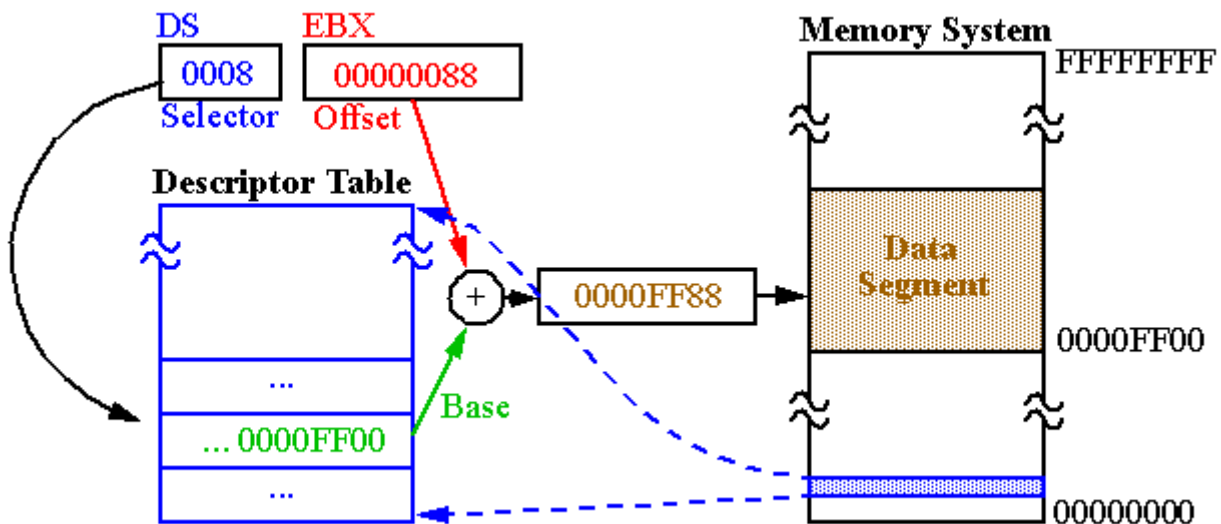


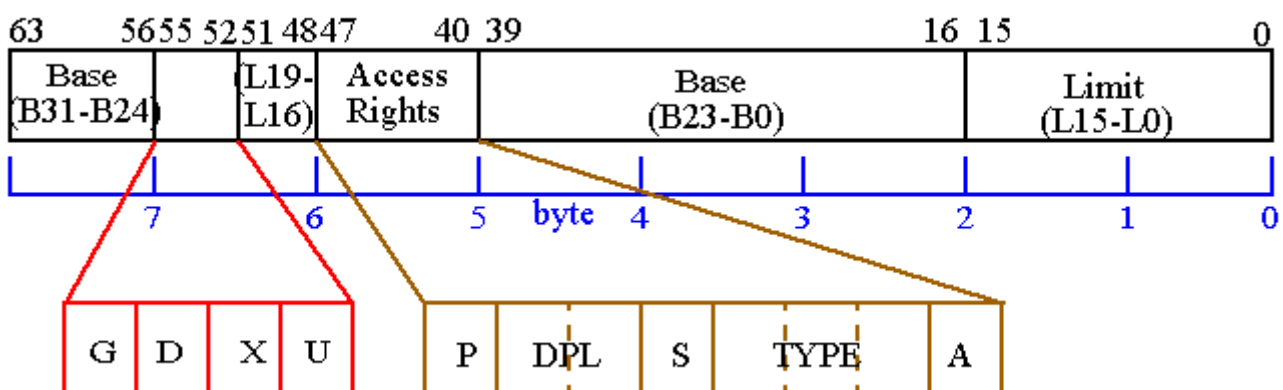
Protected Mode Memory Addressing



◦ *Segments are interpreted differently in Protected Mode vs. Real Mode:*

- *Segment register contains a selector that selects a descriptor from the descriptor table.*
- *The descriptor contains information about the segment, e.g., its base address, length and access rights.*
- *The offset can be 32-bits.*

Segment Descriptors in Protected Mode



• Base address :

◦ *Starting location of the memory segment.*

• Limit :

