Assignment No

Name: Prajwal Khobragade

SYCOB138

Program:

; Assignment number; Title: Write X86/64 ALP to detect protected mode and display the values of GDTR, LDTR, IDTR, TR and MSW.

Registers; Aim: Write ALP to switch from real mode to protected mode and display the values of GDTR, LDTR, IDTR, TR and MSW Registers.

; Program to retrieve and display information about processor mode and registers

```
section .data
   ; Variables and messages declaration
   ano        db 10, "Assignment No.", 10
   ano_len    equ $ - ano
   nline      db 10, 10
   nline_len  equ $ - nline
   space      db " "
   colon      db ":"
   rmsg       db 10, "Processor in real mode"
   rmsg_len   equ $ - rmsg
   pmsg       db 10, "Processor in protected mode"
   pmsg_len   equ $ - pmsg
   gmsg       db 10, "GDTR (Global Descriptor Table Register) : "
   gmsg_len   equ $ - gmsg
   imsg       db 10, "IDTR (Interrupt Descriptor Table Register) : "
   imsg_len   equ $ - imsg
   lmsg       db 10, "LDTR (Local Descriptor Table Register) : "
   lmsg_len   equ $ - lmsg
   tmsg       db 10, "TR (Task Register) : "
   tmsg_len   equ $ - tmsg
   mmsg       db 10, "MSW (Machine Status Word) : "
   mmsg_len   equ $ - mmsg

section .bss
   ; Reserved space for variables
   GDTR       resw 3
   IDTR       resw 3
```

```
    LDTR       resw 1
    TR         resw 1
    MSW        resw 1
    char_ans    resb 4  ; Uninitialized variable for display procedure

section .text
    global _start

_start:
    ; Print assignment number
    Print ano, ano_len

    ; Check processor mode
    SMSW [MSW]            ; Load value into MSW
    mov rax, [MSW]        ; Load value into rax from MSW
    ror rax, 1            ; Load LSB into the carry
    jc p_mode             ; If LSB set, jump to protected mode
    Print rmsg, rmsg_len   ; Print real mode message
    jmp next              ; Continue to next section
p_mode:
    Print pmsg, pmsg_len   ; Print protected mode message
next:
    ; Retrieve and print information about various processor registers
    SGDT [GDTR]           ; Store GDT value in GDTR (48 bits)
    SIDT [IDTR]           ; Store IDT value in IDTR (48 bits)
    SLDT [LDTR]           ; Store LDT value in LDTR (16 bits)
    STR [TR]              ; Store TR value in TR (16 bits)
    SMSW [MSW]            ; Load MSW value into MSW
    Print gmsg, gmsg_len   ; Print GDTR message
    mov ax, [GDTR+4]      ; Load content of 5th and 6th GDTR location into ax
    call display          ; Display content of 5th and 6th
    Print colon, 1        ; Print colon for formatting
    mov ax, [GDTR+2]      ; Load content of 3rd and 4th GDTR location into ax
    call display          ; Display content of 3rd and 4th
```

```
    Print colon, 1        ; Print colon for formatting
    mov ax, [GDTR+0]      ; Load content of 1st and 2nd GDTR location into ax
    call display          ; Display content of 1st and 2nd
    Print imsg, imsg_len   ; Print IDTR message
    mov ax, [IDTR+4]      ; Load content of 5th and 6th IDTR location into ax
    call display          ; Display content of 5th and 6th
    Print colon, 1        ; Print colon for formatting
    mov ax, [IDTR+2]      ; Load content of 3rd and 4th IDTR location into ax
    call display          ; Display content of 3rd and 4th
    Print colon, 1        ; Print colon for formatting
    mov ax, [IDTR+0]      ; Load content of 1st and 2nd IDTR location into ax
    call display          ; Display content of 1st and 2nd
    Print lmsg, lmsg_len   ; Print LDTR message
    mov ax, [LDTR]        ; Load content of LDTR into ax
    call display          ; Display the content
    Print tmsg, tmsg_len   ; Print TR message
    mov ax, [TR]          ; Load content of TR into ax
    call display          ; Display the content
    Print mmsg, mmsg_len   ; Print MSW message
    mov ax, [MSW]         ; Load content of MSW into ax
    call display          ; Display the content

    End                   ; End the program

; Display procedure
display:
    mov rsi, char_ans + 3  ; Point rsi to char_ans + 3
    mov rcx, 4             ; Set loop counter to 4
    mov rbx, 16            ; Set divisor to 16
next_num:
    xor rdx, rdx           ; Initialize rdx with 0
    div rbx                ; Divide: rdx = rax / rbx
    cmp dl, 09H            ; Compare remainder with 9
    jbe add30              ; Jump if below or equal to 9
```

```
    add dl, 07H          ; Adjust for hexadecimal conversion
add30:
    add dl, 30H          ; Convert to ASCII character
    mov [rsi], dl        ; Store character
    dec rsi              ; Decrement pointer
    dec rcx              ; Decrement loop counter
    jnz next_num         ; Jump if not zero
    Print char_ans, 4    ; Print the number in char_ans
    ret                  ; Return

; Macro for printing messages
Print:
    mov rax, 1           ; System call for writing to stdout
    mov rdi, 1           ; File descriptor: stdout
    mov rsi, rdi         ; Pointer to message to print
    mov rdx, rdi         ; Length of message to print
    syscall              ; Invoke syscall to print message
    ret

; Macro for ending the program
End:
    mov rax, 60          ; System call for exit
    xor rdi, rdi         ; Exit code 0
    syscall              ; Invoke syscall to end program
```

Output

```
Assignment No. Processor in protected mode
GDTR (Global Discriptor Table Register) : 0003:C000:007F
IDTR (Interrupt Discriptor Table Register) : 0000:0000:0FFF
LDTR (Local Discriptor Table Register) : 0000
TR (Task Register) : 0040 MSW (Machine Status Word ) : 0033
```