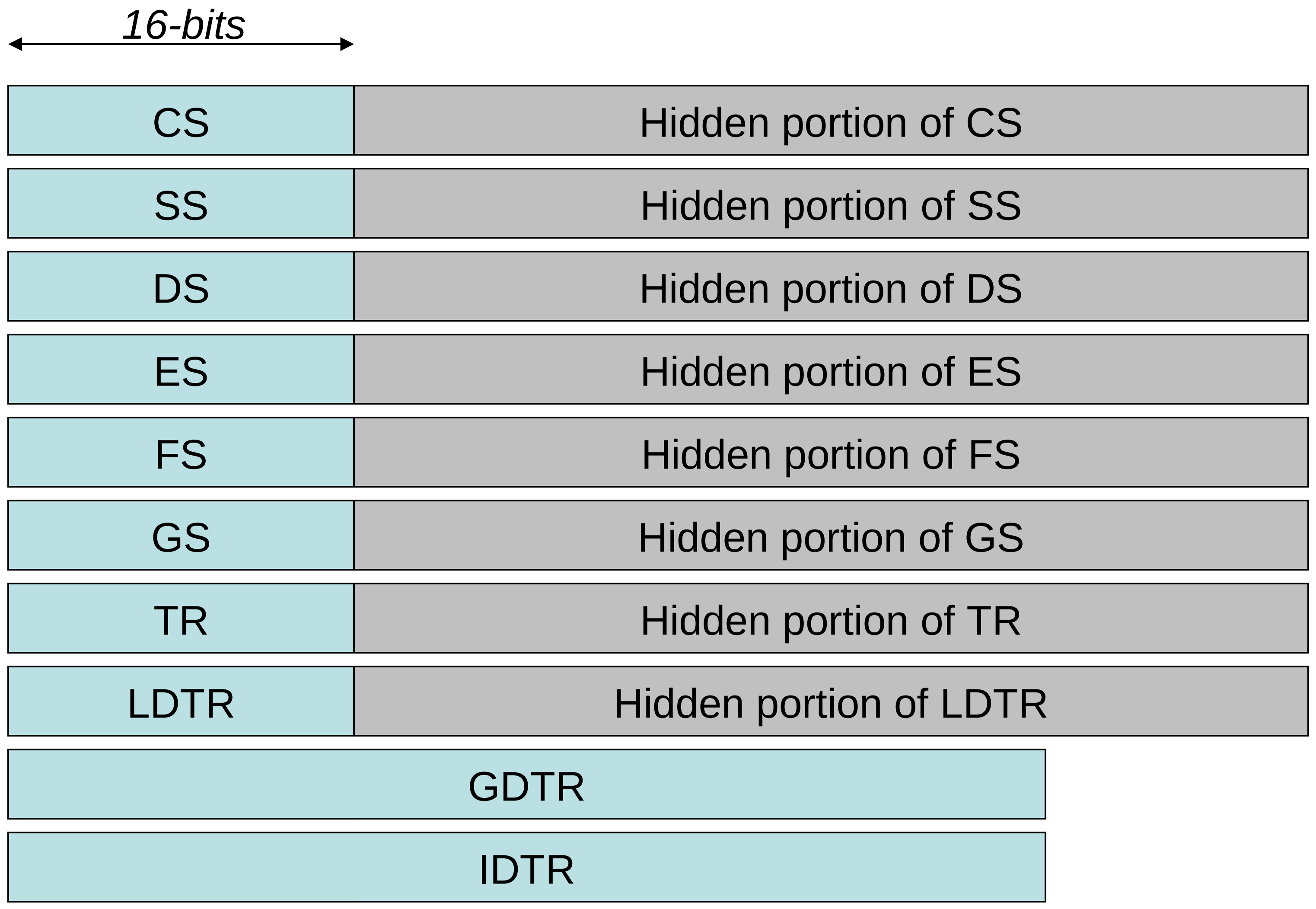
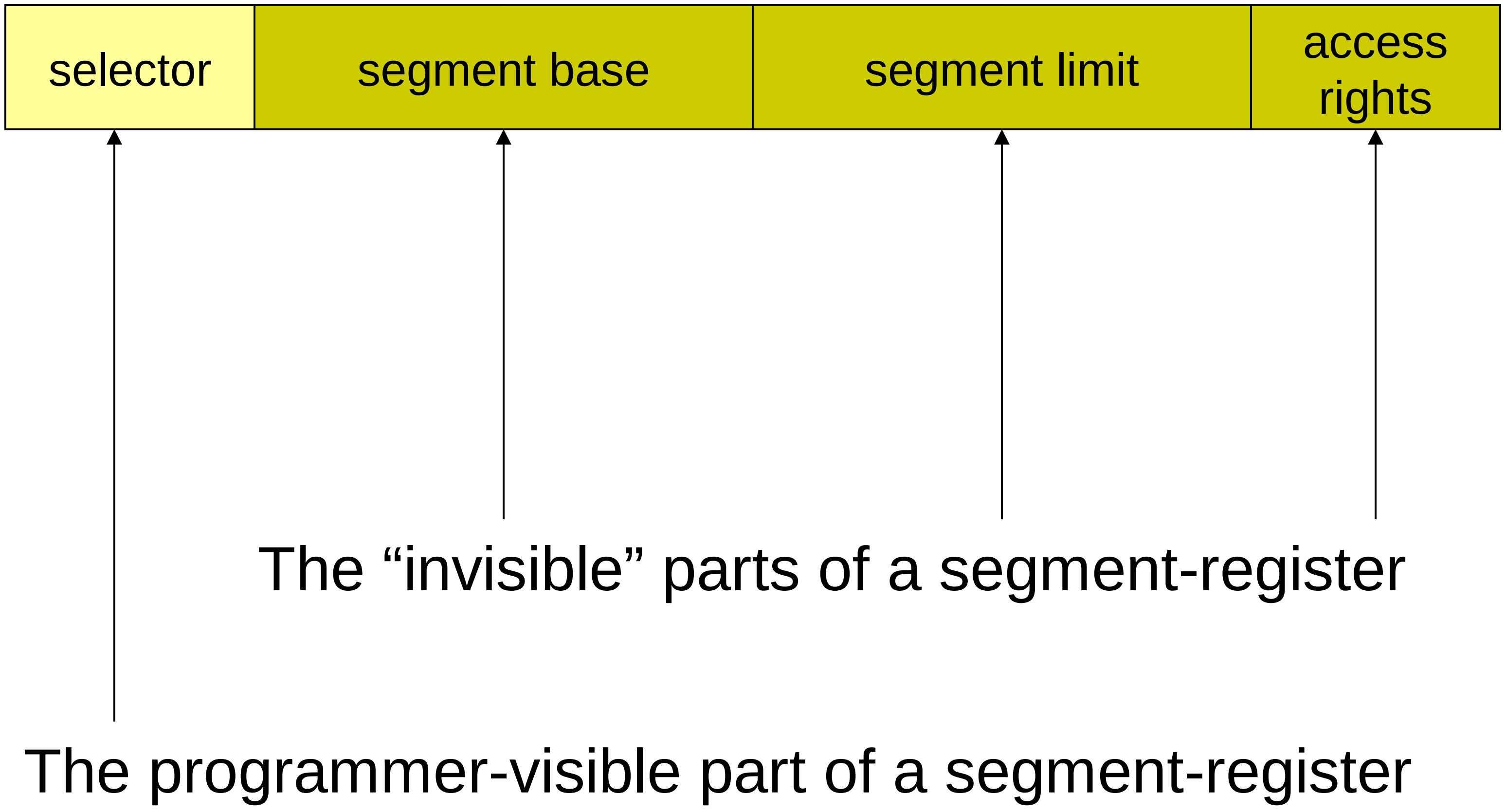