◦ *Length of the segment minus 1.*

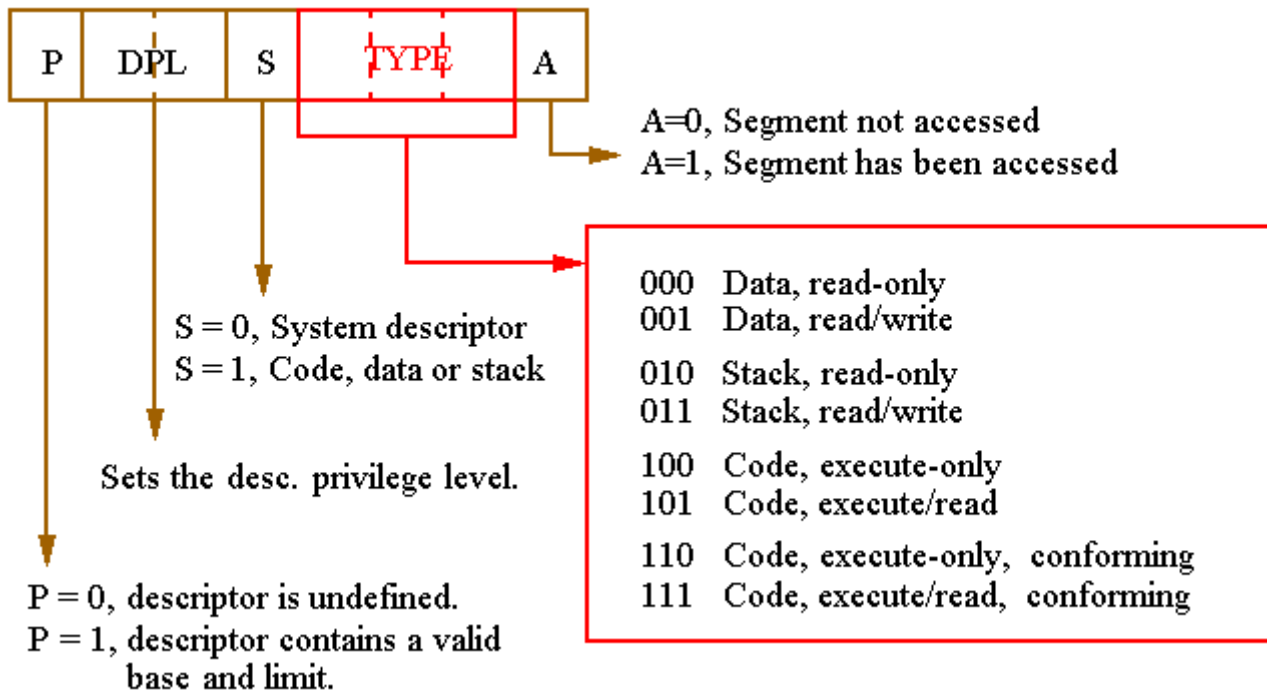
- 20-bits allows segments up to 1 MB.
- This value is shifted by 12 bits to the left when the G (Granularity bit) is set to 1.

Segment Descriptors in Protected Mode

- *Segment Descriptors: Bits 52-55*
- **G bit :**
 - When $G=0$, segments can be 1 byte to 1MB in length.
 - When $G=1$, segments can be 4KB to 4GB in length.
- **U bit :**
 - User (OS) defined bit.
- **D bit :**
 - Indicates how the instructions (80386 and up) access register and memory data in protected mode.
- When $D=0$, instructions are 16-bit instructions, with 16-bit offsets and 16-bit registers. Stacks are assumed 16-bit wide and SP is used.
- When $D=1$, 32-bits are assumed.
 - Allows 8086-80286 programs to run.
- **X bit :**
 - Reserved by Intel

Segment Descriptors in Protected Mode

- *Segment Descriptors: Access Rights (Byte 5):*

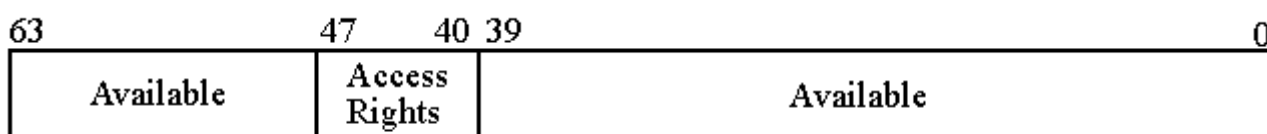


- *The Access Rights (AR) byte controls access to a protected mode segment and how the segment functions in the system.*

Segment Descriptors in Protected Mode

- *Details:*

- *The A (accessed) bit is set automatically by the microprocessor and is never cleared.*
 - *This allows OS code to track frequency of usage.*
- *The P (present) bit should be interpreted as "descriptor-is-valid".*
 - *If this bit is 0, the microprocessor will refuse any attempts to use this descriptor in an instruction.*



- *Although the AR must always be valid, when P=0, the rest of the descriptor can be used in*

any way the OS likes.

