. Before going to that let us also find out what is the ESP value, ESP is 0x38 x slash x0xb438. The ESP is set to 50. Now here we have some small issue that we had found out. There are different V boxes that we are using. The V box is actually assume letting the (Refer Time: 20:33) machine. Some versions of the V box specifically the later versions beyond V box 5 and above, they see the moments such type of an exception comes. Typically a general protection fault there is a error code that is entered in to the stack.

(Refer Slide Time: 20:57)



```
0xd008: 0xb4000070
(gdb) x/x 0xb420
0xb420: 0x00000800
(gdb) x/x 0xb44c
0xb44c: 0x00000000
(gdb) x/x 0xb008
0xb008: 0x00000038
(gdb) x/x 0xd008
0xd008: 0xa000bfff
(gdb) x/2x 0xd008
0xd008: 0xa000bfff
(gdb) x/x 0xb448
0xb448: 0x00000000
(gdb) x/4x 0xb450
0xb450: 0x000000c8
0000
(gdb) x/2x 0xd0c8
0xd0c8: 0xc4000100
(gdb) x/2x 0xd018
0xd018: 0xe0000100
(gdb) x/x 0xb438
0xb438: 0x00000050
(gdb)
```

```
62 Call gate offset: 0x100
63 Line 56: Base: 0xe000 Limit 0x100 PL: 0 ; aligned to address 0xa100
64 KAMA: 0xe040 <- 0xabcdabcd
65 line 59 -> moves control to Line 93 ; retf
66 KAMA: Check stack at 0xc050 (around it) for stack switching
67
68 Line 93 to 95: ES: 0xe3: PL: 1 segment Base: 0xc700; Limit 0x100
69 Line 95 causes a GPF
70 Look at the 13th descriptor in IDT Located at 0xd278; IDT Base is 0xd210
71 This is a Task Gate and the TSS is in 0x88
72 The TSS is at 0xb400 Limit 0x70
73 ISR is a task with TSS at 0xb400
74 EIP: 0x800
75 CS: 0x8 - Base: 0xa000 Limit: 0xffff PL: 0
76 ES:FS:GS - Null
77 SS: 0xc8 - Base: 0xc400; Limit: 0x100; PL: 0
78 DS: 0x18 - Base: 0xe000; Limit: 0x100; PL: 0
79 ESP: 0x50
80
```

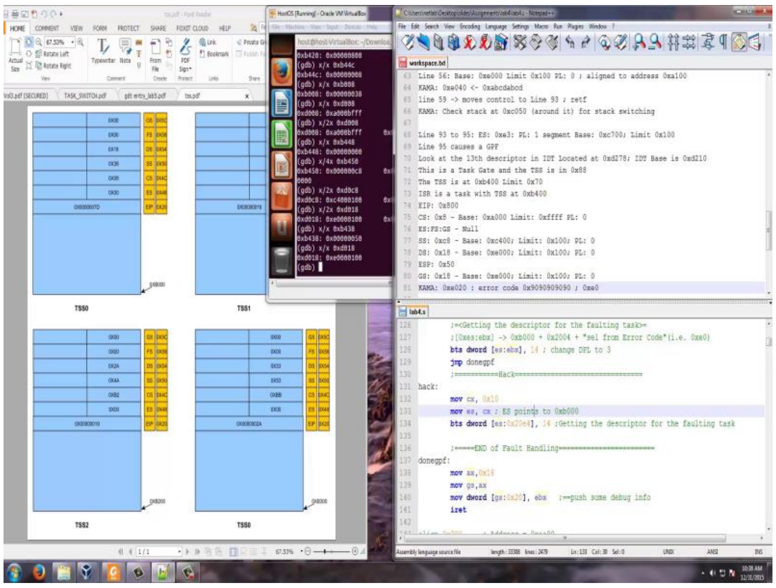
```
118         ;=====Test for enabling hack (VBoX Issue)======
119         cmp dword ebx, 0x90909090
120         je hack
121         ;=====Actual GPF Handler======================
122         and ebx, 0xfffffff8
123         add ebx, 0x2004
124         mov cx, 0x10
125         mov es, cx ; ES points to 0xb000
126         ;=<Getting the descriptor for the faulting task>=
127         ;[0xes:ebx] -> 0xb000 + 0x2004 + "sel from Error Code"(i.e. 0xe0)
128         bts dword [es:ebx], 13 ; change DPL to 3
129         jmp donegpf
130         ;==========Hack================================
131 hack:
132         mov cx, 0x10
133         mov es, cx ; ES points to 0xb000
134         bts dword [es:0x20e4], 14 ;Getting the descriptor for the faulting task
```

In this case, the 0 stack which is currently pointing to c400, ESP is 50 c450 I need to have the error code. But normally sometimes what is happening in the latest virtual boxes especially b10, somehow this error code is not going in to the stack. If you use a (Refer Slide Time: 21:16) previous version error code goes there. Initially we have made it null, initially the value will be 1990, 0x909090. If the error code is pushed in to the stack by the v box then this comparisons statement that 19 will be true because if i (Refer Time: 21:39). If I see, if I pop from the stag pop bbx that I am doing at 117 and I find that it is null that means, no error code has been pushed by the v bags I go and jmp to this hack, otherwise what I do, and EBX with.

