x86 HW3

2019. 05. 24

Jeon Jae Wook
Sungkyunkwan Univ.

Contents

- Describe about 2nd Homework
- 3rd Homework

Describe about 2nd Homework

- Switch to Protected mode from Real mode
  - Load the address of GDT to GDTR
    - Using lgdt instruction
  - Control Register 0(CR0) setting
    - Protected Enable(PE) bit is set
  - jmp SYS_CODE_SEL:Protected_START
    - Jump to Protected_START

```
;-----------------------------------------------------------------------
    lgdt[gdt_ptr]
;----------------------Write your code here-----------------------------
;                                                                      ;
;Control Register 0(CR0) setting                                       ;
;                                                                      ;
;-----------------------------------------------------------------------
    mov eax, cr0
    or eax, 0x00000001
    mov cr0, eax

    jmp SYS_CODE_SEL:Protected_START     ; jump Protected_START
                                         ; Remove prefetch queue
;-----------------------------------------------------------------------
Protected_START:     ; Protected mode starts
[bits 32]            ; Assembly command
```

Describe about 2nd Homework

Global Descriptor Table & Selector

```
gdt:
;----------------------Write your code here-------------------------
    dw  0           ; limit 15:0
    dw  0           ; base 15:0
    db  0           ; base 23:16
    db  0           ; type
    db  0           ; limit 19:16, flags
    db  0           ; base 31:24
;-------------------------------------------------------------------

SYS_CODE_SEL equ    08h
;----------------------Write your code here-------------------------
gdt1:
    dw  0FFFFh      ; limit 15:0
    dw  00000h      ; base 15:0
    db  0           ; base 23:16
    db  9Ah         ; present, ring 0, code, non-conforming, readable
    db  0cfh        ; limit 19:16, flags
    db  0           ; base 31:24
;-------------------------------------------------------------------


;----------------------Write your code here-------------------------
SYS_DATA_SEL equ    10h
gdt2:
    dw  0FFFFh      ; limit 15:0
    dw  00000h      ; base 23:16
    db  0           ; base 23:16
    db  92h         ; present, ring 0, data, expand-up, writable
    db  0cfh        ; limit 19:16, flags
    db  0           ; base 31:24
;-------------------------------------------------------------------


;----------------------Write your code here-------------------------
Video_SEL   equ 18h
gdt3:
    dw  0FFFFh      ; limit 15:0
    dw  08000h      ; base 23:16
    db  0Bh         ; base 23:16
    db  92h         ; present, ring 0, data, expand-up, writable
    db  40h         ; limit 19:16, flags
    db  00h         ; base 31:24
;-------------------------------------------------------------------
```

Describe about 2nd Homework

- Limit and base address of GDT
  - dw gdt_end – gdt – 1
    - Limit address computation and storage of GDT
  - dd gdt
    - Base address stored in the GDT

```
gdt_end:
;----------------------Write your code here-----------------------
gdt_ptr:
        dw          gdt_end - gdt - 1    ; GDT limit
        dd          gdt      ; linear addr of GDT (set above)
;-----------------------------------------------------------------
```

3rd Homework

- 3rd Homework Describe
  - Make three LDT
    - Make descriptors in GDT
    - Load ldt
  - Control transfer using LDT
    - Far jump
    - Far call / return
    - Call gate descriptor
  - Print strings
    - Print on vmware

