x86 HW2

2019. 05. 17

Jeon Jae Wook
Sungkyunkwan Univ.

Contents

- Describe about 1st Homework
- 2nd Homework

Describe about 1st Homework

- "2015000000" display in real mode

```
;Print ID
    mov al, byte [ID]
    mov byte [es : 80*2*1+2*0], al
    mov byte [es : 80*2*1+2*0+1], 0x05

    mov al, byte [ID+1]
    mov byte [es : 80*2*1+2*1], al
    mov byte [es : 80*2*1+2*1+1], 0x05

    mov al, byte [ID+2]
    mov byte [es : 80*2*1+2*2], al
    mov byte [es : 80*2*1+2*2+1], 0x05

    mov al, byte [ID+3]
    mov byte [es : 80*2*1+2*3], al
    mov byte [es : 80*2*1+2*3+1], 0x05
```

```
ID  db 'ID : 2015000000',0
```

```
mov ax, 0xB800
mov es, ax
```

```
mov bp, sp
mov bx, [ss:bp]
```

- Start address of video memory
  - 0xB8000
- Text data property
  - High 4bit : Background color of text
  - Low 4bit : Text color
- Memory access
  - [es:offset]
    - es*10H+offset
  - stack pointer → [ss:offset]

2nd Homework

- 2nd Homework Describe
  - Switch to Protected Mode
    - Make a GDT (Global Descriptor Table)
    - Load GDT
    - Set Control Register 0
    - Check the result
  - Make a LDT
    - Make descriptor in GDT
    - Load LDT
    - Check the result

Switch to Protected Mode

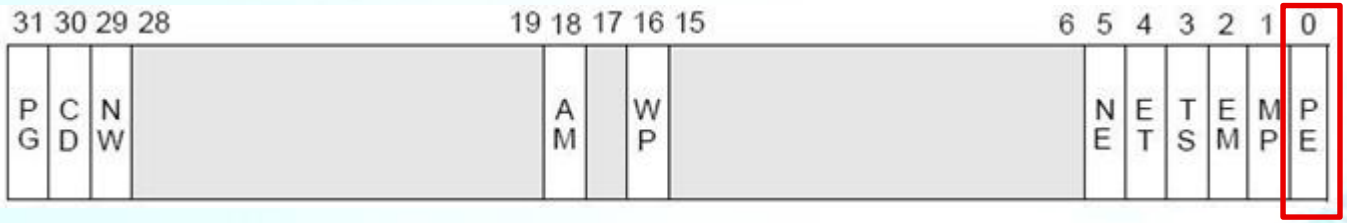
- **Transition from Real Mode to Protected Mode**
  - **Make a GDT**
    - **Contains some segment descriptors like code, data, extra, etc.**
  - **Load the Limit and Base Address of GDT**
    - **Calculate the base address and limit of GDT**
    - **Store this value in GDTR register (gdt_ptr)**
    - **Load the address of GDT into GDTR**
      - **Use lgdt instruction → lgdt [GDTR register]**

System Table Registers

	47(79) … 16	15 … 0
GDTR	32(64)-bit Linear Base Address	16-Bit Table Limit
IDTR	32(64)-bit Linear Base Address	16-Bit Table Limit

Switch to Protected Mode

- **Transition from Real Mode to Protected Mode**
  - **Control Register 0**
    - Register cr0 is the 32-bit version of the MSW reg.(Machine Status Word)
    - It contains the PE-bit(Protection Enabled) at lowest bit position
      - When PE=0 the CPU is in real-mode
      - When PE=1 the CPU is in protected-mode
    - Set ‘1’ the lowest bit(PE bit) of CR0 register to protected mode enable
    - Cannot use cr0 and operand directly
      - mov cr0, 0x12345678 is invalid combination
      - Use another register for setting cr0 PE bit



  - **jmp SYS_CODE_SEL_1:Protected_START**
    - Jump to Protected_START
    - Remove prefetch input queue

Global Descriptor Table

- **Rule of making a GDT**
  - **The 1st descriptor in GDT is Null Descriptor**
  - **GDT must have at least one code and data segment descriptor**
- **Make a GDT used for Homework**
  - **Null Descriptor (idx:0)**
    - **This is not used**
    - **Make all 0s in descriptor**
  - **Code Segment Descriptor (idx:1)**
    - **Base Address : 0x00000000 / Limit : 0xFFFFF**
    - **Type : non-conforming, execute/read, not accessed**
    - **Other Information**
      - **In IA-32 mode and 32-bit code segments**
      - **Descriptor Privilege Level is 0**
      - **Present in Memory**
      - **Limit is interpreted in 4-Kbyte units**
      - **Not available for use by system software**

