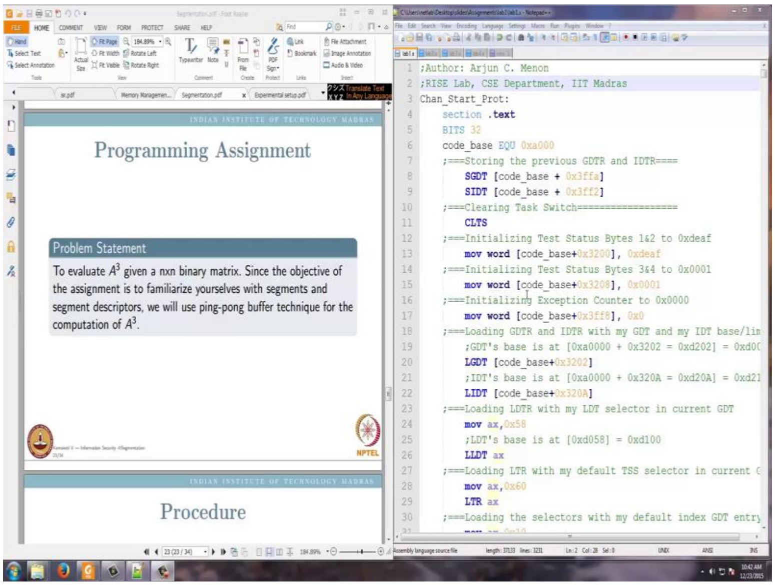
Information Security - II
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 21
Lab1 Part 1 - Week 4

(Refer Slide Time: 00:09)



INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Programming Assignment

Problem Statement

To evaluate A^3 given a nxn binary matrix. Since the objective of the assignment is to familiarize yourselves with segments and segment descriptors, we will use ping-pong buffer technique for the computation of A^3.

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

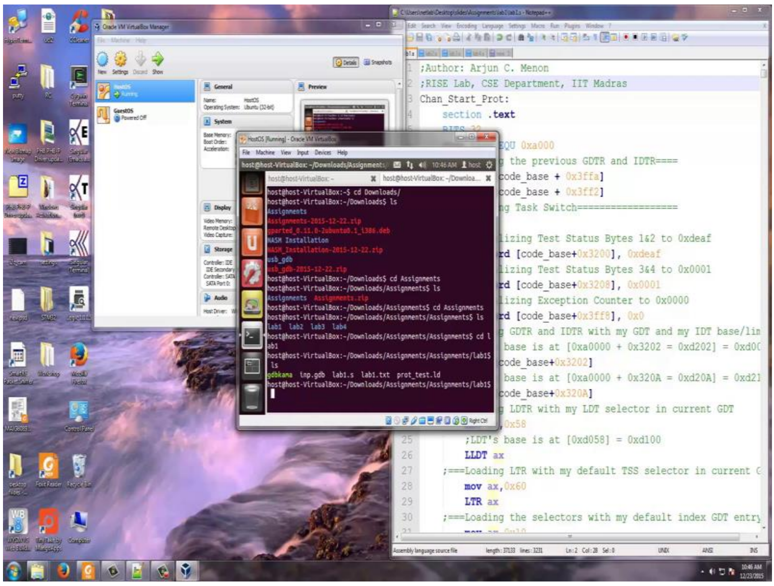
Procedure

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lim
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current C
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

Now, I invite you to the session where we are going to demonstrate the assignment related to Segmentation. Before we go into the segmentation assignment let me just first show you how you invoked the Virtual Box. I hope you have gone through the software installation video, which we have loaded in the website and you have installed the virtual box in the way it should be done along with the guest and the host operating systems, I hope you have done all those things and your laptop is ready.

Now, let me just show you how we invoke the virtual box. There will have a oracle VM virtual box logo in your system and double click on that, you can go and ignore it. Now, you is a host OS and a guest OS you should see this. So first go to the host OS and click start.

(Refer Slide Time: 01:20)



```
host@host-VirtualBox:~$ cd Downloads/
host@host-VirtualBox:~/Downloads$ ls
Assignments
Assignments-2015-12-22.zip
gparted_0.11.0-2ubuntu0.1_i386.deb
NASM Installation
NASM_Installation-2015-12-22.zip
usb_gdb
usb_gdb-2015-12-22.zip
host@host-VirtualBox:~/Downloads$ cd Assignments
host@host-VirtualBox:~/Downloads/Assignments$ ls
Assignments  Assignments.zip
host@host-VirtualBox:~/Downloads/Assignments$ cd Assignments
host@host-VirtualBox:~/Downloads/Assignments/Assignments$ ls
lab1  lab2  lab3  lab4
host@host-VirtualBox:~/Downloads/Assignments/Assignments$ cd l
ab1
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  inp.gdb  lab1.s  lab1.txt  prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
```