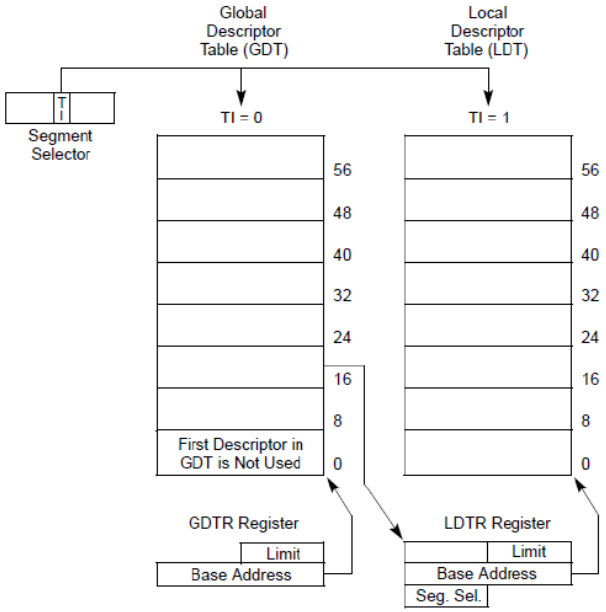
Global Descriptor Table

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		
6	SYS_CODE_SEL_1	Code Segment Descriptor
7		

Local Descriptor Table

- Memory addressing using LDT
  - Make LDTR 1 descriptor in GDT
    - Base address : base address of LDT 1
    - Limit : limit value of LDT 1
    - 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

- **Memory addressing using LDT (Con't)**
  - **Make LDTR 2 descriptor in GDT**
    - **Base address : base address of LDT 2**
    - **Limit : limit value of LDT 2**
    - **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

- Memory addressing using LDT (Con't)
  - Make LDTR 3 descriptor in GDT
    - Base address : base address of LDT 3
    - Limit : limit value of LDT 3
    - 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**
  - 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	Segment Descriptor Registers (Automatically Loaded)		
	15 0			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

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	LDTR1	System Descriptor
5	LDTR2	System Descriptor
6	SYS_CODE_SEL_1	Code Segment Descriptor
7	LDTR3	System Descriptor

Local Descriptor Table

Local Descriptor Table 1

Index	Segment Selector	TYPE
0	LDT1_CODE_SEL_0	Code Segment Descriptor
1	LDT1_CODE_SEL_1	Code Segment Descriptor
2	LDT1_DATA_SEL_0	Data Segment Descriptor

Local Descriptor Table 2

Index	Segment Selector	TYPE
0	LDT2_DATA_SEL_0	Data Segment Descriptor
1	LDT2_CODE_SEL_0	Code Segment Descriptor
2	LDT2_Call_Gate	Call Gate Descriptor
3	LDT2_CODE_SEL_1	Code Segment Descriptor

Local Descriptor Table 3

Index	Segment Selector	TYPE
0	LDT3_CODE_SEL_0	Data Segment Descriptor
1	LDT3_DATA_SEL_0	Code Segment Descriptor

Local Descriptor Table

- **Code Segment Descriptor**
  - 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**
  - 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

JUMP Instruction

- Jump Instruction
  - Far jump
    - Destination is in a different code segment
  - Instructions
    - jmp CS: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

CALL and RET Instruction

- CALL Instruction
  - Far call
    - Destination is in a different code segment
  - Instructions
    - call CS:offset
  - A logical address consisting of
    - A 16-bit segment selector
      - Base address
    - A 32-bit offset
      - EIP ← offset
  - A far call to a code segment at the same privilege level
    - CS ← the new code segment selector and its descriptor
    - EIP ← the offset from the instruction
- RET Instruction
  - retf (far return)

