# Propose Method to Access Protected Mode of Windows Operating System

***Enas F. Aziz***
***Department of Computer Systems, Technical Institute_ Kirkuk***
Accepted:2011/4/26, Received: 2010/11/1

## Abstract

Current day computer systems allow multiple programs to be loaded into memory and to be executed concurrently. This evolution required protection among those various programs. To ensure this protection, operating systems provide protected mode that contains descriptor tables that in its turn, control access to memory segment.

In the Intel Architecture, and more precisely in protected mode, most of the memory management and Interrupt Service Routines are controlled through tables of descriptors. Each descriptor stores information about a single object (e.g. a service routine, a task, a chunk of code or data, whatever) the CPU might need at some time.

In this research, these tables with their contents are studied and given a method to know the contents of these tables and to study the behavior of the O.S, In other words, access to the protected status and work with printing the contents of some special registers that cannot be accessible only within the protected mode (Ring 0).

The program in this research is written in Assembly language (MASM version 7) and tested under Windows Me. The program executed with 32-bit microprocessor, because it contains instruction that can deal with the special registers (GDTR & LDTR) that contains addresses of descriptor tables.

## Introduction

The purpose of protected mode is not to protect your program. The purpose is to protect everyone else (including the operating system) from your program. Protected mode has a number of features designed to enhance an operating system's control over application software, in order to increase security and system stability.

This research tries to reach GDT and LDT in protected mode and locate for empty entry in the LDT, especially entry zero and put inside it the offset address of new routine that includes reading the contents of CR0-CR3 and DR0-DR3 that cannot be accessible unless by this condition. This research executed under Windows Me. (Sreyh, 2004)

## Privilege Rings

The processor provides four levels of privilege called Privilege Rings. Windows uses only two of the privilege levels. The operating system supervisor runs in ring 0. Ring-Zero code can alter any location in memory and any processor register. Application software runs in ring 3. Ring 3 programs cannot access system control registers, nor can they read or write to memory areas the operating system designated as protected. An Intel processor generates the address of a memory operand by combining a segment register with offset values held in one or two registers. Processors compatible with the Intel 80386 through Pentium Pro offer several modes of operation **(**real, V86, and protected**)** modes (Oney, 1996).

## Protected mode

In protected mode the segment part is replaced by 16 bit selector, the 13 upper bits (bit 3 to bit 15) of the selector contains the index of an entry inside a descriptor table. The lowest two bits define the privilege of the request, from 0 to 3 where 0 has the highest priority and 3 the lowest. The remainder bit specifies if the operation is against the GDT or LDT. Each entry contains:-

- the real linear address of the segment
- a limit value for the segment size
- some attribute bits (flags) (Wikipedia**,2010)

Descriptor is chosen from the descriptor table by the segment register. Figure (1) shows segment registers. The 13-bit selector chooses one of the 8192 descriptor from the descriptor table. The TI bit selects either the global descriptor table (TI=0) or the local descriptor table (TI=1).the requested privilege level (RPL) requests the access privilege level of a memory segment. The highest privilege level is 11. If the request privilege level matches or is higher in priority than the privilege level set by the access rights byte, access is granted .For example, if the requested privilege level is 10 and the access rights byte sets the segment privilege level at 11, access is granted because 10 is higher in priority than privilege 11 (Brey, 1997) (Kaplan, 1997-2010).

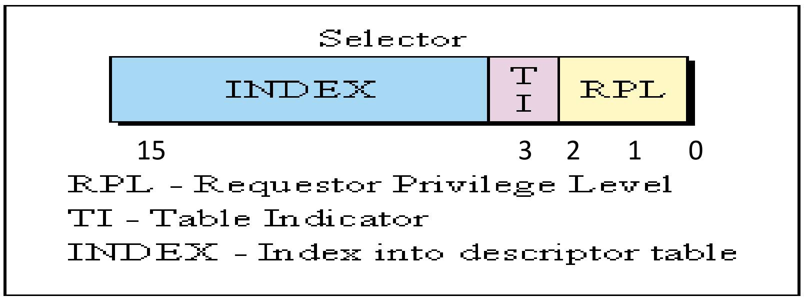
Selector
INDEX
T I
RPL
15
3
2
1
0
RPL - Requestor Privilege Level
TI - Table Indicator
INDEX - Index into descriptor table

**Fig. (1): contents of segment register during protected mode of the 80286 through Pentium Pro**

## Tables in Protected mode

In protected mode, the OS build several tables in the system, these tables are used to store information about processes (Solomon, 1998).These tables called: - 1- Interrupt Descriptor table (IDT)
2- Global Descriptor table (GDT)
3- Local Descriptor table (LDT)

Each table is defined as a (size, linear address) to the CPU through the LIDT, LGDT, LLDT instructions respectively. The IDT is used for descriptors of interrupt Handlers, only the GDT and LDT can hold segment descriptors, as shown in figure (2) (Kaplan,1997-2010). Every 8-byte entry in the GDT is a descriptor, but these can be Task State Segment (TSS) descriptors, Local Descriptor Table (LDT) descriptors (Wikipedia***, 2010).