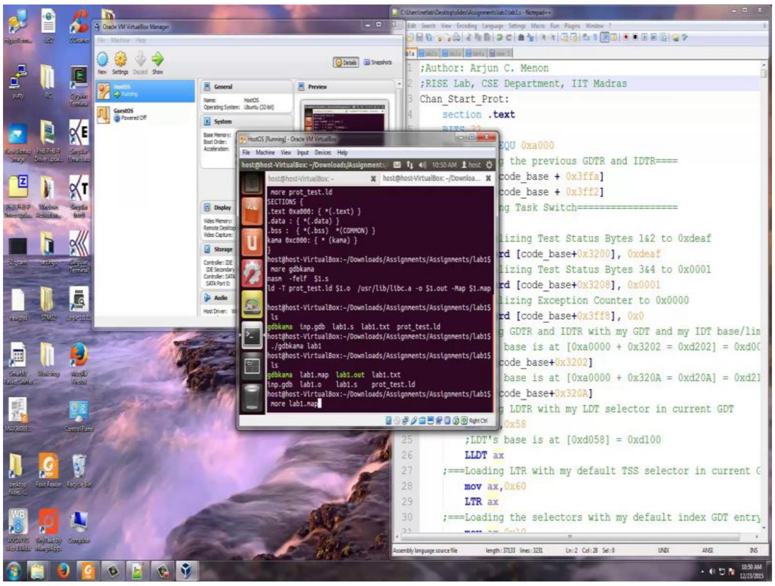
```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
             EQU 0xa000
           g the previous GDTR and IDTR====
           code_base + 0x3ffa]
           code_base + 0x3ff2]
           ng Task Switch=================

           lizing Test Status Bytes 1&2 to 0xdeaf
           rd [code_base+0x3200], 0xdeaf
           lizing Test Status Bytes 3&4 to 0x0001
           rd [code_base+0x3208], 0x0001
           lizing Exception Counter to 0x0000
           rd [code_base+0x3ff8], 0x0
           g GDTR and IDTR with my GDT and my IDT base/lim
            base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
           code_base+0x3202]
            base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
           code_base+0x320A]
           g LDTR with my LDT selector in current GDT
           ,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

.

And you will see some auto capture keyboard all those things. Please close everything here and now going to starts booting. Iit will come to a stage which is the host there you would have entered a password when you had created the system so the same password you enter here. And now you see a Linux Ubuntu system, which is your host OS running here. And what you have downloaded here, you would have already downloaded the Noisome and also all the assignment files using the fire fox web browser inside this. This we had told as a part of our introductory software installation video and you would have done that now go and click this dash home and you will see the terminal, if you do not see terminal just type term and you will see terminal here click on this terminal. So you see a terminal here.

So, now you can open another terminal by another tab in this terminal by going to file and save open tab, so you will have 2 terminals here we need that 1 for compilation and another for this. Now we are going to do Lab2. So, we go to the directory CD downloads and we see CD assignments again do CD assignments, sorry we are going to do lab 1. We go to CD Lab1 and there you see 4 files, the Lab1 dot S. The Lab1 dot S is the assembly program. GDB comma is the executable shell script that will compile your assembly program into extended line x format which you will download on to the guest machine and execute it. Inp dot GDB actually has the comments which we will use to configure the remote machine from the host machine. And prot test dot LD is a loader file, which basically tells where you need to load the software. I will first start with prot test dot LD, I am saying more prot test dot LD that will give you all the things here. Now as you see here, the file essentially says that load the program at 0xa1000. So in the remote machine we are loading the program at 0xa1000, why are we loading it at 0xa1000? We are loading at 0xa1000 because as I told you in the introductory lecture for software installation that the remote machine has a GDB kernel running. The GDB kernel will be occupying certain space in the RAM and the space is less than a1000 address. So, the entire GDB kernel will be occupying the address below a1000, so we will load our program at a1000 and above. That is what this loader file basically says.

(Refer Slide Time: 05:50)



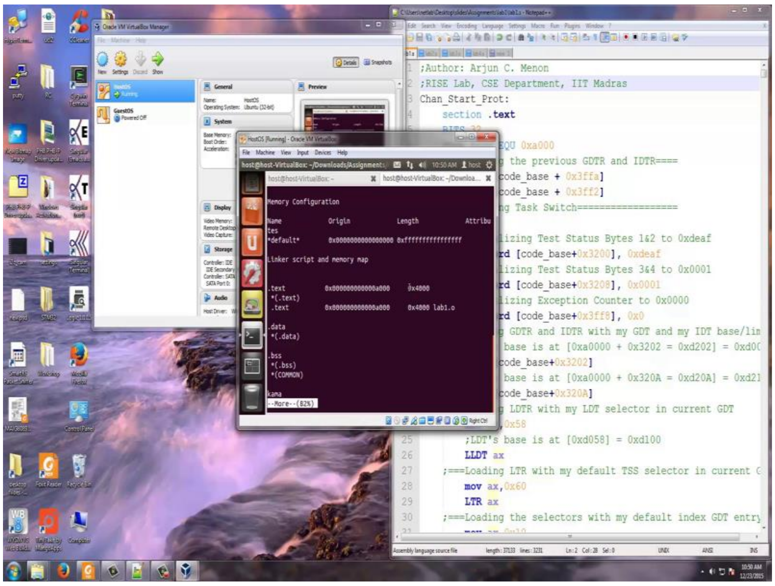
```
 more prot_test.ld
SECTIONS {
.text 0xa000: { *(.text) }
.data : { *(.data) }
.bss :  { *(.bss)  *(COMMON) }
kama 0xc000: { * (kama) }
}
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 more gdbkama
nasm  -felf  $1.s
ld -T prot_test.ld $1.o  /usr/lib/libc.a -o $1.out -Map $1.map

