Page No.
Date

Name: Manthan .M. Sonawane
Panel: E Roll No: 17 Batch: E1

MAIOT LAB 9

Problem Statement: Write an ALP to display the contents of GDTR, IDTR, LDTR, TR, MSW

Theory:

A] 1] SGDT <mem>
- Stores the content of global description table register into destination operand
- The destination operand specifies a memory location

2] SIDT <mem>
- Stores the content of interrupt descriptor table register in the destination operand.
- The destination operand specifies of 6-bytes memory location.

3] SLDT <mem>
- Stores the segment selector from the bcd descriptor table register in destination
- destination operand can be a general purpose register or a memory location.

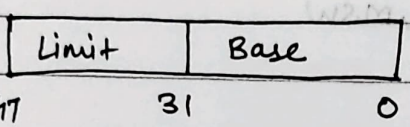
4] STR <mem>
- stores the segment selector from the task register in destination operand.

5] SMSW <mem>

- stores the machine status word (bits through 8 of control register) into destination operand.
- destination operand can be a general purpose register or a memory location

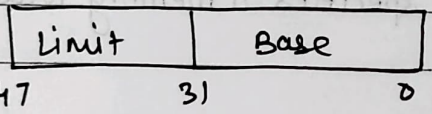
8]

→ GDTR



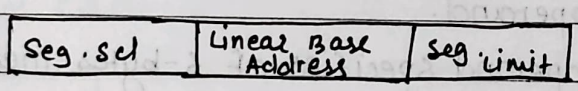
The GDTR hols the 32 bit base address and 16 bit table limit for GDT

→ IDTR



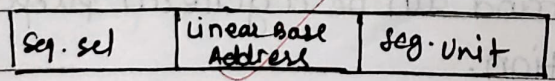
IDTR holds the 32 bits base address and 16 bit table limit for IDT.

→ TR

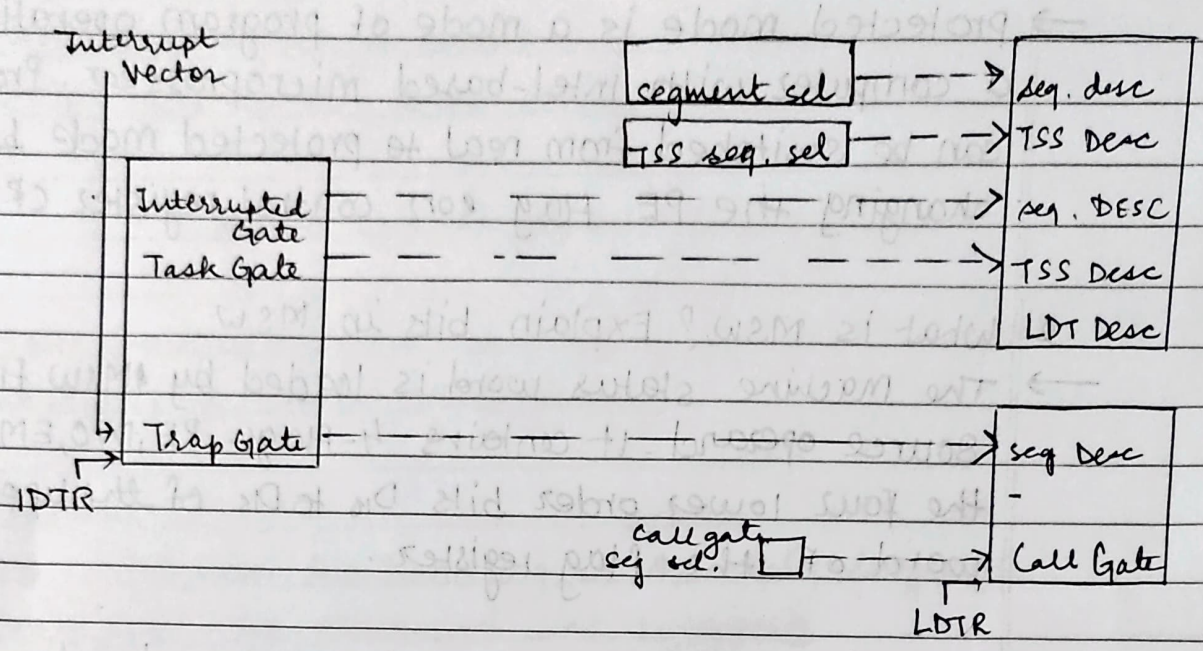


Task register holds 16 bit segment selector, 32 bit base address, 16 bit segment register & descriptor attributes for TSS of current task.