Global Descriptor Table

- **Data Segment Descriptor (idx:2)**
  - **Base Address : 0x00000000 / Limit : 0xFFFFF**
  - **Type : expand up, read/write, not accessed**
  - **Other Information**
    - **In IA-32 mode and 32-bit code segments**
    - **Descriptor Privilege Level is 0**
    - **Present in Memory**
    - **Limit is interpreted in 4-Kbyte units**
    - **Not available for use by system software**
- **Video Segment Descriptor (idx:3)**
  - **Base Address : 0x000B8000 / Limit : 0xFFFF**
  - **Type : expand up, read/write, not accessed**
  - **Other Information**
    - **In IA-32 mode and 32-bit data segments**
    - **Descriptor Privilege Level is 0**
    - **Present in Memory**
    - **Limit is interpreted in byte units**
    - **Not available for use by system software**

Global Descriptor Table

- **Code Segment Descriptor (idx:5)**
  - **Base Address : 0x00000000 / Limit : 0xFFFFF**
  - **Type : non-conforming, execute/read, not accessed**
  - **Other Information**
    - **In IA-32 mode and 32-bit code segments**
    - **Descriptor Privilege Level is 0**
    - **Present in Memory**
    - **Limit is interpreted in 4-Kbyte units**
    - **Not available for use by system software**

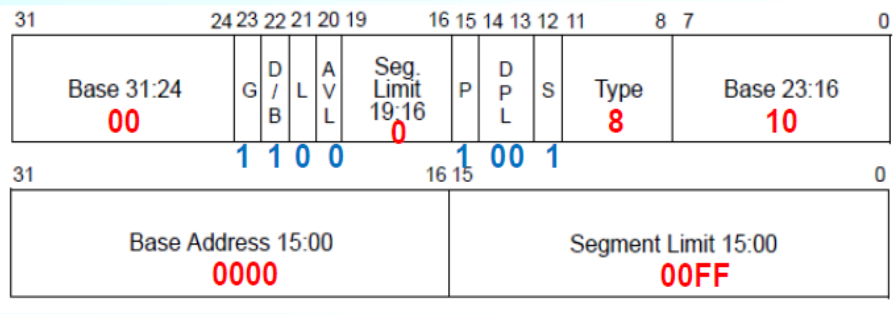
Global Descriptor Table

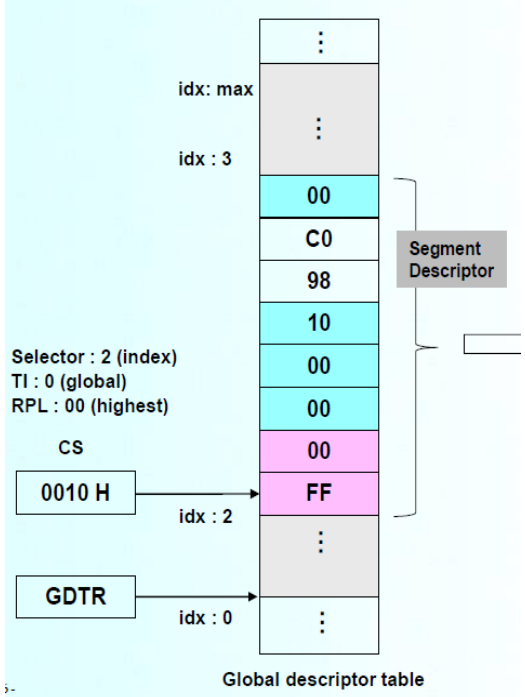
Global Descriptor Table

Index	Segment Selector	TYPE
0	-	NULL Descriptor
1	SYS_CODE_SEL_0	Code Segment Descriptor
2	SYS_DATA_SEL	Data Segment Descriptor
3	VIDEO_SEL	Data Segment Descriptor
4		
5	SYS_CODE_SEL_1	Code Segment Descriptor

Global Descriptor Table

- **Make a Descriptor (Ex2 of lecture note)**
  - **Code segment selector**
    - **0010H (0000 0000 0001 0000B)**
  - **Descriptor 2 contains**
    - **00C0 9810 0000 00FF H**