host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  inp.gdb  lab1.s  lab1.txt  prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ./gdbkama lab1
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  lab1.map  lab1.out  lab1.txt
inp.gdb  lab1.o    lab1.s    prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 more lab1.map
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
           EQU 0xa000
          g the previous GDTR and IDTR====
           code_base + 0x3ffa]
           code_base + 0x3ff2]
          ng Task Switch=================

          lizing Test Status Bytes 1&2 to 0xdeaf
          rd [code_base+0x3200], 0xdeaf
          lizing Test Status Bytes 3&4 to 0x0001
          rd [code_base+0x3208], 0x0001
          lizing Exception Counter to 0x0000
          rd [code_base+0x3ff8], 0x0
          g GDTR and IDTR with my GDT and my IDT base/lim
           base is at [0xa0000 + 0x3202 = 0xd202] = 0xd00
           code_base+0x3202]
           base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
           code_base+0x320A]
          g LDTR with my LDT selector in current GDT
          ,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

Now I will go and tell you what GBD comma has. GBD comma, if you say more GBD comma it has 2 commands first is noisome minus f elf dollar 1 dot S. So, it will compile noisome is the network assembler you would have downloaded it if you had followed the introductory video of installation of the software, you would have downloaded noisome into this machine so it will be available here. So, this will compile the given S 5. In this case, our Lab1 dot S as you see here will be compiled by noisome and in the elf format. And then this compiled file will be converted into a loadable file with inputs from prot test or LD and it will give me a file called Lab1 dot out in this case. This Lab1 dot out I will upload it into the guest machine and the GDB is sitting there in the guest machine will load this file into 0xa1000, because your prot test dot LD says that we need to load it at a1000.

So let us go through that exercise, first I will compile this file called Lab1 and dot S. What is Lab1 dot S? I am going to explain as a part of this assignment. Let us first note, what are all the files here. These are all the files, now I say dot slash GDB comma Lab1 and now you see that Lab1 dot out is already created here. For guest I will tell you what is Lab1 dot map.

(Refer Slide Time: 07:49)