Segment-registers’ hidden bits

A look at how segmentation attributes are cached within the CPU’s segment-registers

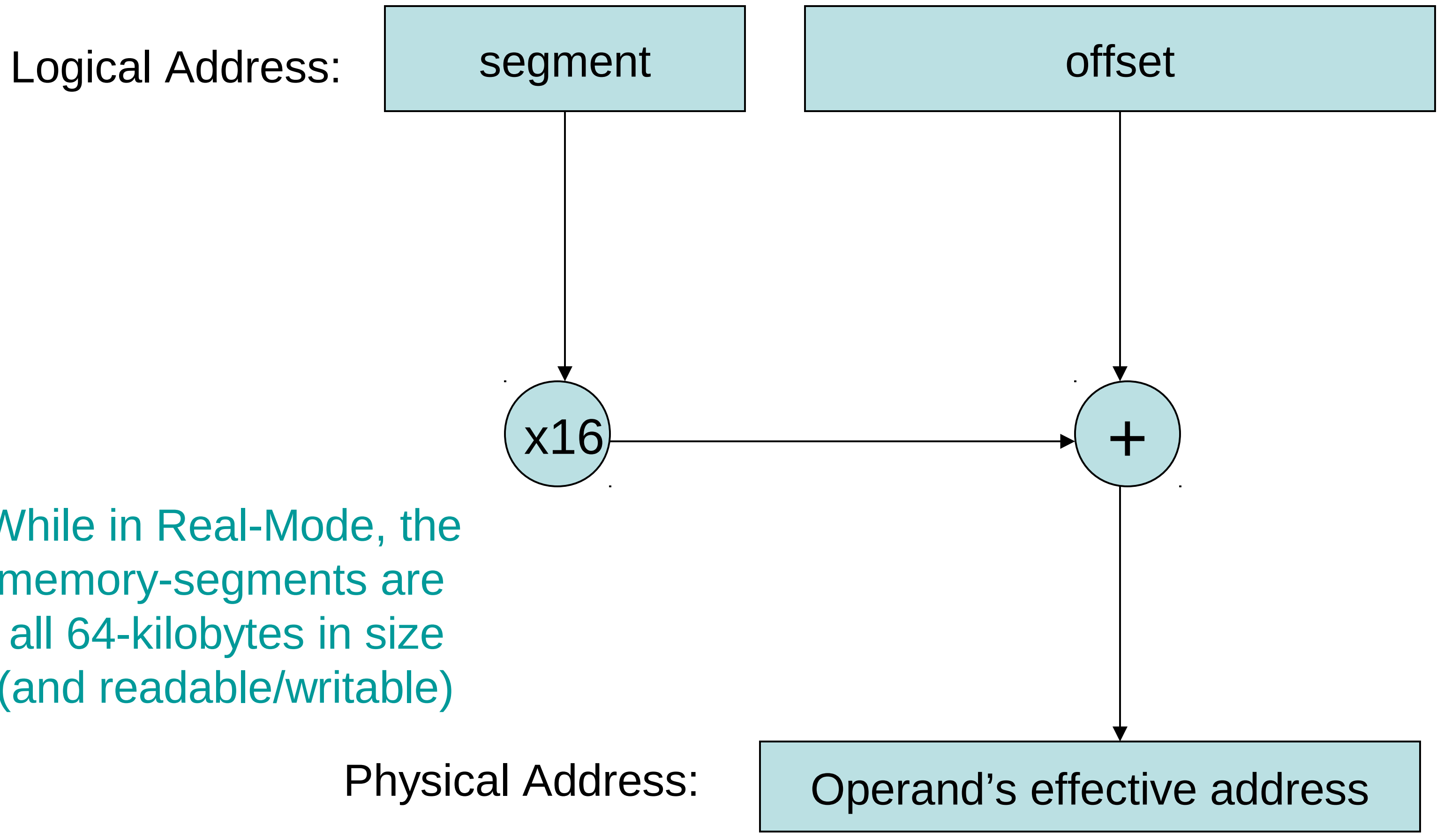
CPU segment-registers