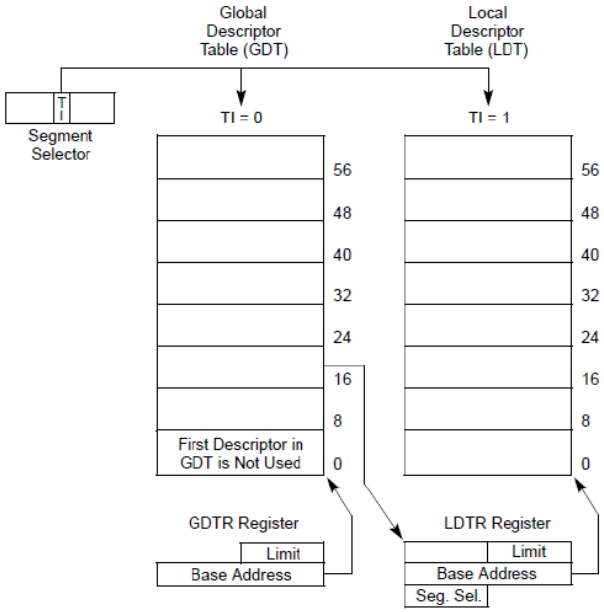





```
Segment_Selector    equ     10h
gdt:
            dw          00FFh           ; limit 15:0
            dw          0000h           ; base 15:0
            db          10h             ; base 23:16
            db          98h             ; type
            db          C0h             ; limit 19:16, flags
            db          00h             ; base 31:24
```

Local Descriptor Table

- Memory addressing using LDT
  - Make LDTR descriptor in GDT
    - Base address : base address of LDT
    - Limit : 0xFFFF
    - Type : System Descriptor, LDT
    - Other Information
      - In IA-32mode
      - Descriptor Privilege Level is 0
      - Present in Memory
      - Limit is interpreted in byte units
      - Not available for use by system software



Local Descriptor Table

- **Local Descriptor Table Register**
  - **An LDT is accessed with its segment selector**
  - **The LDTR register holds**
    - **16-bit segment selector**
    - **Base address and segment limit**
    - **Descriptor attributes for LDT**
  - **Load LDT**
    - **LLDT instruction**
    - **Load a segment selector of LDTR descriptor**
      - **The base, limit, attributes from LDT are automatically loaded in the LDTR**

	System Segment Registers (15 … 0)	Segment Descriptor Registers (Automatically Loaded)				
				Attributes		
Task Register	Seg. Sel.	32(64)-bit Linear Base Address	Segment Limit			
LDTR <Intel>	Seg. Sel.	32(64)-bit Linear Base Address	Segment Limit			

Local Descriptor Table

- **Code Segment Descriptor (idx : 0)**
  - Base Address : 0x00000000 / Limit : 0xFFFFF
  - Type : non-conforming, execute/read, not accessed
  - Other Information
    - In IA-32 mode and 32-bit code segments
    - Descriptor Privilege Level is 0
    - Present in Memory
    - Limit is interpreted in 4-Kbyte units
    - Not available for use by system software
- **Data Segment Descriptor (idx : 1)**
  - Base Address : 0x00000000 / Limit : 0xFFFFF
  - Type : expand up, read/write, not accessed
  - Other Information
    - In IA-32 mode and 32-bit data segments
    - Descriptor Privilege Level is 0
    - Present in Memory
    - Limit is interpreted in 4-Kbyte units
    - Not available for use by system software

