Memory Accessing in GDT and LDT

Presented By :

Roshan Saud

Pranjal Rimal

Gambhir Jung Kunwar

INTRODUCTION TO DESCRIPTOR TABLE

- Actually, a descriptor table is a linear array of up to 8K (8192) descriptors. The upper 13 bits of the selector field are an index into a descriptor table. Each descriptor table has a 24-hit base register to locate the descriptor table in physical memory and it has a 16-bit limit register which confines descriptor access to the defined limits of the table

- ***What actually is a descriptor?***

  - A descriptor is a data structure used to define the properties and boundaries of memory segments. These descriptors are crucial for the CPU to manage and protect memory efficiently, supporting the segmentation model of memory management.

- In 80286 microprocessor there are 3 different types of descriptor table i.e. Global Descriptor Table (GDT), Local Descriptor Table (LDT) and Interrupt Descriptor Table (IDT).

- The Global Descriptor Table (GDT) contains descriptors available to all tasks. The Local Descriptor Table (LDT) contains descriptors that can he private to a task.
- The GDT may contain all descriptor types except interrupt and trap descriptors. The LDT contains segment, task gate, and call gate descriptors.
- A segment cannot be accessed by a task if its segment descriptor does not exist in either GDT or LDT at the time of access.
- The LGDT (Load Global Descriptor Table) and LLDT (Load Local Descriptor Table) instructions load the base and limit of the GDT and LDT.
- The LGDT and LLDT are privileged, and these instructions may only be executed by programs at privilege level 0.
- The LGDT instruction loads a six-byte field containing the 16-bit limit and 24-bit physical base address of the GDT.
- The LLDT instruction loads a selector which refers to an LDT descriptor containing the base addresses and limit.

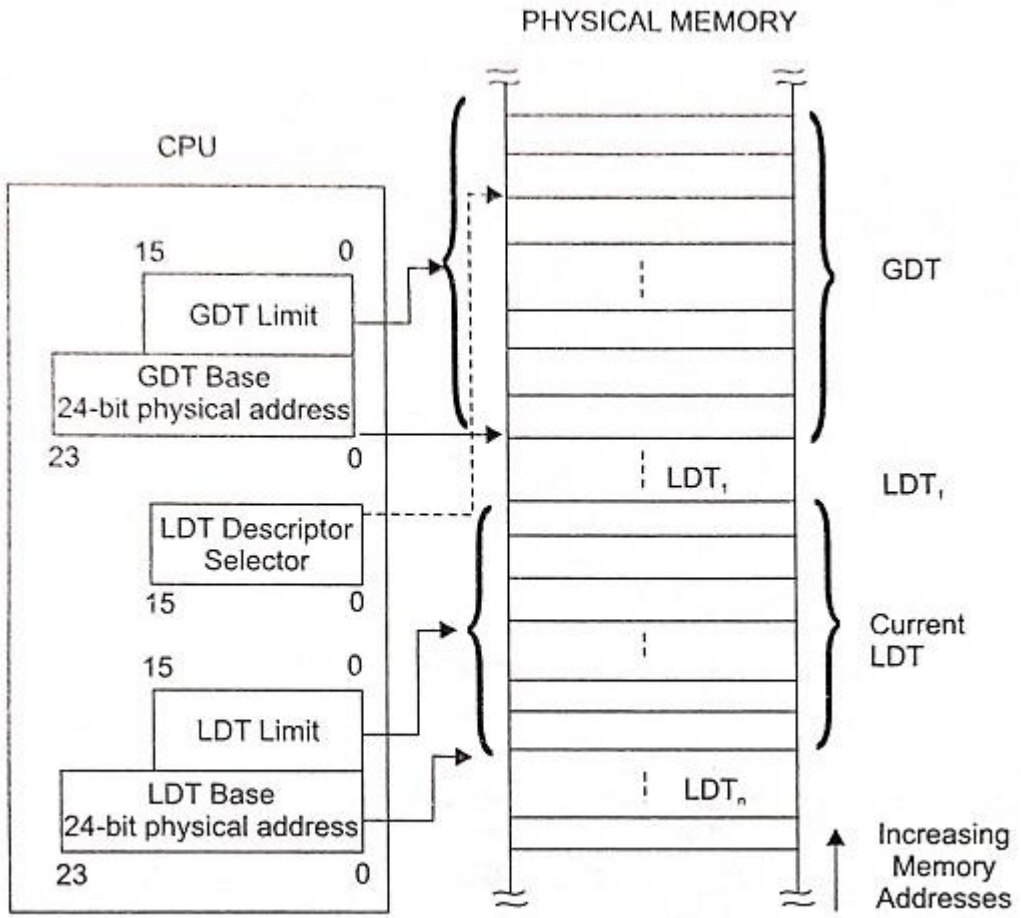

Fig. 11.27 *Local and global descriptor table definition*

Interrupt Descriptor Table

- In the protected mode, the 80286 processor has a third descriptor table known as interrupt Descriptor Table (IDT).
- The IDT can be used to define up to 256 interrupts.
- The IDT contains task gates, interrupt gates and trap gates.
- The IDT has a 24-bit physical base and a 16-bit limit register in the CPU.
- The privileged LIDT (Load Interrupt Descriptor Table) instruction loads these registers with a 6-byte value in same way of the LGDT instruction.
- Usually, the IDT entries are made through INT instructions, external interrupt vectors, or exceptions. The IDT should have 256 bytes in size to allocate space for all reserved interrupts.