What will be the error code there? The error code there would be actually the segment descriptor which is basically the segment selector which actually created the general protection fault. I know that I will get what will be in EBX here would be the index of the descriptor that actually created the fault. In this case e3 which is basically e0, e0 is the one which created the fault, so I will have e0 here. And why I got that fault e0? Because I am trying to use a privilege 1 segment it should be a privilege 3 segment. So what will be there in EBX? After this end operation will be e0, if the error code has been pushed in to the stack. Now what I do I add 2004 to the CBX so 2 to the c3. Basically, I get this second byte of the descriptor. Now, please note that when I am doing this I am moving in to ES the value is 0x10. Now you see that ES actually points to b000. What is there in EBX? It is 20e7. So, what is ES colon EBX? It will be d027, sorry d0e7.

D0e7 is nothing but the second 4 bytes of e3. In this if I am going to change the 14 bit to 1, essentially what I am doing? I am making that privilege level of that segment to 3. Currently, the privilege level of e3 as you see on the top is 1, by making that 14 bit of the second 4 bytes of that descriptor 1 what I am converting it I am making it as PL 3.

(Refer Slide Time: 24:57)



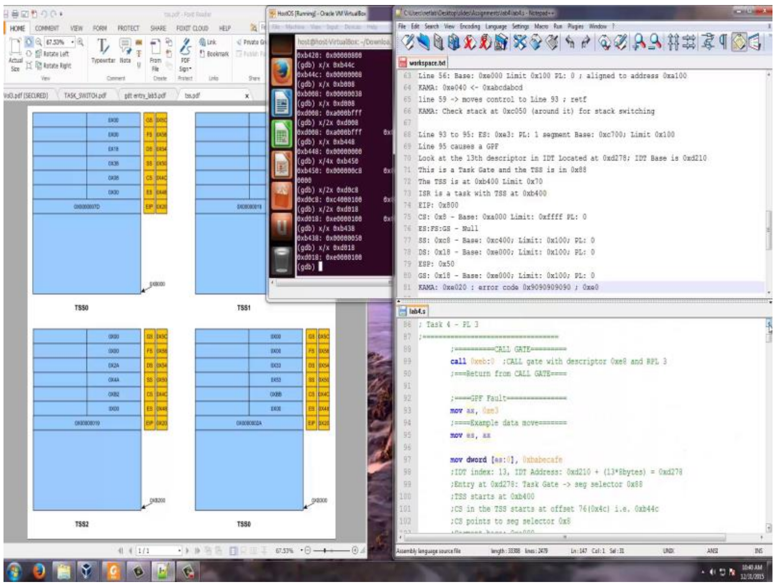
```
0xb420: 0x00000000
(gdb) x/x 0xb44c
0xb44c: 0x00000000
(gdb) x/x 0xb008
0xb008: 0x00000038
(gdb) x/x 0xd008
0xd008: 0xa000bfff
(gdb) x/2x 0xd008
0xd008: 0xa000bfff
(gdb) x/x 0xb448
0xb448: 0x00000000
(gdb) x/4x 0xb450
0xb450: 0x000000c8
0000
(gdb) x/2x 0xd0c8
0xd0c8: 0xc4000100
(gdb) x/2x 0xd018
0xd018: 0xe0000100
(gdb) x/x 0xb438
0xb438: 0x00000050
(gdb) x/x 0xd018
0xd018: 0xe0000100
(gdb)
```

```
Line 56: Base: 0xe000 Limit 0x100 PL: 0 ; aligned to address 0xa100
KAMA: 0xe040 <- 0xabcdabcd
line 59 -> moves control to Line 93 ; retf
KAMA: Check stack at 0xc050 (around it) for stack switching

Line 93 to 95: ES: 0xe3: PL: 1 segment Base: 0xc700; Limit 0x100
Line 95 causes a GPF
Look at the 13th descriptor in IDT Located at 0xd278; IDT Base is 0xd210
This is a Task Gate and the TSS is in 0x88
The TSS is at 0xb400 Limit 0x70
ISR is a task with TSS at 0xb400
EIP: 0x800
CS: 0x8 - Base: 0xa000 Limit: 0xffff PL: 0
ES:FS:GS - Null
SS: 0xc8 - Base: 0xc400; Limit: 0x100; PL: 0
DS: 0x18 - Base: 0xe000; Limit: 0x100; PL: 0
ESP: 0x50
GS: 0x18 - Base: 0xe000; Limit: 0x100; PL: 0
KAMA: 0xe020 : error code 0x9090909090 ; 0xe0
```

```
        ;=<Getting the descriptor for the faulting task>=
        ;[0xes:ebx] -> 0xb000 + 0x2004 + "sel from Error Code"(i.e. 0xe0)
        bts dword [es:ebx], 14 ; change DPL to 3
        jmp donegpf
        ;===========Hack=================================
hack:
        mov cx, 0x10
        mov es, cx ; ES points to 0xb000
        bts dword [es:0x20e4], 14 ;Getting the descriptor for the faulting task

        ;=====END of Fault Handling======================
donegpf:
        mov ax,0x18
        mov gs,ax
        mov dword [gs:0x20], ebx   ;==push some debug info
        iret
```