Local Descriptor Table

Global Descriptor Table

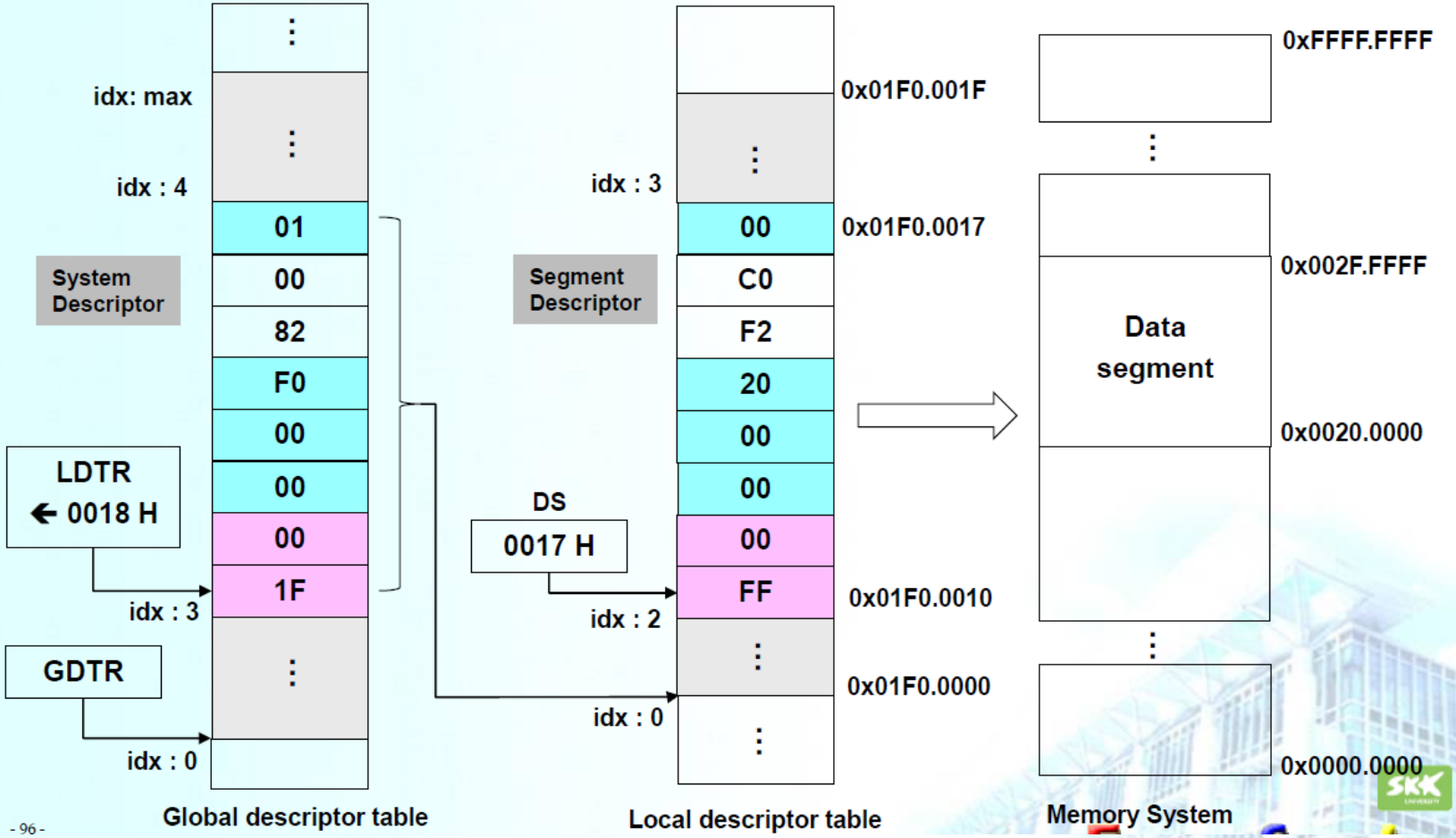
Index	Segment Selector	TYPE
0	-	NULL Descriptor
1	SYS_CODE_SEL_0	Code Segment Descriptor
2	SYS_DATA_SEL	Data Segment Descriptor
3	VIDEO_SEL	Data Segment Descriptor
4	LDTR	System Descriptor
5	SYS_CODE_SEL_1	Code Segment Descriptor

Local Descriptor Table

Index	Segment Selector	TYPE
0	LDT_CODE_SEL_0	Code Segment Descriptor
1	LDT_DATA_SEL_0	Data Segment Descriptor