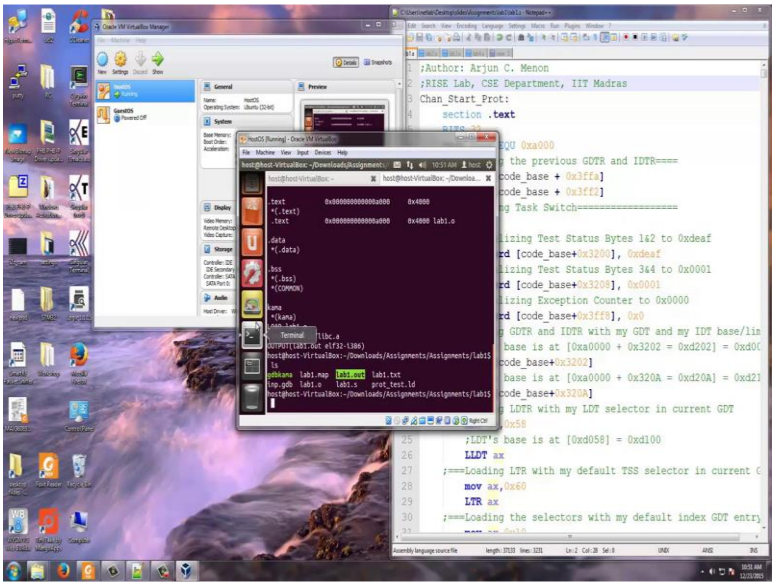


```
Memory Configuration

Name             Origin             Length             Attribu
tes
*default*        0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text           0x000000000000a000     0x4000
 *(.text)
 .text          0x000000000000a000     0x4000 lab1.o

.data
 *(.data)

.bss
 *(.bss)
 *(COMMON)

kama
--More--(82%)
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
              EQU 0xa000
             g the previous GDTR and IDTR====
             code_base + 0x3ffa]
             code_base + 0x3ff2]
             ng Task Switch=================

             lizing Test Status Bytes 1&2 to 0xdeaf
             rd [code_base+0x3200], 0xdeaf
             lizing Test Status Bytes 3&4 to 0x0001
             rd [code_base+0x3208], 0x0001
             lizing Exception Counter to 0x0000
             rd [code_base+0x3ff8], 0x0
             g GDTR and IDTR with my GDT and my IDT base/lim
              base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0
             code_base+0x3202]
              base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
             code_base+0x320A]
             g LDTR with my LDT selector in current GDT
             ,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

Lab 1 dot map says that there is a file dot text in the file which is loaded at 0x it is of size 0x4000 and this is of the format elf 32. So this basically tells us what are all there in your Lab1 dot S, we will discuss that in great detail when we look at Lab1 dot S.

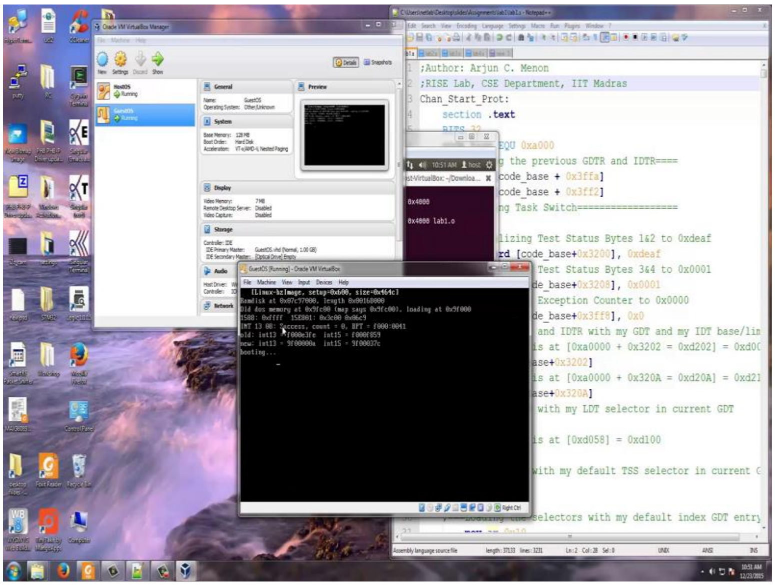
(Refer Slide Time: 08:23)



```
.text           0x000000000000a000     0x4000
 *(.text)
 .text          0x000000000000a000     0x4000 lab1.o

.data
 *(.data)

.bss
 *(.bss)
 *(COMMON)

kama
 *(kama)
                 libc.a
OUTPUT(lab1.out elf32-i386)
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  lab1.map  lab1.out  lab1.txt
inp.gdb  lab1.o    lab1.s    prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
```

Now Lab1 dot out is basically created. This Lab1 dot out has to be loaded on to the bare metal machine so that we understand how this works on that machine. Now we go back to the oracle VM virtual box manager and now we will go and start the guest OS machine.

(Refer Slide Time: 08:47)



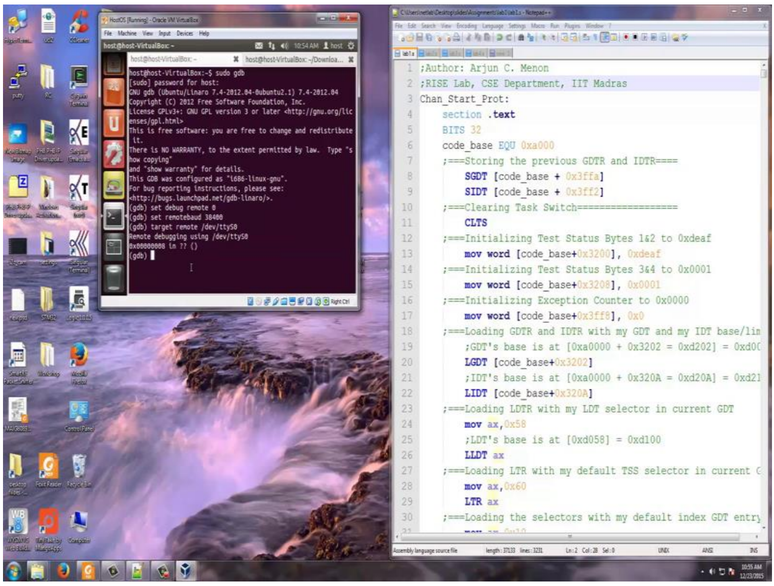
```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
```