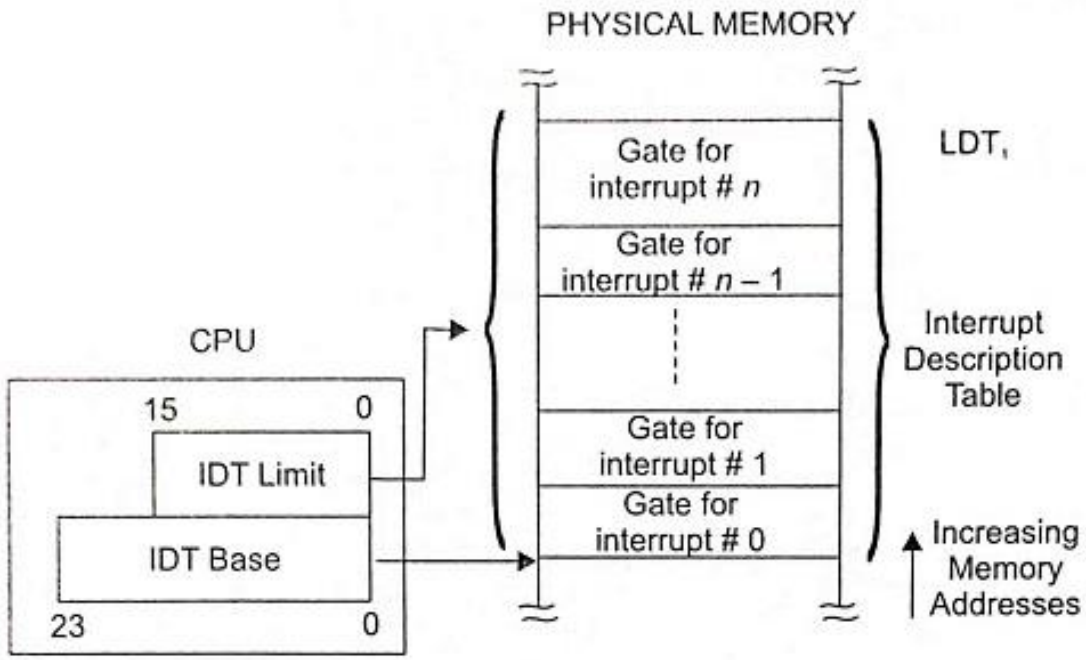

Fig. 11.29 *Interrupt descriptor table*

MEMORY ACCESSING USING GDT AND LDT

1. **Using GDT**
    - The GDT is accessed via the GDTR (Global Descriptor Table Register), which stores the base address and limit of the GDT. The CPU uses this information to find and use the descriptors in the GDT.
    - When accessing memory, segment registers (e.g., CS, DS) reference indices in the GDT. These indices point to the relevant descriptors, which the CPU uses to calculate the linear address by adding the base address to the effective address (offset).
    - The CPU then checks the segment limit and access rights defined in the descriptor to ensure the access is valid, ensuring controlled and protected memory accesses.

2. Using LDT

- The LDT is accessed through the Local Descriptor Table Register (LDTR), which the CPU uses to locate the LDT for the current task.
- When accessing memory, segment registers reference indices in the LDT. These indices point to process-specific descriptors, which the CPU uses to calculate the linear address by adding the base address to the effective address (offset).
- The CPU checks the segment limit and access rights defined in the descriptor to ensure valid access, supporting process-specific memory segmentation and enhancing security through isolation.

Privilege

- The 80286 processor can support a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors within a task.
- The privilege levels are numbered 0 through 3. Level 0 is the most privileged level whereas Level 4 is the least privileged level. Privilege levels provide protection within a task.
- The operating system routines interrupt handlers, and other system software can he protected from unauthorized accesses within the virtual address space of each task using the four levels of privilege.
- Each task in the system has a separate stack for each privilege levels. Tasks, descriptors, and selectors have a privilege level attribute that can find out whether the descriptor may be used.
- The task privilege has an effect on the use of instructions and descriptors. The descriptor and selector privilege only effect access to the descriptor.