Local Descriptor Table

- **Memory addressing using LDT (EX3 of lecture note)**



- 96 -

Jump Instruction

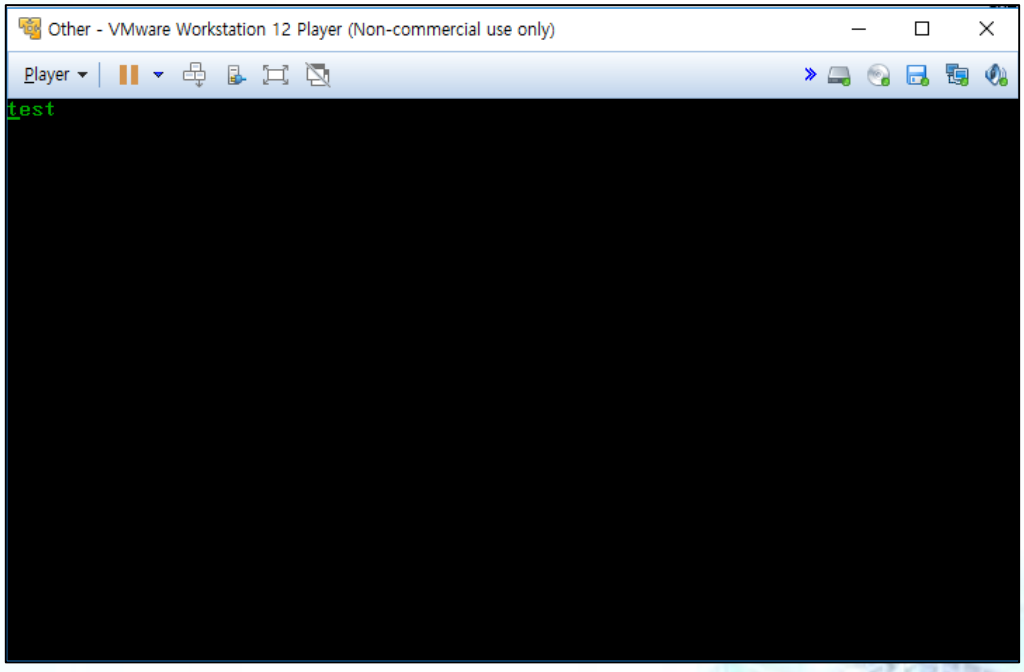
- Jump Instruction
  - Far jump
    - Destination is in a different code segment
  - Instructions
    - jmp CS selector:offset
  - A logical address consisting of
    - A 16-bit segment selector
      - Base address
    - A 32-bit offset
      - EIP ← offset
  - A far jump to a code segment at the same privilege level
    - CS ← the new code segment selector and its descriptor
    - EIP ← the offset from the instruction

- Transfer control (move other code segment)
  - Far Jump (jmp 0x04:LDT0_Start)
    - Protected_START → LDT0_Start

Caution

- When you use comments with the code below
  - Only ‘test’ string print

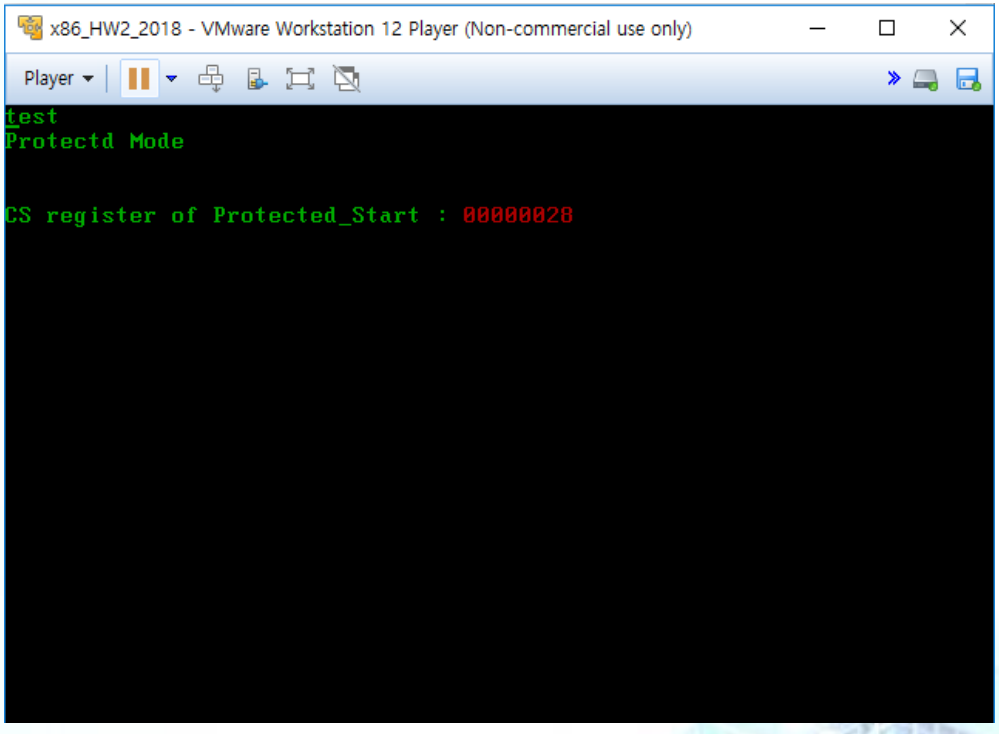
```
    jmp $
;----------------------Write your code here-----------------------
;                                                                  ;
; Store the value of Selector which indicates the domain           ;
; of Video Memory on ES Register                                   ;
;                                                                  ;
;------------------------------------------------------------------
    call print_protected
    call print_cs_Protected
    ;jmp $
;-------------------------write your code here---------------------
; Put base address of ldt to ldtr descriptor                       ;
; Load ldt                                                         ;
; Jump to LDT0_Start                                               ;
;                                                                  ;
;------------------------------------------------------------------
    ;jmp 0x04:LDT0_Start

LDT0_Start:

    call print_cs_LDT0_Start

    ;jmp $
```