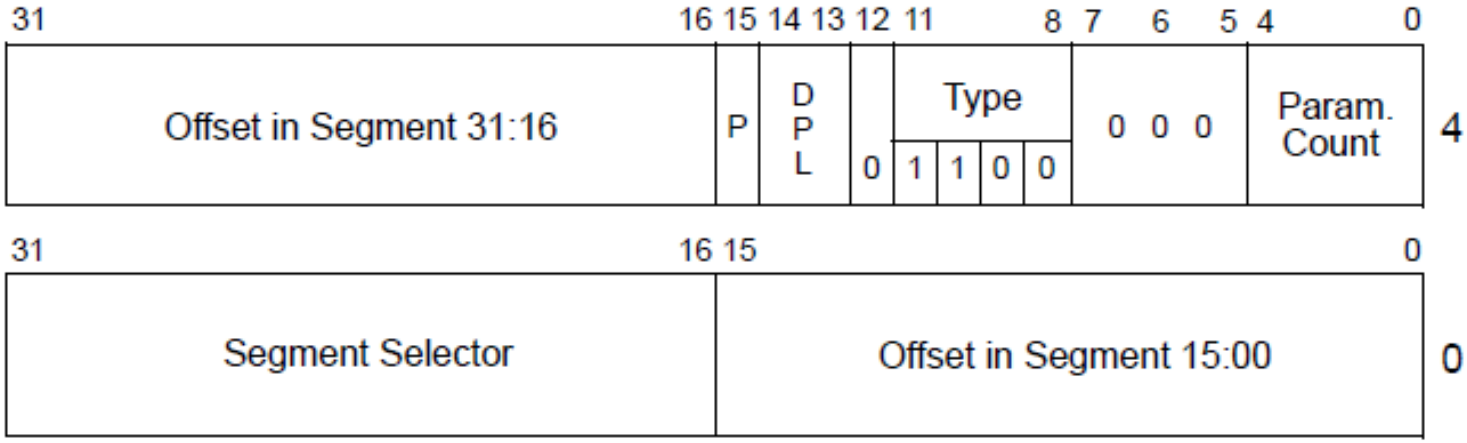
CALL and RET Instruction

- When executing a far call
  - Push CS register on the stack
  - Push EIP register on the stack
  - Begin execution of the called procedure

- When executing a far return
  - Pop EIP register(top of stack value)
  - Pop CS register(top of stack value)
  - Resumes execution of the calling procedure

Call Gate Descriptor

- **A call-gate descriptor**
  - **May reside in the GDT or in an LDT**
  - **Not int the interrupt descriptor table (IDT)**
    - **Other Information**
      - **Descriptor Privilege Level is 0**
      - **Param. Count value is 0**
      - **Gate valid value is 0**



3rd Homework

- **Transfer control (move other code segment)**
  - **Jump**
    - **Protected_START → LDT1_Start**
      - **Using LDT1_CODE_SEL_0 in LDT 1**
    - **LDT1_Start→ LDT2_Start**
      - **Using LDT2_CODE_SEL_1 in LDT 2**
    - **LDT2_Start→ LDT3_Start**
      - **Using LDT3_CODE_SEL_0 in LDT 3**
    - **LDT3_Start→ GDT_Return**
      - **Using SYS_CODE_SEL_1 in GDT**

3rd Homework

- **Transfer control (move other code segment)**
  - **Call / RET**
    - **LDT1_Start → LDT1_Next**
      - **Using LDT1_CODE_SEL_1 in LDT 1**
    - **LDT1_Next→ LDT1_Start**
      - **Using far return instruction**
    - **LDT2_Start → LDT2_Next**
      - **Using Call-gate descriptor**
    - **LDT2_Next→ LDT2_Start**
      - **Using far return instruction**

