**Selectors - Descriptor**

**Descriptor**

1. Location
2. Length
3. Access Right

We have global descriptor and local descriptor each of (8192).

**Base address** 🡪 specifies starting location of memory segment.

80286 🡪 24 bit (can access any segment at any location within 16 MB of memory).

80386 🡪 32 bit (can access any segment at any location within 4GB mb of memory).

**Limit** 🡪 contains the last offset address in the segment.

80286 16-bits [base + limit] (1-64k byte length)

80386 20 bits [base + limit] (1M or 4k-4G byte length)

G bit (granularity bit)

G=0 (limit to 20 bit) 🡪 FFFFFH

G=1 appends FFFH (multiply by 4k 🡪 FFFFFFFFH

D bit specifies the instruction 16 or 32 bit

D=0 16 bit (instruction of 16 bit, all instruction, registers, offsets are 16)

D=1 32 bit (instruction of 16 bit, all instruction, registers, offsets are 32)

AV – availability (if two programs want to use the same segment)

AV=0. The segment is not available.

AV=1. The segment is available.

**Access right**

1. Control the access to protected mode segment.
2. If segment goes beyond this limit, program is interrupted by MP indicating a general protection fault.
3. Can specify whether a data segment can be written or is write protected.

Segment register

|  |  |  |
| --- | --- | --- |
| selector | TI | RPL |
| 15 3 | 2 | 1 0 |

Access Right bit

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | D | Pl | S | E | ED/C | R/W | A |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

TI🡪 0 for global descriptor, 1 local descriptor.

RPL🡪 requested privilege level of memory segment– 00-11 highest to lowest.

A🡪 0 segment not accessed, 1 segment is accessed

R🡪 0 data can’t read, 1 data can be read

W🡪 0 data cannot be written; 1 data can be written.

ED🡪 0 segment expand upward (data segment),1 expand downward (stack segment).

C🡪 0 ignore DPL,

S🡪 0 system descriptor, 1 code or data segment descriptor.

P🡪 0 undefined descriptor, 1 segment contain valid base + limit.

E🡪 0 data segment, 1 code segment.