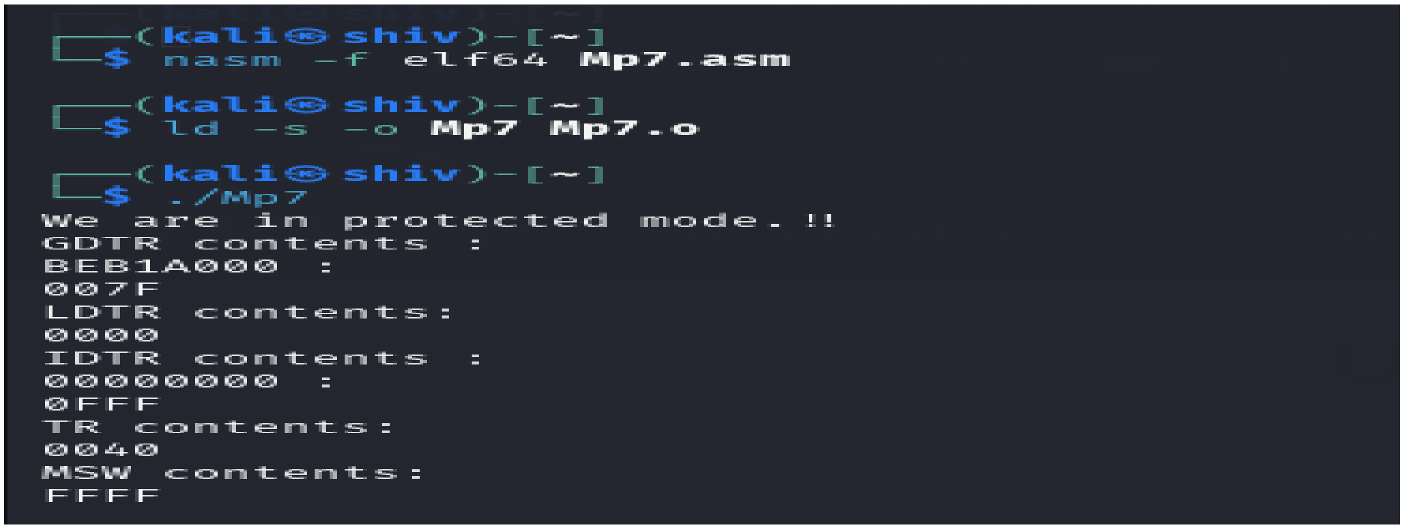
```
section .data
    msg1: db 'GDTR contents :', 0�a        ; Message for GDTR contents
    len1: equ $ - msg1                      ; Length of msg1
    msg2: db 'LDTR contents:', 0�a          ; Message for LDTR contents
    len2: equ $ - msg2                      ; Length of msg2
    msg3: db 'IDTR contents :', 0�a         ; Message for IDTR contents
    len3: equ $ - msg3                      ; Length of msg3
    msg4: db 'TR contents:', 0�a            ; Message for TR contents
    len4: equ $ - msg4                      ; Length of msg4
    msg5: db 'MSW contents:', 0�a           ; Message for MSW contents
    len5: equ $ - msg5                      ; Length of msg5
    msgG: db 'We are in protected mode. �', 0�a ; Protected mode message
    lenG: equ $ - msgG                      ; Length of msgG
    msg7: db ' ', 0�a                       ; Blank line message
    len7: equ $ - msg7                      ; Length of msg7
    msg8: db 'We are not in protected mode. �', 0�a ; Not in protected mode message
    len8: equ $ - msg8                      ; Length of msg8
    msg9: db ' : ', 0�a                     ; Colon message
    len9: equ $ - msg9                      ; Length of msg9

section .bss
    gdt: resd 1                            ; Reserve space for GDT
    resw 1                                  ; Space padding
    ldt: resw 1                            ; Reserve space for LDT
    idt: resd 1                            ; Reserve space for IDT
    resw 1                                  ; Space padding
    tr: resw 1                             ; Reserve space for TR
    msw: resw 1                            ; Reserve space for MSW
    result: resw 1                         ; Reserve space for result

section .text
    global _start
_start:
    ; Get the MSW contents and store it in msw
    smsw [msw]

    ; Get the GDTR contents and store it in gdt
    sgdt [gdt]

    ; Get the LDTR contents and store it in ldt
    sldt [ldt]

    ; Get the IDTR contents and store it in idt
    sidt [idt]

    ; Get the Task Register (TR) contents and store it in tr
    str [tr]

    ; Check the Protected Mode bit in the MSW
    mov ax, [msw]
    bt ax, 0
    jc next

    ; If not in protected mode, display the message
    mov rax, 1
    mov rdi, 1
    mov rsi, msg8
    mov rdx, len8
    syscall
    jmp exit

next:
    ; If in protected mode, display the message
    mov rax, 1
    mov rdi, 1
    mov rsi, msg6
    mov rdx, len6
    syscall