- *The S (system) bit indicates how the descriptor is to be interpreted.*
 - *S=1 indicates a system descriptor (more on this later).*
 - *S=0 indicates a code, data or stack descriptor.*

Segment Descriptors in Protected Mode

- *Details:*

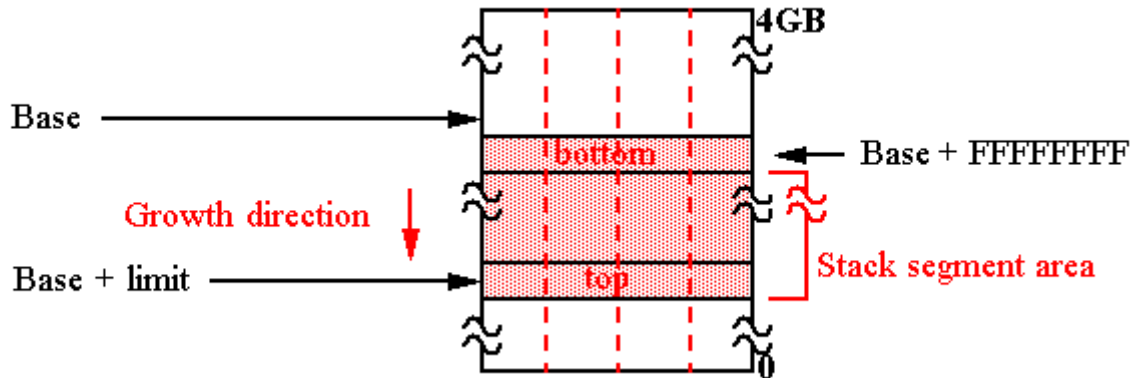
- *Non-system (S =0) segments:*

- *Type=0: The data segment is basically a ROM.*
- *Type=1: Both read and write operations allowed.*
- *Code can **NOT** be fetched and executed from either of these segment types.*
- *Type=2 or 3: A stack segment is defined analogously to Types 0 and 1.*
 - *However, the interpretation of the limit field is different.*
 - *In this case, all offsets must be greater than the limit.*
 - *The upper limit is set to base address + FFFF (with D=0) or base address + FFFFFFFF (with D=1).*
 - *This means the stack segment **ends 1 byte below the base address.***

- *Expanding of the stack segment simply involves decreasing the limit.*

Segment Descriptors in Protected Mode

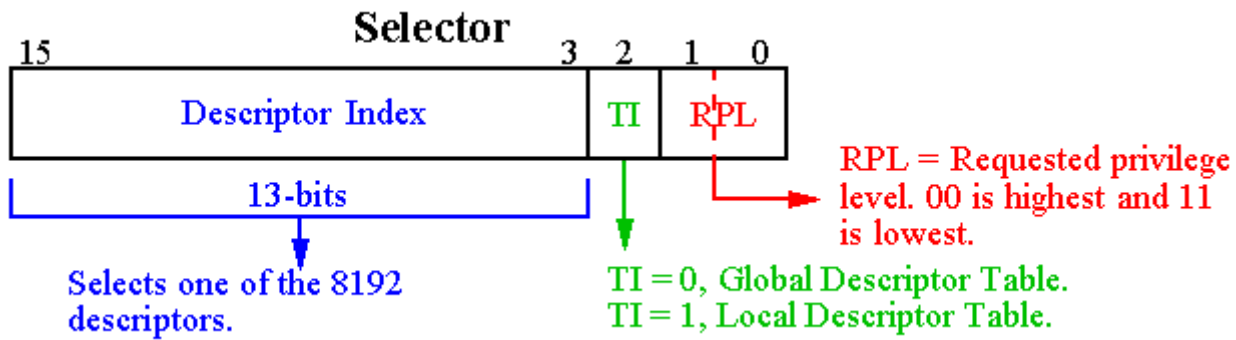
- *Details:*



- *Type=4: A code segment with no read permission.*
 - *This means no constants are allowed, since they cannot be read out.*
- *Type=5: A code segment in which constants may be embedded.*
 - *In either case, no writing (self-modifying code) is permitted.*
- *Type=6 and 7: Analogous to Types 4 and 5 **without** privilege protection.*
 - *We'll discuss the meaning of "conforming" soon.*

