Roll No. - 2
AARON PHILIP
1032210163

MAIOT ASSIGNMENT 9

PROBLEM STATEMENT:
Write an ALP to display contents of GDTR, IDTR, LDTR, TR & MSW

OBJECTIVES:
① Understand the concept of protected mode.
② Understand the values of GDTR, IDTR, LDTR, TR & MSW Registers

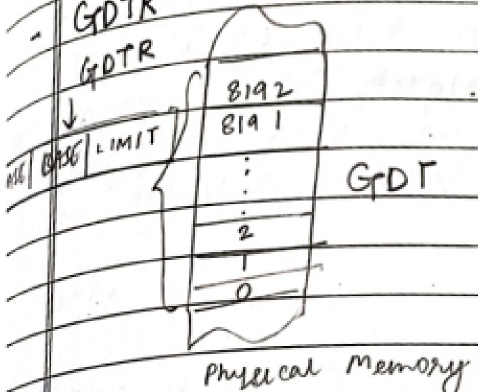
THEORY:

Explain the following instructions used to read the contents of the respective registers
① SGDT (Store Global Descriptor table)
. It is an x86 instruction used to store the base address & limit of the Global Descriptor Table (GDT). The GDT is a data structure used by the x86 architecture to define memory segments.
② SIDT (Store Interrupt Descriptor Table)
It is an x86 instruction used to store the base address & limit of the Instruction Descriptor Table (IDT) in a specified memory location.
③ SLDT (Store Local Descriptor Table)
It is an x86 instruction used to store the segment selector of the Local Descriptor Table (LDT in a specified register

(4) STR (Store Task Register)
It is an x86 instruction used to store the current Task's TSS (Task State Segment) selector in a specified register.

(5) SMSW (Store Machine Status Word)
It is an x86 instruction used to store the contents of the Machine status word (MSW) in a specified register.

- GDTR



ALGORITHM

1) declare & initialize necessary variables for storing messages & register values
2) Check current processor mode by using SMSW instruction & testing 0th bit.
3) Display the appropriate message based on the processor mode (real / protected)
4) Retrieve the values of GDTR, LDTR, IDTR & TR registers using their respective instructions

FOR EDUCATIONAL USE

5) convert the register values to their original ASCII representatn using subroutine.
6) Display the converted register values.
7) Terminate the program

PLATFORM
NASM (Netwide Assembler)
gedit (Linux Open-source Ubuntu)

CONCLUSION
Thus implemented the program using assembly language to display the contents of System registers used in protected mode memory management & MSW.

FAQs

Q1) What is protected mode? How does the processor switch from real to protected mode?
1) Protected mode is a mode of operation in the x86 architecture that provides hardware-based memory protection, enabling multiple programs to run concurrently without interfering with each others memory spaces.
The transition from real mode to protected mode is initiated by setting the PE (Protection Enable) bit in the control register CR0 using the instruction:
MOV CR0, reg.

Q2) What is MSW? Explain bits present in MSW.

A) MSW stands for Machine Status Word, which is a 16-bit register in the x86 architecture that contains status flags & control bits related to file operation.

1) MP (Monitor Coprocessor) Bit:
- Indicates whether the processor is capable of monitoring the coprocessor for errors.

2) EM (Emulate) Bit:
controls whether the processor emulates OR traps instructions.

3) TS (Task Switched) Bit:
This bit is set when the processor switches to a different task / context.

3) Explain the difference between real address & protected mode.

Real mode is a legacy operating mode of the x86 processor that provides direct access to the system resources & memory without any memory protection / privilege levels.

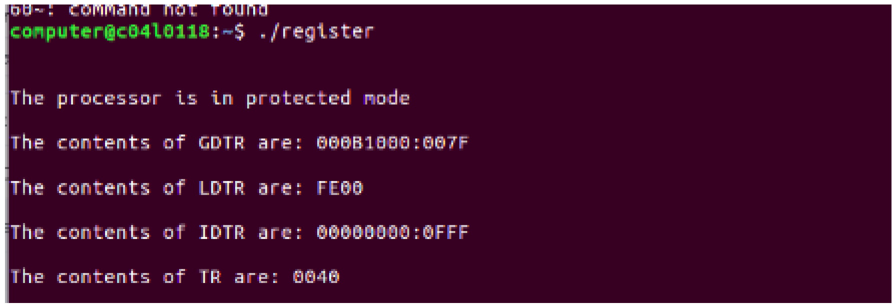
In contrast, protected mode provides memory protection, multitasking & advanced interrupt handling, making it the preferred mode of operation for modern operating systems.

CODE :
```
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 real 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

%macro display 4
mov rax,%1
mov rdi,%2
mov rsi,%3
mov rdx,%4
```

```
syscall
%endmacro

section .text
global _start

_start:
	smsw eax
	mov [msw1],eax
	bt eax,0
	jc protected
	display 1,1,real,real_len  ;display real mode
	jmp end

protected:
display 1,1,pro,pro_len  ;display protected mode

sgdt[gdt1]
sldt[ldt1]
sidt[idt1]
str[t1]

display 1,1,gmsg,gmsg_len
mov bx,[gdt1+4]
call original_ascii
mov bx,[gdt1+2]
call original_ascii
display 1,1,col,col_len
mov bx,[gdt1]
call original_ascii

display 1,1,lmsg,lmsg_len
mov bx,[ldt1]
call original_ascii

display 1,1,imsg,imsg_len
mov bx,[idt1+4]
call original_ascii
mov bx,[idt1+2]
call original_ascii
display 1,1,col,col_len
mov bx,[idt1]
call original_ascii
```

```
display 1,1,tmsg,tmsg_len
mov bx,[t1]
call original_ascii

display 1,1,mmsg,mmsg_len
mov bx,[msw1+2]
call original_ascii
mov bx,[msw1]
call original_ascii

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
          display 1,1,buff,4
ret

end:    display 1,1,nline,nlen
     display 60,0,0,0
```

OUTPUT :



```
60~: command not found
computer@c04l0118:~$ ./register


The processor is in protected mode

The contents of GDTR are: 000B1000:007F

The contents of LDTR are: FE00

The contents of IDTR are: 00000000:0FFF

The contents of TR are: 0040
```