

```

1  section .data
2      msg1: db 'GDTR contents :', 0?a ; Message for GDTR contents
3      len1: equ $ - msg1 ; Length of msg1
4      msg2: db 'LDTR contents:', 0?a ; Message for LDTR contents
5      len2: equ $ - msg2 ; Length of msg2
6      msg3: db 'IDTR contents :', 0?a ; Message for IDTR contents
7      len3: equ $ - msg3 ; Length of msg3
8      msg4: db 'TR contents:', 0?a ; Message for TR contents
9      len4: equ $ - msg4 ; Length of msg4
10     msg5: db 'MSW contents:', 0?a ; Message for MSW contents
11     len5: equ $ - msg5 ; Length of msg5
12     msgG: db 'We are in protected mode. ?', 0?a ; Protected mode message
13     lenG: equ $ - msgG ; Length of msgG
14     msg7: db ' ', 0?a ; Blank line message
15     len7: equ $ - msg7 ; Length of msg7
16     msg8: db 'We are not in protected mode. ?', 0?a ; Not in protected mode
17     message
18     len8: equ $ - msg8 ; Length of msg8
19     msg9: db ' : ', 0?a ; Colon message
20     len9: equ $ - msg9 ; Length of msg9
21
22     section .bss
23     gdt: resd 1 ; Reserve space for GDT
24     resw 1 ; Space padding
25     ldt: resw 1 ; Reserve space for LDT
26     idt: resd 1 ; Reserve space for IDT
27     resw 1 ; Space padding
28     tr: resw 1 ; Reserve space for TR
29     msw: resw 1 ; Reserve space for MSW
30     result: resw 1 ; Reserve space for result
31
32     section .text
33     global _start
34     _start:
35     ; Get the MSW contents and store it in msw
36     smsw [msw]
37
38     ; Get the GDTR contents and store it in gdt
39     sgdt [gdt]
40
41     ; Get the LDTR contents and store it in ldt
42     sltd [ldt]
43
44     ; Get the IDTR contents and store it in idt
45     sidt [idt]
46
47     ; Get the Task Register (TR) contents and store it in tr
48     str [tr]

```

```

49      ; Check the Protected Mode bit in the MSW
50      mov ax, [msw]
51      bt ax, 0
52      jc next
53
54      ; If not in protected mode, display the message
55      mov rax, 1
56      mov rdi, 1
57      mov rsi, msg8
58      mov rdx, len8
59      syscall
60      jmp exit
61
62 next:
63      ; If in protected mode, display the message
64      mov rax, 1
65      mov rdi, 1
66      mov rsi, msgG
67      mov rdx, lenG
68      syscall
69
70      ; Display GDTR contents
71      mov rax, 1
72      mov rdi, 1
73      mov rsi, msg1
74      mov rdx, len1
75      syscall
76      mov bx, word[gdt + 4]          ; Load upper 1G bits of GDTR
77      call HtoA
78      mov bx, word[gdt + 2]          ; Load middle 1G bits of GDTR
79      call HtoA
80      mov rax, 1
81      mov rdi, 1
82      mov rsi, msg9
83      mov rdx, len9
84      syscall
85      mov bx, word[gdt]              ; Load lower 1G bits of GDTR
86      call HtoA
87
88      ; Display LDTR contents
89      mov rax, 1
90      mov rdi, 1
91      mov rsi, msg7
92      mov rdx, len7
93      syscall
94      mov rax, 1
95      mov rdi, 1
96      mov rsi, msg2
97      mov rdx, len2
98      syscall
99      mov bx, word[ldt]              ; Load LDTR

```

```

100     call HtoA
101
102     ; Display IDTR contents
103     mov rax, 1
104     mov rdi, 1
105     mov rsi, msg7
106     mov rdx, len7
107     syscall
108     mov rax, 1
109     mov rdi, 1
110     mov rsi, msg3
111     mov rdx, len3
112     syscall
113     mov bx, word[idt + 4]           ; Load upper 1G bits of IDTR
114     call HtoA
115     mov bx, word[idt + 2]           ; Load middle 1G bits of IDTR
116     call HtoA
117     mov rax, 1
118     mov rdi, 1
119     mov rsi, msg9
120     mov rdx, len9
121     syscall
122     mov bx, word[idt]               ; Load lower 1G bits of IDTR
123     call HtoA
124
125     ; Display TR contents
126     mov rax, 1
127     mov rdi, 1
128     mov rsi, msg7
129     mov rdx, len7
130     syscall
131     mov rax, 1
132     mov rdi, 1
133     mov rsi, msg4
134     mov rdx, len4
135     syscall
136     mov bx, word[tr]               ; Load TR
137     call HtoA
138
139     ; Display MSW contents
140     mov rax, 1
141     mov rdi, 1
142     mov rsi, msg7
143     mov rdx, len7
144     syscall
145     mov rax, 1
146     mov rdi, 1
147     mov rsi, msg5
148     mov rdx, len5
149     syscall
150     mov bx, word[msw]             ; Load MSW

```

```

151      call HtoA
152
153  exit:
154      ; Exit the program
155      mov rax, G0
156      mov rdi, 0
157      syscall
158
159  HtoA:
160      ; Convert the value in BX to a hexadecimal string
161      mov rcx, 4                ; Loop for 4 hex digits
162      mov rdi, result          ; Destination for the result
163  dup1:
164      rol bx, 4                ; Rotate left by 4 to get next nibble
165      mov al, bl
166      and al, 0fh
167      cmp al, 09h
168      jg p3
169      add al, 30h
170      jmp p4
171  p3:
172      add al, 37h
173  p4:
174      mov [rdi], al
175      inc rdi
176      loop dup1
177
178      ; Print the result (hexadecimal value)
179      mov rax, 1
180      mov rdi, 1
181      mov rsi, result
182      mov rdx, 4
183      syscall
184      ret
185

```

## 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

```