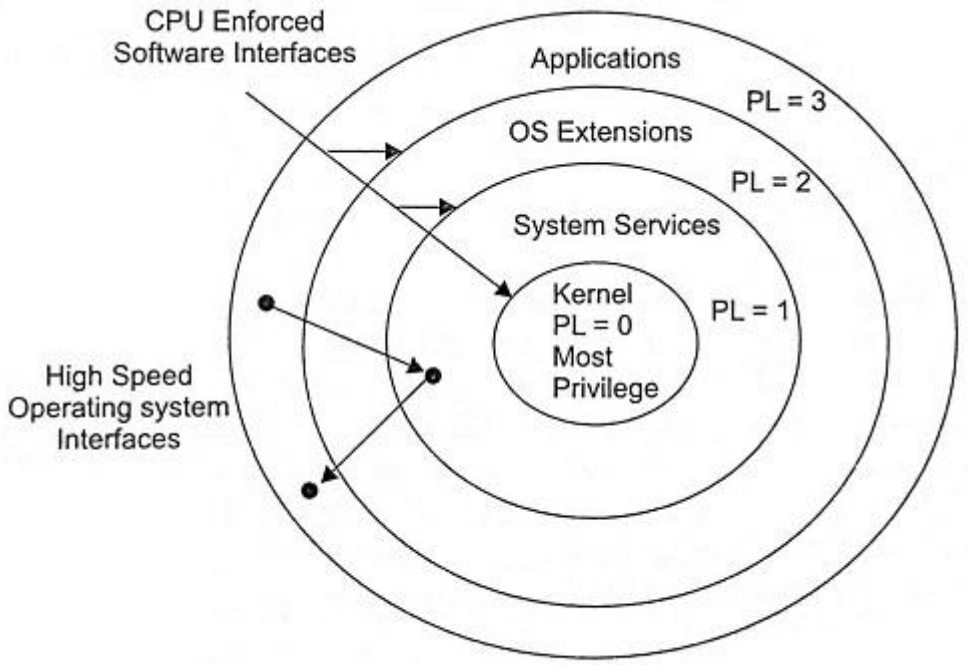

Fig. 11.30 *Four-level privilege*

Descriptor Structure

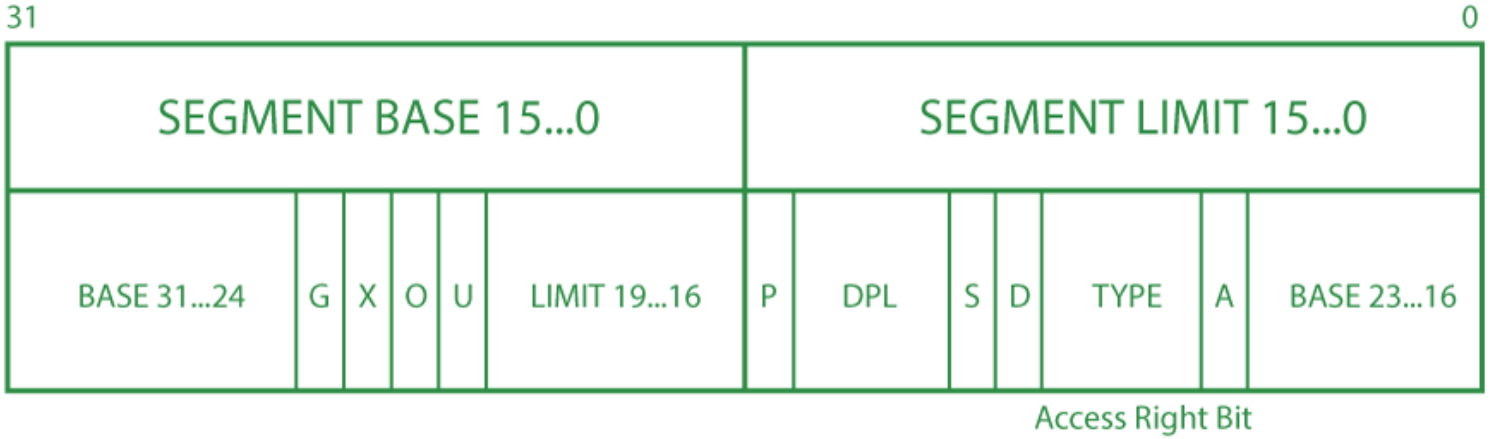

Fig:- General Format of Segment Descriptor

1. Base Address

- Specifies the starting address of the memory segment. All accesses to this segment are relative to this base address. It contains the 32-bit base address for a segment. Thus define the location of the segment within the 4 gigabyte ((232 -1)-bit) linear address space.

2. Segment Limit

- Defines the size of the segment. The limit specifies the maximum offset within the segment. Combined with the base address, it determines the segment's boundaries. The x86 concatenates the two fragments of the limit field to form a 20-bit value. The x86 interprets this 20-bit value in two ways, depending on the setting of the granularity bit(G).

3. Access Byte

- Contains information about the segment type (e.g., code, data), privilege level (user or kernel mode), and whether the segment is present in memory. It defines the access rights and the behaviour of the segment.

THANK YOU