Segment Registers in Protected Mode

- *Interpretation:*



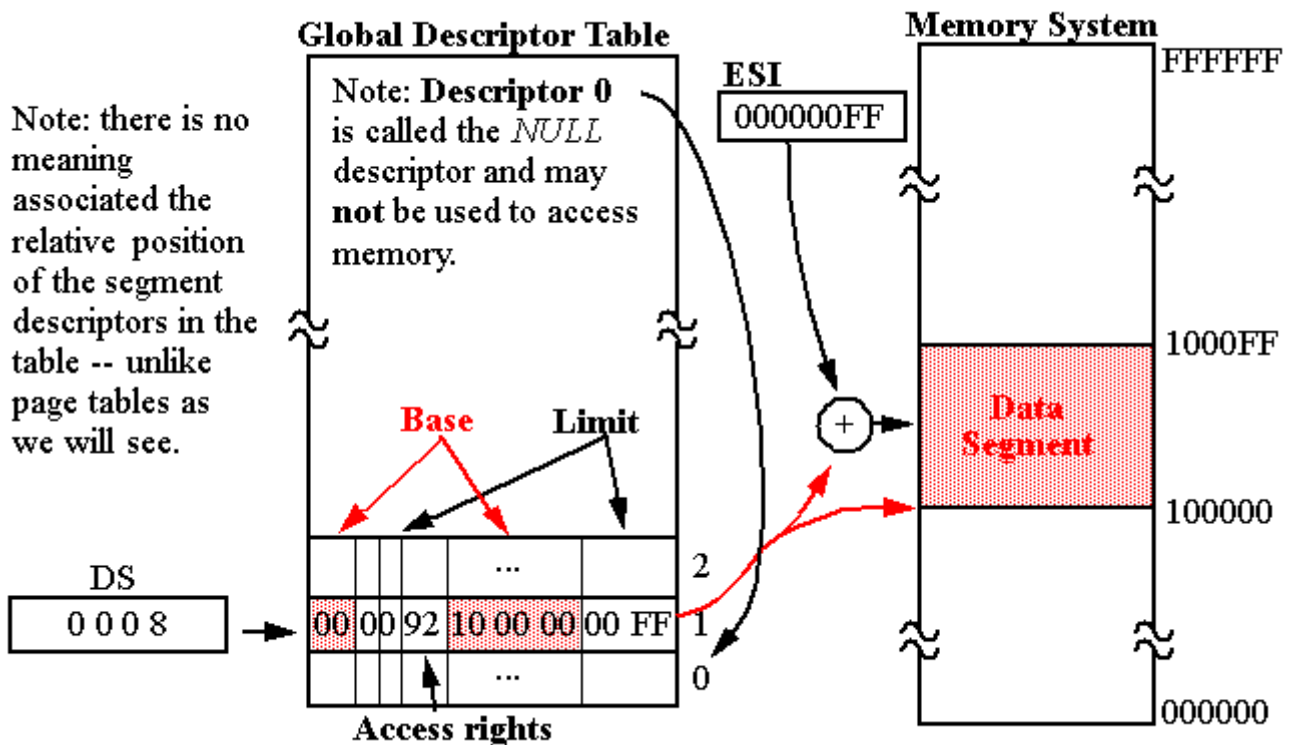
• Descriptor Index and Table Index (TI) :

- The 13 bit descriptor index selects one of up to 8K descriptors in either the GDT and LDT, as specified by the TI bit.
- Therefore, these 14 bits allows access to 16K 8-byte descriptors.

• RPL :

- The desired privilege level of the program.
- Access is granted if the RPL value is lower (higher in privilege) than the AR of the segment. Otherwise, a privilege violation is issued.