Caution

- **When you use comments with the code below**
  - **Switch to the protected mode**

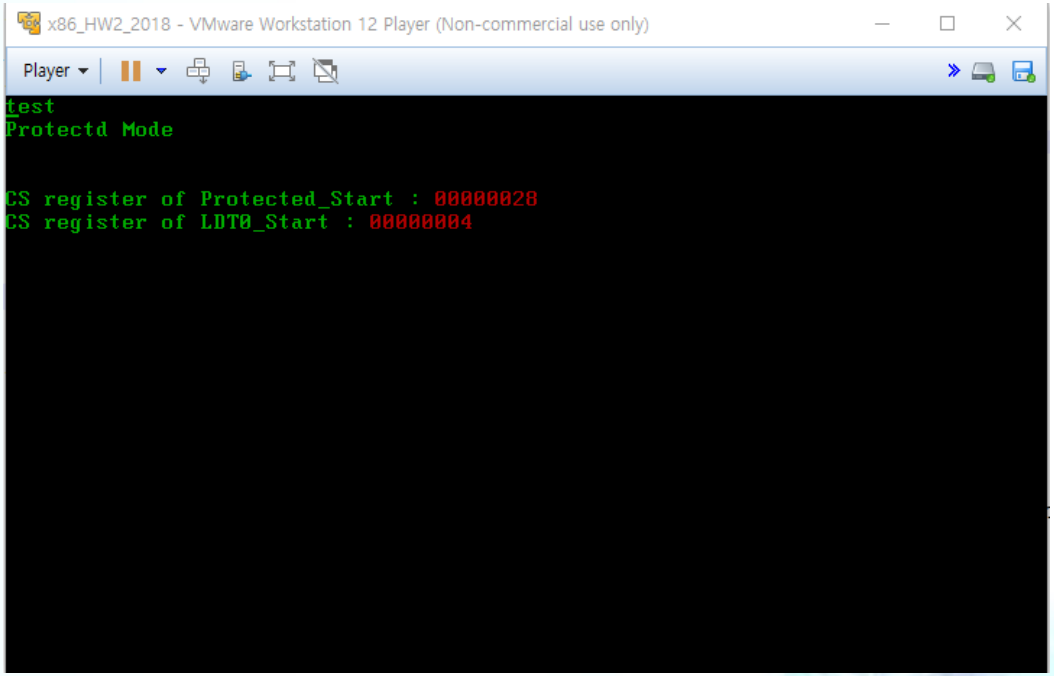
```
    ;jmp $
;----------------------Write your code here------------------------
;                                                                  ;
; Store the value of Selector which indicates the domain           ;
; of Video Memory on ES Register                                   ;
;                                                                  ;
;------------------------------------------------------------------
    call print_protected
    call print_cs_Protected
    jmp $
;-------------------------write your code here----------------------
; Put base address of ldt to ldtr descriptor                       ;
; Load ldt                                                         ;
; Jump to LDT0_Start                                               ;
;                                                                  ;
;------------------------------------------------------------------
    ;jmp 0x04:LDT0_Start

LDT0_Start:

    call print_cs_LDT0_Start

    ;jmp $
```