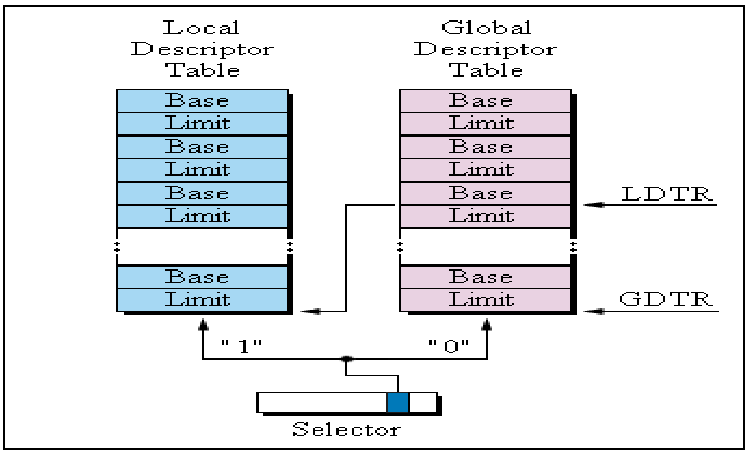

**Fig. (2): Descriptor tables**

The locations of these two tables inside two special registers, the Global Descriptor Table Register (GDTR) and the Interrupt Descriptor Table Register (IDTR). The GDTR and IDTR both use a 48-bit format, containing the 32-bit base address of the table and the 16-bit limit. Each table can contain up to 64KB or 8192 descriptor. Each descriptor in GDT is 64 bits long and contains many different fields. When the system is multitasking, all tasks share the GDT. This is also true of the IDT, each task uses same one. If one task changes the GDT or IDT, all tasks are affected. The LDT is commonly used to define descriptor used by a single process; normally, each process has its own LDT. The location of the LDT is defined by the (LDTR). The LDTR is a 16-bit register, which contains a global selector, this selector refer to an entry in the GDT containing the base, limit, etc. of the LDT. The contents of the LDTR are normally changed on each context switch, allowing each process to refer to its own LDT (Oney, 1996).

## Global and Local Descriptor Table Format

The first entry of GDT is reserved, and the corresponding selector called null segment selector. There are two groups of descriptor in GDT:-

**A- CODE/DATA or SEGMENT Descriptors**

The descriptor contains a base address, a segment limit, and access control flags that govern memory access, as shown in (fig.3) (Brey, 1997).

**B- System Descriptors**

The structure of this descriptor is similar to Code/Data descriptor and there are some differences as in figure (3) in this figure bit (44, 52, 53, and 54) are always zero (Brey, 1997).

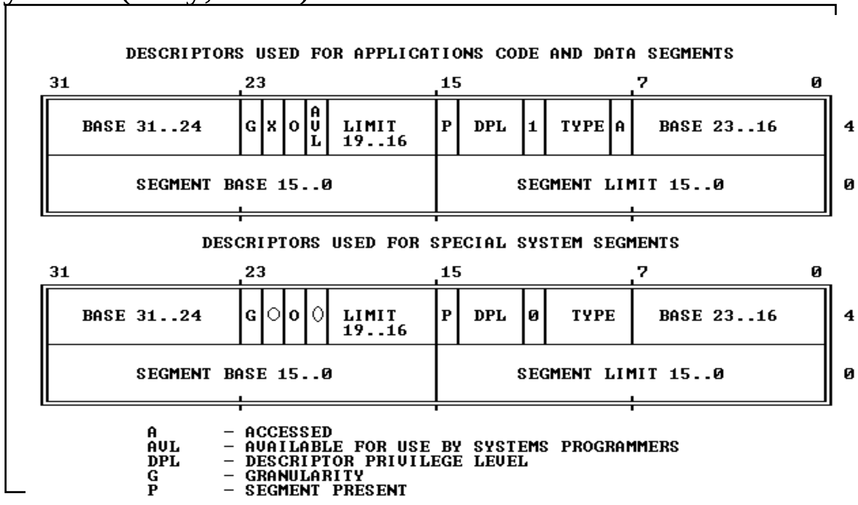

**Fig. (3): DATA/ CODE & System descriptors for the Intel 80386 through the Pentium pro microprocessor**

LLDT and SLDT are two instructions that can be used to load the address of the LDT into LDTR and to store this address. The LLDT is a privilege instruction, while the SLDT is not (Brey, 1997). The LDT is accessed in the manner as the GDT. The only different in access is that the TI bit is cleared for a global access and set for a local access (show in fig.1). Another difference exits if the LDTR and GDTR are examined. The first 16 descriptors in LDT are always empty (Oney, 1996).