Segmentation Address Translation



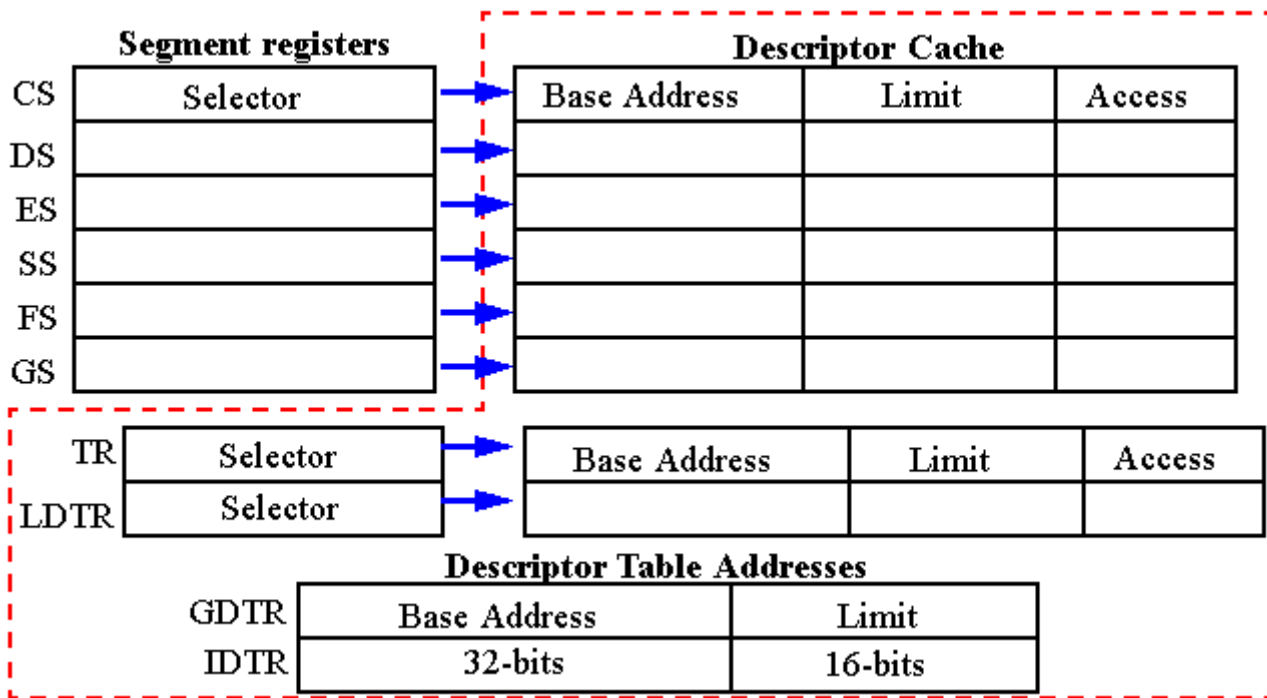
- *So instead of left shifting by 4 bits in Real Mode to form the segment address, we right shift by 3 bits and use the value as a table index.*

Segmentation Address Translation

- *There are actually three different descriptor tables, **GDT**, **LDT** and **IDT**.*
 - *Exactly one **GDT** and **IDT** must be defined for Protected Mode operation.*
- *Global Descriptor Table (**GDT**).*
 - *The GDT is used by all programs.*
- *Local Descriptor Table (**LDT**).*
 - *An LDT can optionally be defined on a per-task basis and is used to expand the addressable range of the task.*
- *Interrupt Descriptor Table (**IDT**).*
 - *The IDT is a direct replacement to the interrupt vector table used in 8086 systems.*
- *Note that references to **IDT** are done through the hardware interrupt mechanism, and not from a program via a selector.*

Segmentation Address Translation

- *Programmer invisible registers:*
 - *The **GDT** and **IDT** (and LDT) are located in the memory system.*



- *The addresses of the GDT and IDT and their limits (up to 64K bytes) are loaded in special registers, **GDTR** and **IDTR**, before switching to Protected Mode is possible.*