```
0x4000

0x4000 lab1.o
```

```
  [Linux-bzImage, setup=0x600, size=0x464c]
Ramdisk at 0x07c97000, length 0x00160000
Old dos memory at 0x9fc00 (map says 0x9fc00), loading at 0x9f000
1588: 0xffff  15E801: 0x3c00 0x06c9
INT 13 08: Success, count = 0, BPT = f000:0041
old: int13 = f000e3fe  int15 = f000f859
new: int13 = 9f00000a  int15 = 9f00037c
booting...
```

Now, there it will say GDB boot, go and press the enter key. Note that in the guest machine I told that there is a simple GDB kernel running and this is that. This will be showing booting, so let it be in that shape. So this, if it has crossed the cursor as crossed that booting dot, dot, dot it is actually already booted. Now, what we will do is we have already compiled this.

(Refer Slide Time: 09:23)



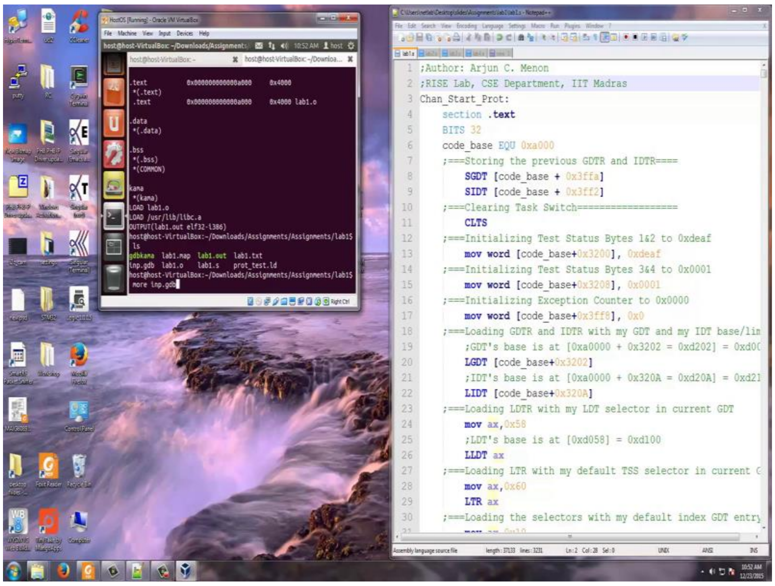
```
host@host-VirtualBox:~$ sudo gdb
[sudo] password for host:
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/lic
enses/gpl.html>
This is free software: you are free to change and redistribute
 it.
There is NO WARRANTY, to the extent permitted by law.  Type "s
how copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
(gdb) set debug remote 0
(gdb) set remotebaud 38400
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
0x00000000 in ?? ()
(gdb) 
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lim
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd00
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

.

We will now go to another tab, I said we opened 2 tabs here so we will go to another tab here. There are 1 tabs, 1 tab is for compilation, another tab is for connecting. So this tab we will use for connecting to the remote machine. I just says sudo GDB, I entered the password because GDB is a privilege program it works under super users so you have to do sudo GDB. Now GDB has come up, now I want to connect to the remote machine. So how do I do? The configuring as I told you is there in the inp dot GDB,

(Refer Slide Time: 10:17)



```
.text           0x000000000000a000      0x4000
 *(.text)
 .text          0x000000000000a000      0x4000 lab1.o

.data
 *(.data)

.bss
 *(.bss)
 *(COMMON)

kama
 *(kama)
LOAD lab1.o
LOAD /usr/lib/libc.a
OUTPUT(lab1.out elf32-i386)
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  lab1.map  lab1.out  lab1.txt
inp.gdb  lab1.o    lab1.s    prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 more inp.gdb
```