## Special Registers

Below list of special registers used in 80386 microprocessor and above are:-

1- GDTR (Global Descriptor Table Register)
2- IDTR (Interrupt Descriptor Table Register)
3- LDTR (Local Descriptor Table Register)
4- TR (Task Register): identifies the currently executing task by pointing to the Task state segment (TSS).

5- CR0-CR3 (Control Registers): are special read-only registers that store a constant. Attempts to write to a constant register are illegal or ignored.
6- DR0-DR7 (Debug Registers): These registers are accessed by variants of the MOV instruction. A debug register may be either the source operand or destination operand. The debug registers are privileged resources; the MOV instructions that access them can only be executed at privilege level zero. Any attempt to read or write the debug registers when executing at any other privilege level causes a general protection exception (Oney, 1996).

## Work Method

1- Store the address of new routine (The routine contain reading the special register like (CR0-CR3) and (DR0-DR3) in memory location that cannot be accessed just in protected mode (ring 0).

```
Mov ebx, offset subrot
Mov [NLOC], bx
Shr  ebx, 16
Mov [NLOC+6], bx
```

2- Attempt to reach the GDT by reading the contents of GDTR (Last four bytes) that contain the base address of GDT using SGDT assembly instruction

3- Reading the content of LDTR using SLDT and add it with the base address of GDT to reach to LDT entry in GDT and then take from this entry the base address of LDT

4- Because the first 16 entry of LDT is empty we take advantage of this feature by storing the offset address of the above new routine inside one of them, in this work it selects entry 0

```
{Eax=address LDT (entry 0)
Mov edi, eax
Mov esi offset NLOC
Movsd
Movsd
Call [new routine address] }
```

5- By running that entry it should jump to the new service routine and execute it

## Conclusion

1- By this work we can access to protected mode Ring 0 (System mode).
2- It is possible to access special registers inside processor.
3- Also by this work it's possible to add additional service to OS.
4- It is possible to add malicious program (virus) in GDT and LDT.

## References

- Brey, B., (1997): The Inte l8086 /8088, 80186 /80188, 80286, 80386, 80486, Pentium, and Pentium Pro processor, architecture, programming, and interfacing , Prentice-Hall.Inc., pp.643-644, pp.57-58.
- Kaplan,Y., (1997-2010): Protected mode memory management.
- Oney,W., (1996): System programming for Windows 95.United state, Redmond, Washington, pp.68-89.
- Solomon,D.A., (1998): Inside Windows NT, second edition ,Microsoft Press.
- Sreyh,R.J., (2004): Simple Middleware for Windows NT, MSc. Theses, Baghdad University, Baghdad.
- Wikipedia**, the free encyclopedia, (2010): An X86 Processor Mode, Protected Mode, Share Alike License.
- Wikipedia***, the free encyclopedia, (2010): Global Descriptor Table, Share Alike License.

# أسلوب مقترح للوصول الى وضع المحمية في نظام التشغيل ويندوز

**إيناس فائق عزيز**

**قسم أنظمة الحاسوب، المعهد التقني _ كركوك**

تاريخ الاستلام:١/١١/٢٠١٠ ،تاريخ القبول:٢٦/٤/٢٠١١

## الخلاصة

نظم الحاسوب اليوم تسمح بتحميل برامج متعددة في الذاكرة، ويتم تنفيذها بشكل متزامن. هذا التطــور يتطلب حماية بين تلك البرامج المختلفة. لتأمين هذه الحماية، توفر أنظمة التشغيل نظاما محميــا عــن طريــق جداول وصفية تحمل هذه الجداول وصف للذاكرة وتسيطر على الوصول لمناطق مختلفة من الذاكرة.

في معمارية إنتل، وبشكل أكثر تحديدا في الوضع المحمي، إدارات الذاكرة و روتينات المقاطعة يتم السيطرة عليهم من خلال جداول وصفية. كل جدول تخزن معلومات حول كائن واحد (على سبيل المثال روتين الخدمــة، مهمة، مقطع من برنامج ، أيا كان) وحدة المعالجة المركزية قد تحتاج لهذه المعلومات في بعض الوقت.

في هذا البحث ، تمت دراسة هذه الجداول ومداخلها ودراسة محتوياتها أيضا مع توضيح طريقة لتغيير محتويات بعض المداخل لدراسة سلوك نظام التشغيل ، بمعنى أخر الوصول إلى وضع المحمية والعمل فيه مع طباعة محتويات بعض المسجلات التي لا يمكن الوصول إليها إلا داخل وضع المحمية.

أنجز هذا البحث باستخدام لغة التجميع (MASM الإصدار ٧) تحت بيئــة Windows Me لكونهــا تحتــوي ايعازات تستطيع التعامل مع مسجلات خاصة نحتاجها في هذا البحث (GDTR and LDTR) والذي بــدورهما يحتويان عناوين تلك الجداول.