Segmentation Address Translation

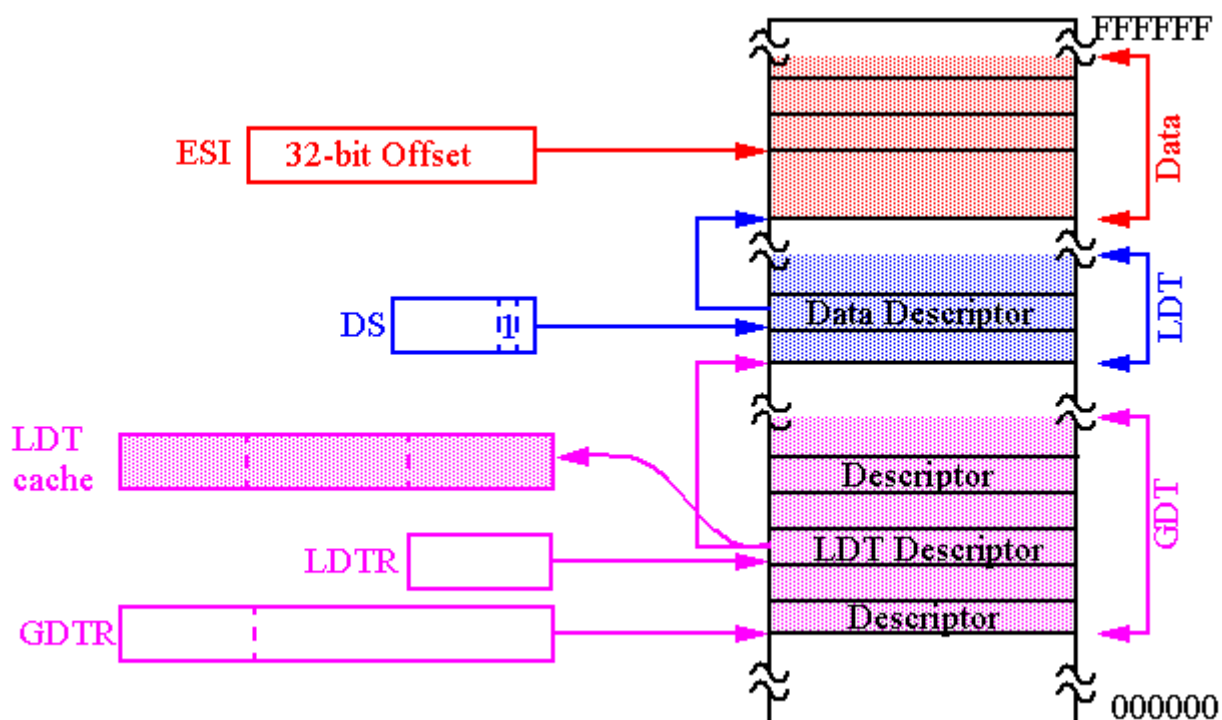
- *Programmer invisible registers:*
 - *The other registers enclosed by the red-dotted line are part of the descriptor cache.*
 - *The cache is used to reduce the number of actual memory references needed to construct the physical address.*
 - *There is one cache register for each of the 6 segment registers, CS, DS, etc. and the LDTR (Local Descriptor Table Register) and TR (Task Register) selectors.*
 - *The base address, limit and access rights of the descriptor are loaded from memory every time the corresponding selector changes.*

- The LDTR and TR selectors refer to special **system** descriptors in the GDT.
 - These registers provide hardware acceleration support for task switching.

- Let's first consider how LDTs are used to extend the address space of individual tasks.

Local Descriptor Tables

- The LDTR selector indexes a GDT **system** descriptor describing the segment containing the LDT while the cache stores the actual LDT descriptor.



- The LDTR selector can be loaded with a new value when another task is run.

Local Descriptor Tables

- LDT Segment Descriptor:

63	56	55	52	51	48	47	40	39	16	15	0
Base (31-24)	0000	Lim (19-16)	P000	0010	Base (23-0)					Limit (15-0)	

- *Bit 44: The S flag is clear to indicate an LDT descriptor.*
- *Bit 40-43: The Type field is extended to 4 bits (no Accessed bit). Type 2 (0010) indicates a LDT descriptor.*
- *Bit 47: If the Present bit is not set (e.g. there is no LDT defined), the 80x86 will not allow you to load the LDTR with its selector.*
- *Bit 0-15, 16-19: Although the limit is still 20 bits (and the G bit is also valid), segments larger than 64KB don't make sense!*