→ LDTR



The LDTR holds the 16 bit segment selector, 32 bit base address, 16 bit segment register & descriptor attributes for LDT.



Algorithm

- Start
- Display message using sys write call
- Read CR0
- Checking PE bit, if 1 = protected mode
- load the number of digits to display
- Rotate no ~~of~~ left by four bits
- convert numbers on ASCII
- Display no from buffer.
- Exit using sys-exit call
- Stop

FAQ's:

1] what is protected mode? How is protected mode secure processor? Using register and which flag.

→ protected mode is a mode of program operation in a computer with Intel-based microprosseor. Processor can be switched from real to protected mode by changing the PE flag on control register CR0.

2 What is MSW? Explain bits in MSW.

→ The machine status word is loaded by LMSW from the source operand. It contains 4-flags - PE, MP, EM, TS of the four lower order bits D_0 to D_3 of the upper word of the flag register.

3. Explain difference b/w real & protected mode.

→

Real Mode	Protected Mode
1. Processor works as 8088/8086	1. Processor works on full capacity
2. Mode holds only on 1MB memory addressing ability	2. Here mode has more than 1MB to 6B memory addressing ability.
3. Handles one task at a time	3. Handles multiple task at a time.
4. Translation is not required	4. Translation is required
5. Does not support memory management.	5. Supports memory management.

```
%macro write 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
section .data
        gmsg db 10,10,"The contents of GDTR are: "
        gmsg_len equ $-gmsg

        lmsg db 10,10,"The contents of LDTR are: "
        lmsg_len equ $-lmsg

        imsg db 10,10,"The contents of IDTR are: "
        imsg_len equ $-imsg

        tmsg db 10,10,"The contents of TR are: "
        tmsg_len equ $-tmsg

        mmsg db 10,10,"The contents of MSW/CR0 are: "
        mmsg_len equ $-mmsg
```

```
        pro db 10,10,"The processor is in protected mode "
        pro_len equ $-pro

        real db 10,10,"The processor is in protected mode "
        real_len equ $-real

        col db ":"
        col_len equ $-col

        nline db 10,10
        nlen equ $-nline

section .bss
        buff resb 4
        gdt1 resb 6
        idt1 resb 6
        ldt1 resw 1
        t1 resb 2
        msw1 resb 4

section .text
```

```
global _start

_start:

        smsw eax

        mov [msw1],eax

        bt eax,0

        jc protected

        write real,real_len

        jmp end

protected:

        write pro,pro_len

        sgdt [gdt1]

        sldt [ldt1]

        sidt [idt1]

        str [t1]


        write gmsg,gmsg_len

        mov bx,[gdt1+4]

        call original_ascii

        mov bx,[gdt1+2]

        call original_ascii

        write col,col_len

        mov bx,[gdt1]
```

```
call original_ascii



write lmsg,lmsg_len

mov bx,[ldt1]

call original_ascii


write lmsg,lmsg_len

mov bx,[idt1+4]

call original_ascii

mov bx,[idt1+2]

call original_ascii

write col,col_len

mov bx,[idt1]

call original_ascii



write tmsg,tmsg_len

mov bx,[t1]

call original_ascii


write mmsg,mmsg_len
```

```
        mov bx,[idt1+4]
        call original_ascii
        mov bx,[idt1+2]
        call original_ascii

end:
        write nline,nlen
        mov rax,60
        mov rdi,0
        mov rsi,0
        mov rdx,0
        syscall

original_ascii:
        mov rax,0
        mov rcx,4
        mov rdi,buff
        up2: rol bx,4
        mov dl,bl
        and dl,0fh
        cmp dl,09h
        jbe down2
```

```
        add dl,07h
down2:  add dl,30h
        mov [rdi],dl
        inc rdi
        loop up2
        write buff,4
    ret
```

OUTPUT