3rd Homework

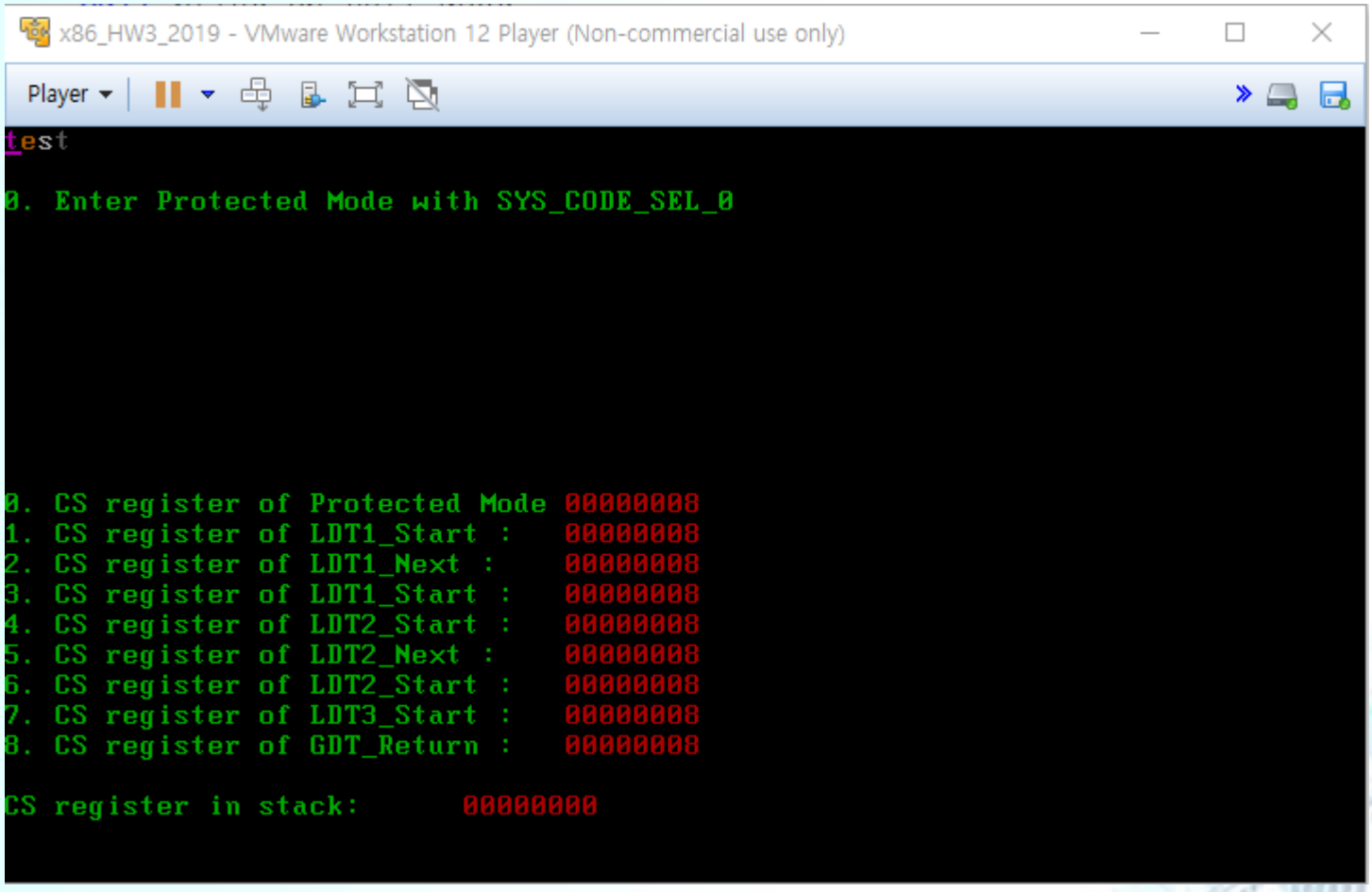
- **Print strings**
  - **Use printf_s**
    - **call Printf_s**
      - **eax register value for variable to print**
      - **edi for position in VMware and bl for property(color)**

3rd Homework

- **Print strings**
  - **Strings for each label**
    - **LDT1_Start**
      - **MSG_LDT1_Start_0**
      - **MSG_LDT1_Start_1**
    - **LDT1_Next**
      - **MSG_LDT1_Next**
    - **LDT2_Start**
      - **MSG_LDT2_Start_0**
      - **MSG_LDT2_Start_1**
    - **LDT2_Next**
      - **MSG_LDT2_Next**
    - **LDT3_Start**
      - **MSG_LDT3_Start**
    - **GDT_Return**
      - **MSG_GDT_Return**