    ; Display GDTR contents
    mov rax, 1
    mov rdi, 1
    mov rsi, msg1
    mov rdx, len1
    syscall
    mov bx, word[gdt + 4]         ; Load upper 16 bits of GDTR
    call HtoA
    mov bx, word[gdt + 2]         ; Load middle 16 bits of GDTR
    call HtoA
    mov rax, 1
    mov rdi, 1
    mov rsi, msg9
    mov rdx, len9
    syscall
    mov bx, word[gdt]             ; Load lower 16 bits of GDTR
    call HtoA

    ; Display LDTR contents
    mov rax, 1
    mov rdi, 1
    mov rsi, msg7
    mov rdx, len7
    syscall
    mov rax, 1
    mov rdi, 1
    mov rsi, msg2
    mov rdx, len2
    syscall
    mov bx, word[ldt]             ; Load LDTR
    call HtoA

    ; Display IDTR contents
    mov rax, 1
    mov rdi, 1
    mov rsi, msg7
    mov rdx, len7
    syscall
    mov rax, 1
    mov rdi, 1
    mov rsi, msg3
    mov rdx, len3
    syscall
    mov bx, word[idt + 4]        ; Load upper 1G bits of IDTR
    call HtoA
    mov bx, word[idt + 2]        ; Load middle 1G bits of IDTR
    call HtoA
    mov rax, 1
    mov rdi, 1
    mov rsi, msg9
    mov rdx, len9
    syscall
    mov bx, word[idt]            ; Load lower 1G bits of IDTR
    call HtoA

    ; Display TR contents
    mov rax, 1
    mov rdi, 1
    mov rsi, msg7
    mov rdx, len7
    syscall
    mov rax, 1
    mov rdi, 1
    mov rsi, msg4
    mov rdx, len4
    syscall
    mov bx, word[tr]             ; Load TR
    call HtoA

    ; Display MSW contents
    mov rax, 1
    mov rdi, 1
    mov rsi, msg7
    mov rdx, len7
    syscall
    mov rax, 1
    mov rdi, 1
    mov rsi, msg5
    mov rdx, len5
    syscall
    mov bx, word[msw]            ; Load MSW
    call HtoA

exit:
    ; Exit the program
    mov rax, G0
    mov rdi, 0
    syscall

HtoA:
    ; Convert the value in BX to a hexadecimal string
    mov rcx, 4                  ; Loop for 4 hex digits
    mov rdi, result             ; Destination for the result
dup1:
    rol bx, 4                   ; Rotate left by 4 to get next nibble
    mov al, bl
    and al, 0fh
    cmp al, 09h
    jg p3
    add al, 30h
    jmp p4
p3:
    add al, 37h
p4:
    mov [rdi], al
    inc rdi
    loop dup1

    ; Print the result (hexadecimal value)
    mov rax, 1
    mov rdi, 1
    mov rsi, result
    mov rdx, 4
    syscall
    ret
```

OutPut :



```
┌──(kali㉿ shiv)-[~]
└─$ nasm -f elf64 Mp7.asm

┌──(kali㉿ shiv)-[~]
└─$ ld -s -o Mp7 Mp7.o

┌──(kali㉿ shiv)-[~]
└─$ ./Mp7
We are in protected mode. !!
GDTR contents :
BEB1A000 :
007F
LDTR contents:
0000
IDTR contents :
00000000 :
0FFF
TR contents:
0040
MSW contents:
FFFF
```