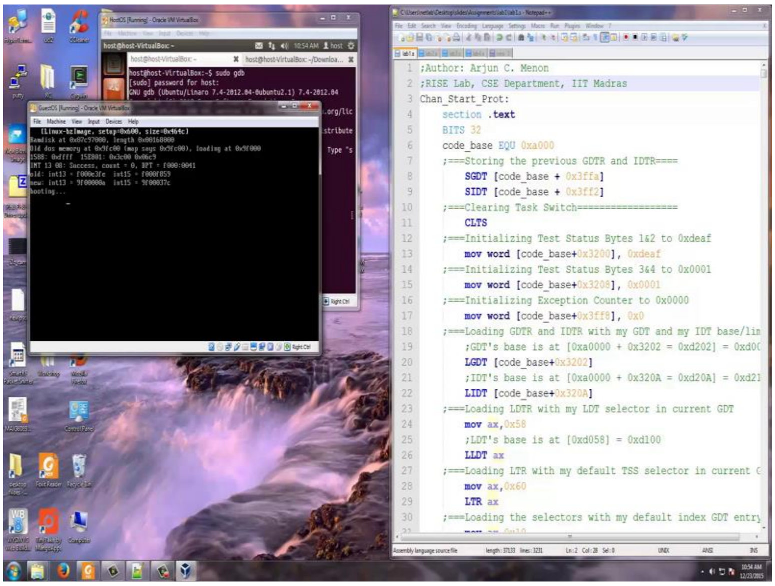
```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lim
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

HostOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
host@host-VirtualBox: ~/Downloads/Assignments/
10:52 AM
host
host@host-VirtualBox: ~
host@host-VirtualBox: ~/Downloa...
Right Ctrl
C:\Users\netlab\Desktop\slides\Assignments\lab1\lab1.s - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Assembly language source file
length : 37133 lines : 3231
Ln : 2 Col : 28 Sel : 0
UNIX
ANSI
INS
10:52 AM
12/23/2015

I go to the next tab and you see there are commence here. So first I do set debug remote 1. So I will do set debug remote 0, because 1 means it will keep giving you lot more output of what it is trying to do with the remote it will be sort of culturing your screen, so I will just say set debug remote 0. You can go to remote 1 and remote 2 and if you keep on increasing this debug level it will give you more information about how it is communicating with the hardware machine.

The next step is please note here I am setting remote bode 38400. Note that this machine is going to talk to the host machine through a mimicked serial port and bode rate of that serial port we need to set it as 38400, this is set. Now, this target remote slash devS0, you can actually even press right click and copy this and paste it here also, right click and it will give you this yeah target remote. This basically is this slash dev slashS0 essentially says I am communicating through the serial port. Now I have established a contact with the host machine with the guest machine, the bare metal hardware which is running here.

(Refer Slide Time: 12:02)



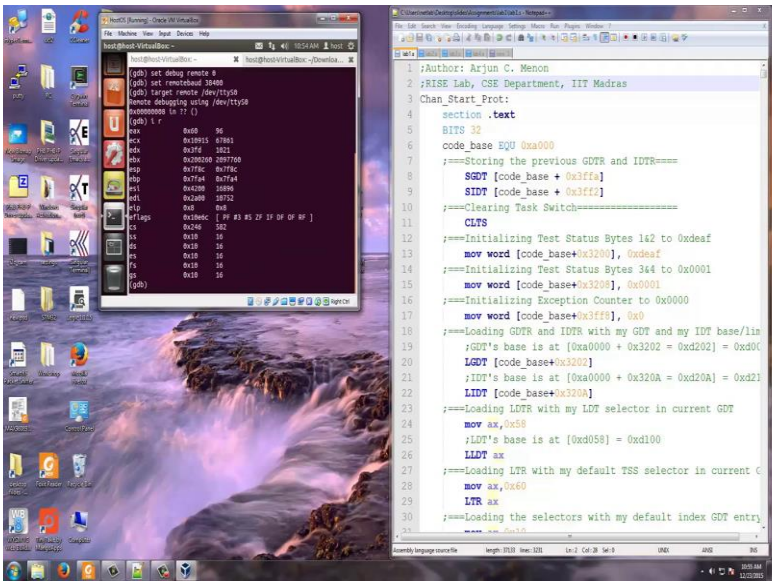
```
host@host-VirtualBox:~$ sudo gdb
[sudo] password for host:
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
```

```
   [Linux-bzImage, setup=0x600, size=0x464c]
Ramdisk at 0x07c97000, length 0x00168000
Old dos memory at 0x9fc00 (map says 0x9fc00), loading at 0x9f000
1588: 0xffff  15E801: 0x3c00 0x06c9
INT 13 08: Success, count = 0, BPT = f000:0041
old: int13 = f000e3fe  int15 = f000f859
new: int13 = 9f00000a  int15 = 9f00037c
booting...
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lin
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current C
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

So this is the bare metal machine, now my host machine gets established contact with this machine. So, you can confirm that by this comment 0x888.

(Refer Slide Time: 12:22)