So, by making PL 3 then I go and do general protection for it. If I do that, what I am doing here is I am automatically going to 20e4 because I do not have the error code I go to automatically 20e4 and set that 14 bit to 1. That is what I am doing in this part of this code called hack. And then what I do back is I move ax comma 0x18 and push some budeg in 4 whatever CBX and pushing in to this. So, what is 18 here s slash xd018, this e000, so DS is already. So GS actually becomes 0x18 which is base is 0xe000 and limit is 0x100 and PL 0. I am moving to 0xe020, some error go. This can be just 90, 90, 90, 90, 90 or it can be some e0 depending on the virtual box version we are using.

(Refer Slide Time: 26:40)



```
0xb420: 0x00000000
(gdb) x/x 0xb44c
0xb44c: 0x00000008
(gdb) x/x 0xb008
0xb008: 0x00000038
(gdb) x/x 0xd008
0xd008: 0xa000bfff
(gdb) x/2x 0xd008
0xd008: 0xa000bfff
(gdb) x/x 0xb448
0xb448: 0x00000000
(gdb) x/4x 0xb450
0xb450: 0x000000c8
0000
(gdb) x/2x 0xd0c8
0xd0c8: 0xc4000100
(gdb) x/2x 0xd018
0xd018: 0xe0000100
(gdb) x/x 0xb438
0xb438: 0x00000050
(gdb) x/x 0xd018
0xd018: 0xe0000100
(gdb)
```

```
Line 56: Base: 0xe000 Limit 0x100 PL: 0 ; aligned to address 0xa100
KAMA: 0xe040 <- 0xabcdabcd
line 59 -> moves control to Line 93 ; retf
KAMA: Check stack at 0xc050 (around it) for stack switching

Line 93 to 95: ES: 0xe3: PL: 1 segment Base: 0xc700; Limit 0x100
Line 95 causes a GPF
Look at the 13th descriptor in IDT Located at 0xd278; IDT Base is 0xd210
This is a Task Gate and the TSS is in 0x88
The TSS is at 0xb400 Limit 0x70
ISR is a task with TSS at 0xb400
EIP: 0x800
CS: 0x8 - Base: 0xa000 Limit: 0xffff PL: 0
ES:FS:GS - Null
SS: 0xc8 - Base: 0xc400; Limit: 0x100; PL: 0
DS: 0x18 - Base: 0xe000; Limit: 0x100; PL: 0
ESP: 0x50
GS: 0x18 - Base: 0xe000; Limit: 0x100; PL: 0
KAMA: 0xe020 : error code 0x9090909090 ; 0xe0
```

```
; Task 4 - PL 3
;==================================
        ;==========CALL GATE=========
        call 0xeb:0  ;CALL gate with descriptor 0xe8 and RPL 3
        ;===Return from CALL GATE====

        ;====GPF Fault===============
        mov ax, 0xe3
        ;====Example data move=======
        mov es, ax

        mov dword [es:0], 0xbabecafe
        ;IDT index: 13, IDT Address: 0xd210 + (13*8bytes) = 0xd278
        ;Entry at 0xd278: Task Gate -> seg selector 0x88
        ;TSS starts at 0xb400
        ;CS in the TSS starts at offset 76(0x4c) i.e. 0xb44c
        ;CS points to seg selector 0x8
```

So I will just repeat what I am trying to do. Once it is DPF comes some virtual boxes below version 5 actually pushes the error code which should be pushed. The virtual boxes after version 5, for some reason does not push error code into the stack. So what we do is, we just pop EBX if that EBX has null value that means, the virtual box has not pushed error code there the x86 machine (Refer Time: 27:06) is not pushing the error code so I jmp to hack.

Otherwise what will that error code have, the error code in the case of a GPF is such type of a segmentation violation will have the value of the segment descriptor. So will use this segment descriptor that should be e3 and end it with fff8 so essentially I get e0 here to which I had 2004 I will have 20e7. Now I am using the 10th descriptor which has base as b000, b000 plus this 20e7 will take me to the second word of the descriptor in which I am setting the 14 bit to 1. Essentially, I am changing the descriptor e0 from PL 0 to PL 3 and I am done GPF. In the hack since I do not have the error code I directly go and change that 20e4 with the 14 bit there and that basically works here. So, 20e4 is set here because of the error code. Normally, this is better because this is more generic, but this is an adopting that we need to do because some virtual box, do not push the error code. And then we just go and move to this e020 this debug information what was there in EXB, was EBX had the null value or did it have the other value.