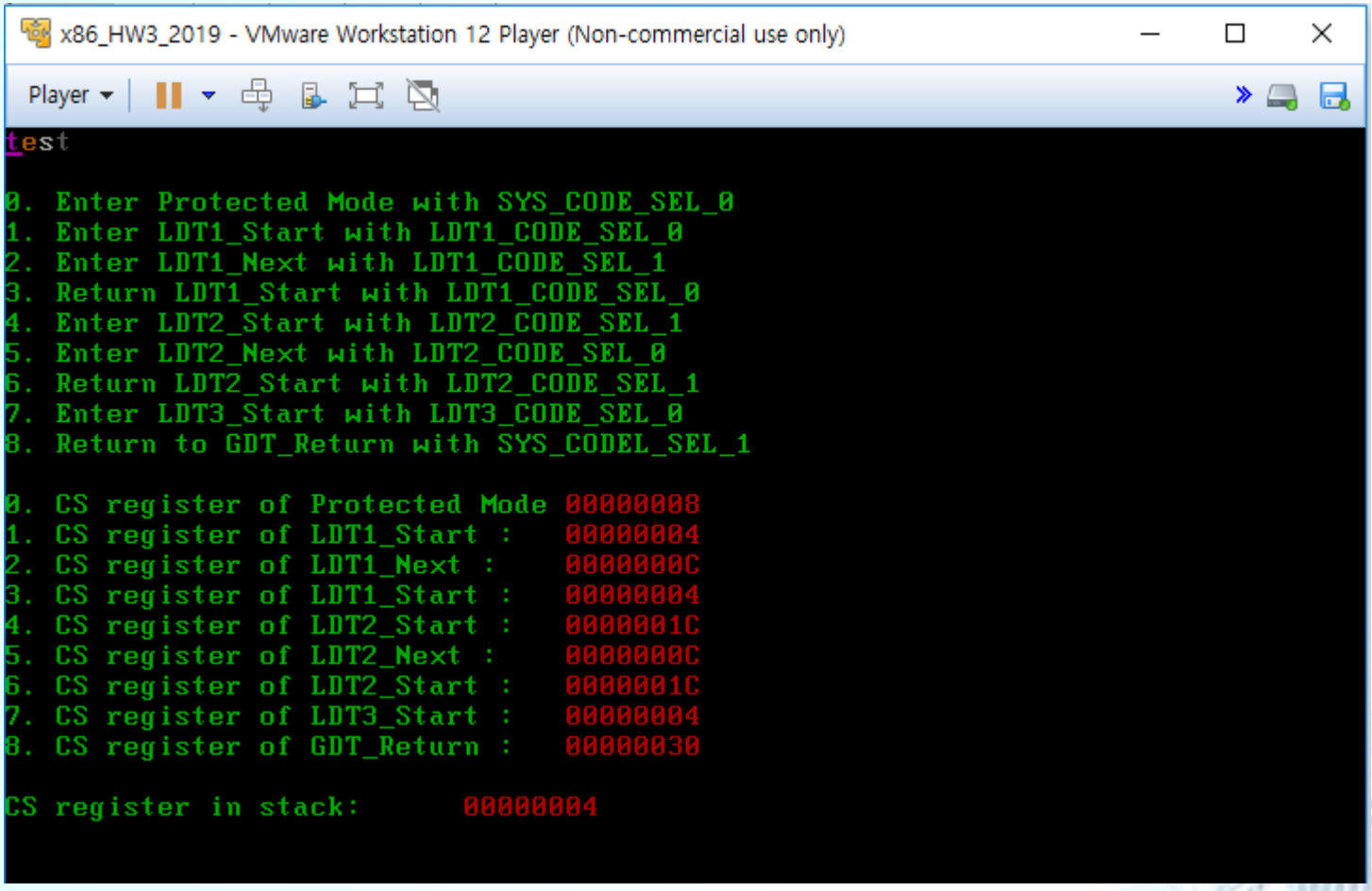
3rd Homework

- **Initial program**



3rd Homework

- **Result program**



3rd Homework

- Time and Place
  - May 31th(Fri) 19:00
  - Semi-conductor building 2 floor computer room
    - 400212, 400202
- How to submit
  - .asm and .bin files
  - I-Campus, until May 31th(Fri) 18:59
    - format
      - 2010310000_HW3.asm
      - 2010310000_HW3.bin