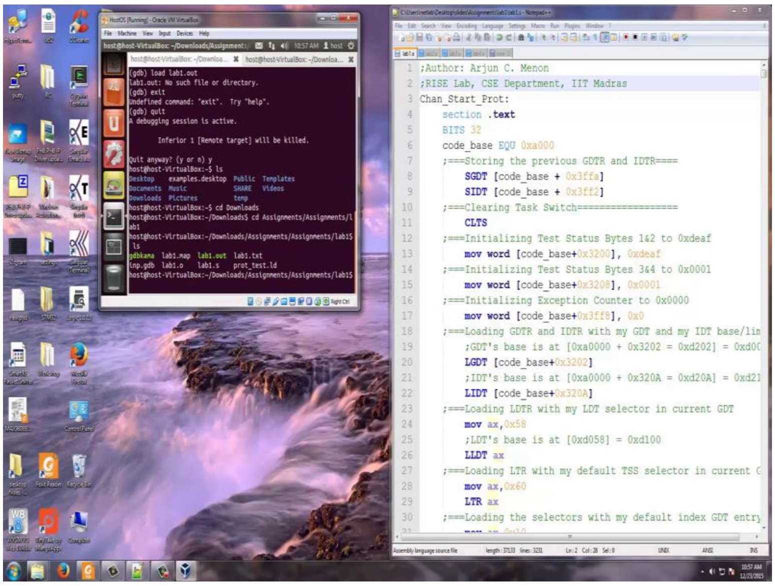


```
(gdb) set debug remote 0
(gdb) set remotebaud 38400
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
0x00000008 in ?? ()
(gdb) i r
eax            0x60     96
ecx            0x10915  67861
edx            0x3fd    1021
ebx            0x200260 2097760
esp            0x7f8c   0x7f8c
ebp            0x7fa4   0x7fa4
esi            0x4200   16896
edi            0x2a00   10752
eip            0x8      0x8
eflags         0x10e6c  [ PF #3 #5 ZF IF DF OF RF ]
cs             0x246    582
ss             0x10     16
ds             0x10     16
es             0x10     16
fs             0x10     16
gs             0x10     16
(gdb)
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lin
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current C
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

I will give you some very simple interface things. If I put i r this will tell the content of all the general purpose registers, the segment registers your instruction pointer and your E flags of your remote machine that is your guest machine. So, by this I can go and find the content of the registers in the remote machine, I can find the segment selectors and also the E flags.

(Refer Slide Time: 12:50)



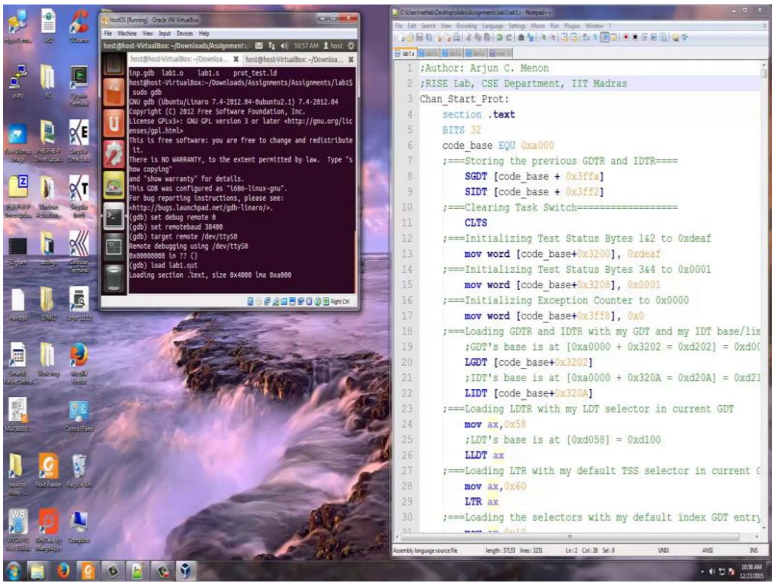
```
(gdb) load lab1.out
lab1.out: No such file or directory.
(gdb) exit
Undefined command: "exit".  Try "help".
(gdb) quit
A debugging session is active.

        Inferior 1 [Remote target] will be killed.

Quit anyway? (y or n) y
host@host-VirtualBox:~$ ls
Desktop    examples.desktop  Public  Templates
Documents  Music             SHARE   Videos
Downloads  Pictures          temp
host@host-VirtualBox:~$ cd Downloads
host@host-VirtualBox:~/Downloads$ cd Assignments/Assignments/l
ab1
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 ls
gdbkama  lab1.map  lab1.out  lab1.txt
inp.gdb  lab1.o    lab1.s    prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lin
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd0C
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

Similarly, I can also go and find out what is there in say a1000 x slash x, means access the memory in the address in extra decimal format the output also we need in extra decimal format (Refer Time: 13:07). This gives you the content of 4 bytes, so in a1000 83 is stored, in a1001 ec is stored, a1002 1c is stored and in a1003 85 is stored. Now, I can also say x slash 2x0xa1000, this will give me 24 bytes. So, in a1000 again 83, a1001 ec, a1002 1c, a1003 85 now it can go and say a1004 is d2, a1005 is 7f, a1006 is 1f and a1007 is b8. I can query the memory, I can query all the general purpose registers. So, this is basically the debug interface that we will have.

Now what I am going to do is I have already compiled Lab1 dot out. So, I am going to load Lab1 dot out here. Why this file direct is happening because I have not moved to this directly, so just I exist GDB again, quit this GDB, used a word quit exist does not work quit. Now, I am in virtual box I need to again go to downloads here I need to go to assignments again assignments and then lab 1, now you see I have Lab1 dot out here.