Caution

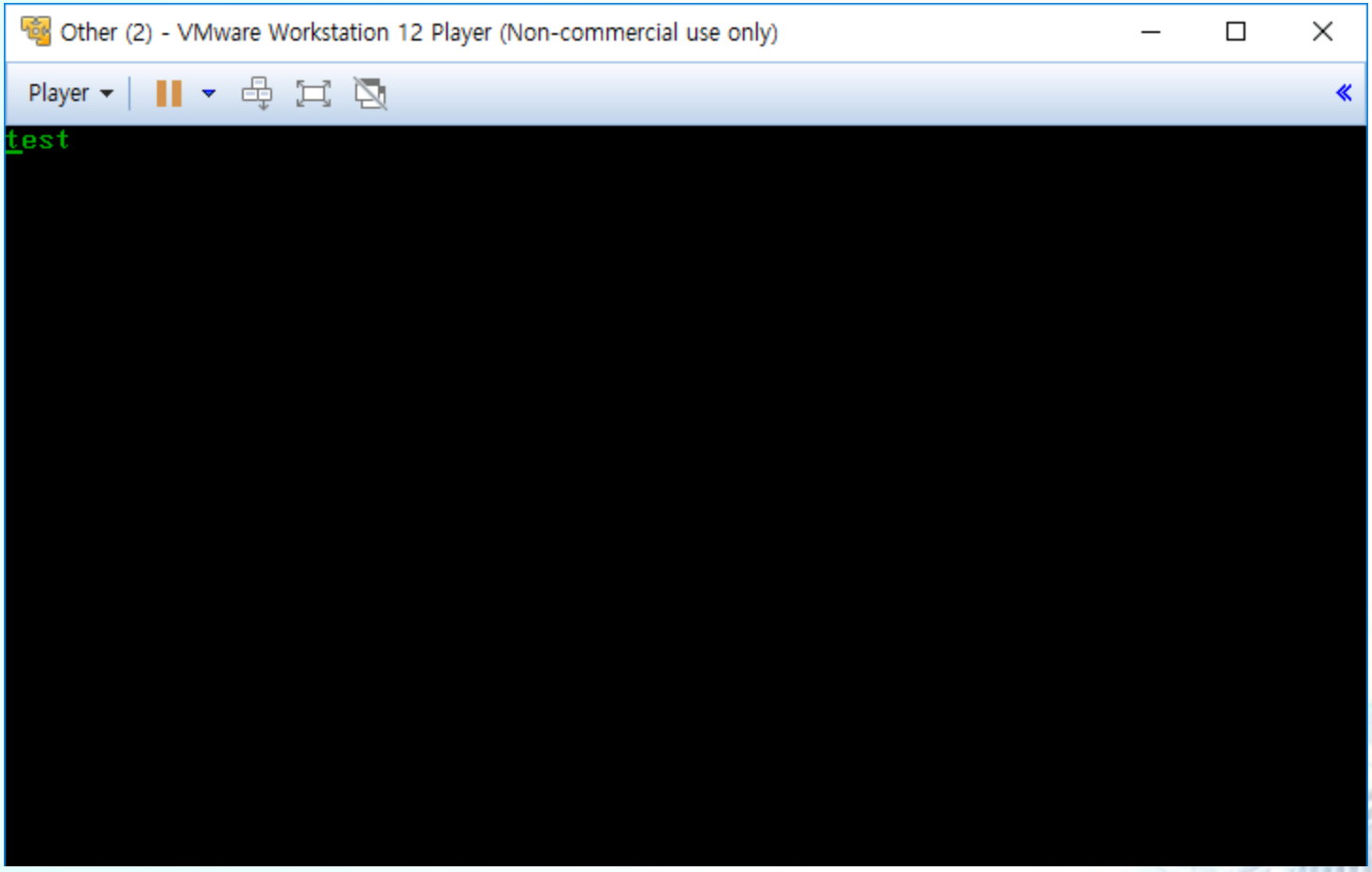
- **When you use comments with the code below**
  - **Jump to LDT0_Start label using CS in LDT**

```
    jmp $
;----------------------Write your code here-----------------------
;                                                                 ;
; Store the value of Selector which indicates the domain          ;
; of Video Memory on ES Register                                  ;
;                                                                 ;
;-----------------------------------------------------------------
    call print_protected
    call print_cs_Protected
    ;jmp $
;-------------------------write your code here---------------------
; Put base address of ldt to ldtr descriptor                      ;
; Load ldt                                                        ;
; Jump to LDT0_Start                                              ;
;                                                                 ;
;-----------------------------------------------------------------
    jmp 0x04:LDT0_Start

LDT0_Start:

    call print_cs_LDT0_Start

    jmp $
```



