```
section .data
      rmodemsg db 10,'Processor is in Real Mode'
      rmsg_len:equ $-rmodemsg

      pmodemsg db 10,'Processor is in Protected Mode'
      pmsg_len:equ $-pmodemsg

      gdtmsg db 10,'GDT Contents are::'
      gmsg_len:equ $-gdtmsg

      ldtmsg db 10,'LDT Contents are::'
      lmsg_len:equ $-ldtmsg

      idtmsg db 10,'IDT Contents are::'
      imsg_len:equ $-idtmsg

      trmsg db 10,'Task Register Contents are::'
      tmsg_len: equ $-trmsg

      mswmsg db 10,'Machine Status Word::'
      mmsg_len:equ $-mswmsg

      colmsg db ':'

      nwline db 10

section .bss
      gdt resd 1
          resw 1
      ldt resw 1
      idt resd 1
          resw 1
      tr  resw 1

      cr0_data resd 1

      dnum_buff resb 04

%macro print 2
      mov rax,01
      mov rdi,01
      mov rsi,%1
      mov rdx,%2
      syscall
%endmacro

section .text
global _start
_start:
      smsw eax          ;Reading CR0. As MSW is 32-bit cannot use RAX
register.

      mov [cr0_data],rax
```

```
      bt rax,1          ;Checking PE bit, if 1=Protected Mode, else Real
Mode
      jc prmode
      print rmodemsg,rmsg_len
      jmp nxt1

prmode:     print pmodemsg,pmsg_len

nxt1: sgdt [gdt]
      sldt [ldt]
      sidt [idt]
      str [tr]
      print gdtmsg,gmsg_len

      mov bx,[gdt+4]
      call print_num

      mov bx,[gdt+2]
      call print_num

      print colmsg,1

      mov bx,[gdt]
      call print_num

      print ldtmsg,lmsg_len
      mov bx,[ldt]
      call print_num

      print idtmsg,imsg_len

      mov bx,[idt+4]
      call print_num

      mov bx,[idt+2]
      call print_num

      print colmsg,1

      mov bx,[idt]
      call print_num

      print trmsg,tmsg_len

      mov bx,[tr]
      call print_num

      print mswmsg,mmsg_len

      mov bx,[cr0_data+2]
      call print_num
```

```
      mov bx,[cr0_data]
      call print_num

      print nwline,1


exit: mov rax,60
      xor rdi,rdi
      syscall

print_num:
      mov rsi,dnum_buff ;point esi to buffer

      mov rcx,04        ;load number of digits to printlay

up1:
      rol bx,4          ;rotate number left by four bits
      mov dl,bl         ;move lower byte in dl
      and dl,0fh        ;mask upper digit of byte in dl
      add dl,30h        ;add 30h to calculate ASCII code
      cmp dl,39h        ;compare with 39h
      jbe skip1         ;if less than 39h skip adding 07 more
      add dl,07h        ;else add 07
skip1:
      mov [rsi],dl            ;store ASCII code in buffer
      inc rsi                 ;point to next byte
      loop up1          ;decrement the count of digits to printlay
                        ;if not zero jump to repeat

      print dnum_buff,4 ;printlay the number from buffer

      ret
```