(Refer Slide Time: 14:53)

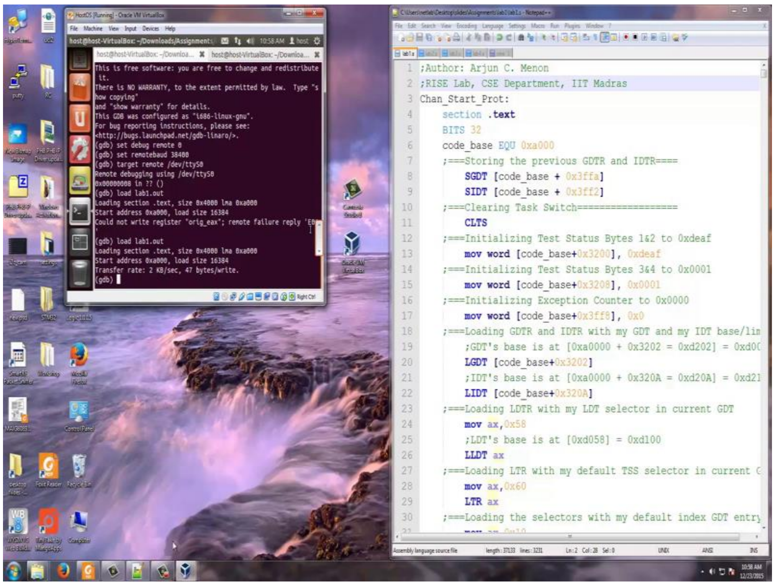


```
inp.gdb  lab1.o    lab1.s    prot_test.ld
host@host-VirtualBox:~/Downloads/Assignments/Assignments/lab1$
 sudo gdb
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/lic
enses/gpl.html>
This is free software: you are free to change and redistribute
 it.
There is NO WARRANTY, to the extent permitted by law.  Type "s
how copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
(gdb) set debug remote 0
(gdb) set remotebaud 38400
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
0x00000000 in ?? ()
(gdb) load lab1.out
Loading section .text, size 0x4000 lma 0xa000
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lin
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd00
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

So now I can says sudo GDB, GDB is already here I got set debug remote 0 set remote bode 38400, target remote slash dev slash ttyS0 everything is there now I say load Lab1 dot out, it is loading. You see that this program Lab1 dot s is of size 4000. We are really mimicking the serial port.

(Refer Slide Time: 15:40)



```
This is free software: you are free to change and redistribute
 it.
There is NO WARRANTY, to the extent permitted by law.  Type "s
how copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
(gdb) set debug remote 0
(gdb) set remotebaud 38400
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
0x00000000 in ?? ()
(gdb) load lab1.out
Loading section .text, size 0x4000 lma 0xa000
Start address 0xa000, load size 16384
Could not write register "orig_eax"; remote failure reply 'E0
'

(gdb) load lab1.out
Loading section .text, size 0x4000 lma 0xa000
Start address 0xa000, load size 16384
Transfer rate: 2 KB/sec, 47 bytes/write.
(gdb)
```

```
;Author: Arjun C. Menon
;RISE Lab, CSE Department, IIT Madras
Chan_Start_Prot:
    section .text
    BITS 32
    code_base EQU 0xa000
    ;===Storing the previous GDTR and IDTR====
        SGDT [code_base + 0x3ffa]
        SIDT [code_base + 0x3ff2]
    ;===Clearing Task Switch==================
        CLTS
    ;===Initializing Test Status Bytes 1&2 to 0xdeaf
        mov word [code_base+0x3200], 0xdeaf
    ;===Initializing Test Status Bytes 3&4 to 0x0001
        mov word [code_base+0x3208], 0x0001
    ;===Initializing Exception Counter to 0x0000
        mov word [code_base+0x3ff8], 0x0
    ;===Loading GDTR and IDTR with my GDT and my IDT base/lin
        ;GDT's base is at [0xa0000 + 0x3202 = 0xd202] = 0xd00
        LGDT [code_base+0x3202]
        ;IDT's base is at [0xa0000 + 0x320A = 0xd20A] = 0xd21
        LIDT [code_base+0x320A]
    ;===Loading LDTR with my LDT selector in current GDT
        mov ax,0x58
        ;LDT's base is at [0xd058] = 0xd100
        LLDT ax
    ;===Loading LTR with my default TSS selector in current G
        mov ax,0x60
        LTR ax
    ;===Loading the selectors with my default index GDT entry
```

So now you see that there is some error message. There is an error in communicating, so again I load it, every time you may have to load it, once or twice. Once, you get this transfer at these 3 comments, now you have successful loaded. We have successfully loaded and now we can start the assignment in detail.