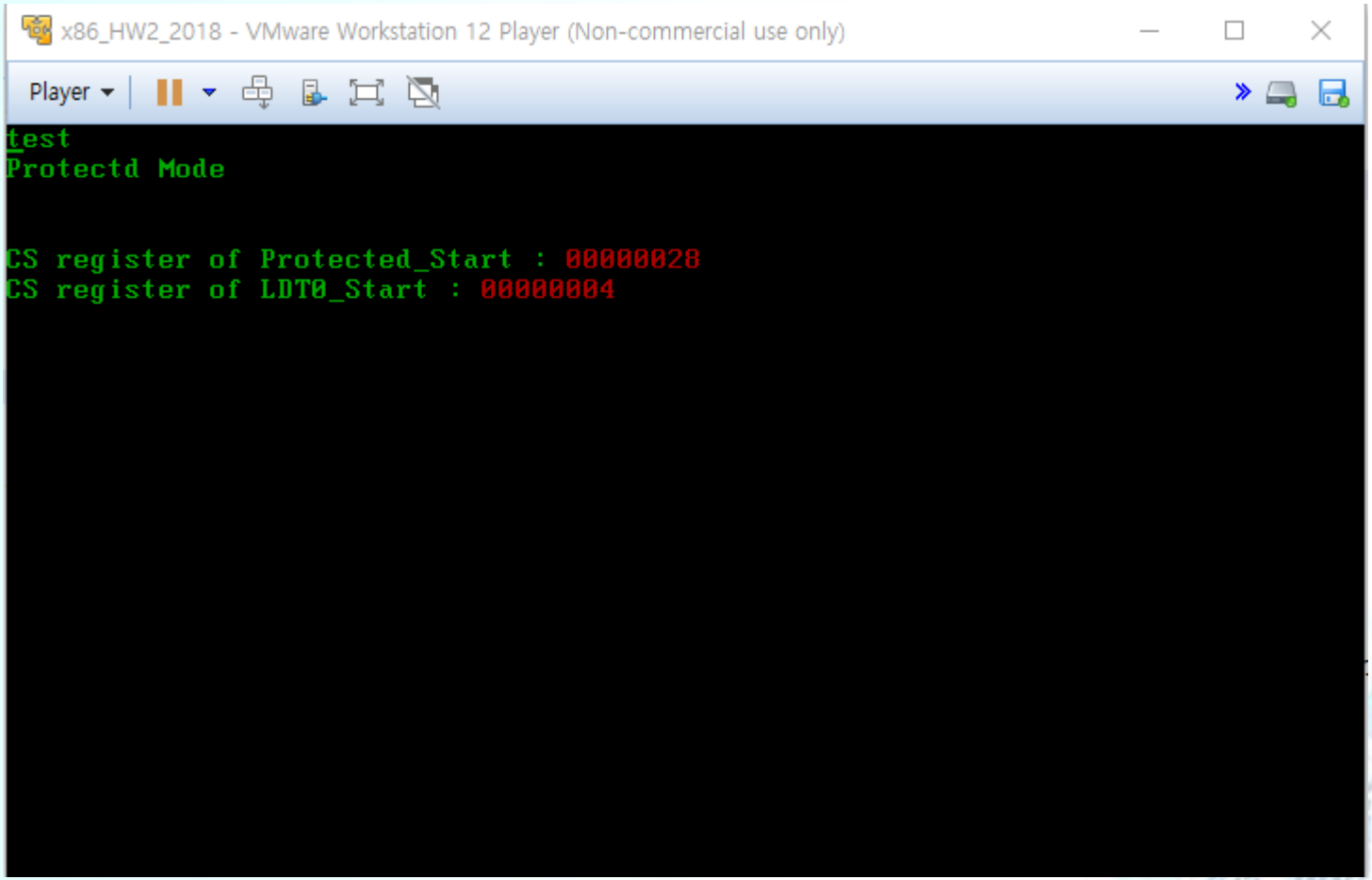
2nd Homework

Initial program



2nd Homework

Result program



2nd Homework

- Time and Place
  - May 24th(Fri) 19:00
  - Semi-conductor building 2 floor computer room
    - 400212, 400202

- How to submit
  - .asm and .bin files
  - I-Campus, until May 24th 18:59
    - format
      - 2010310000_HW2.asm
      - 2010310000_HW2.bin