“Hidden” part of segment-registers

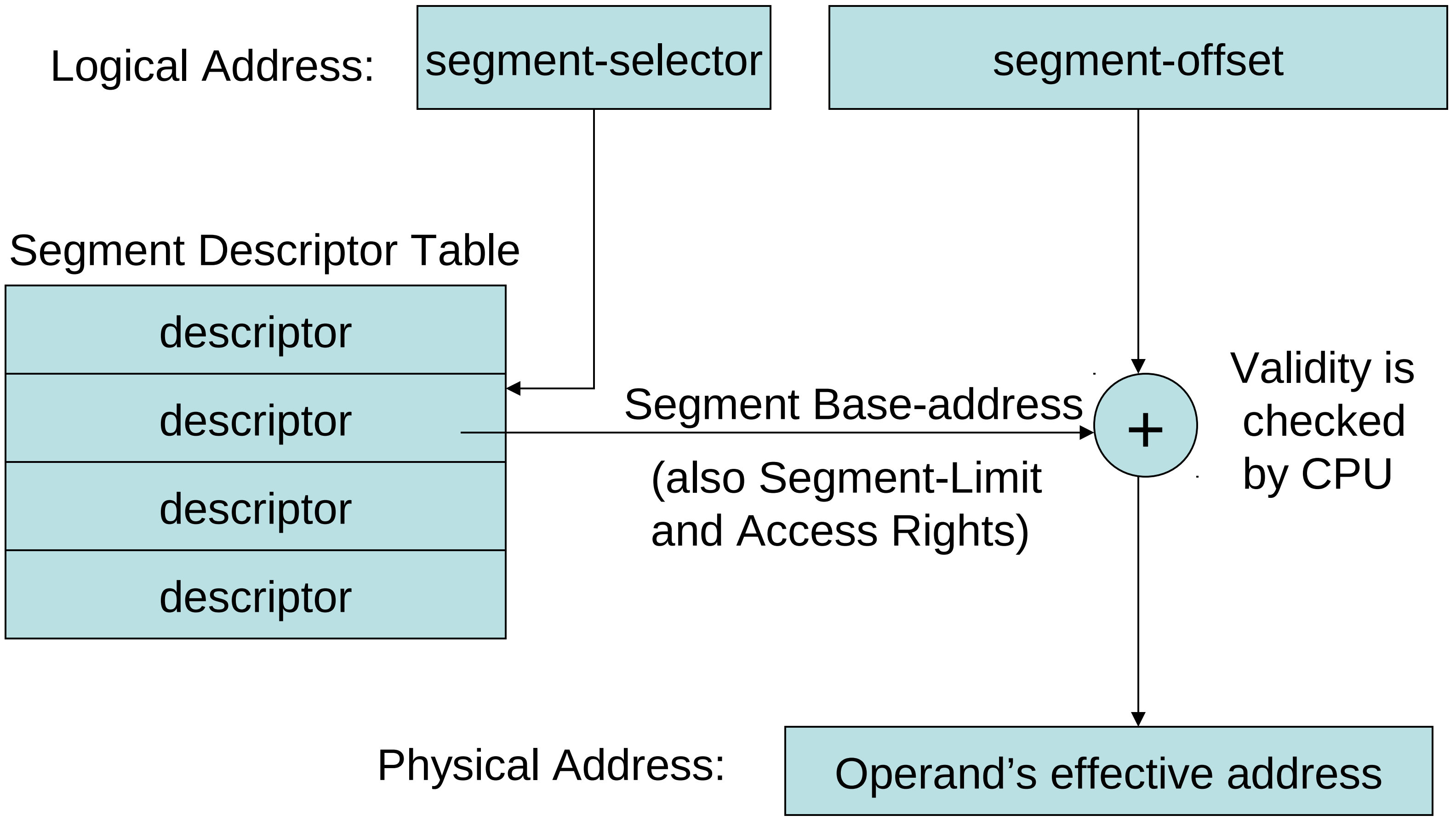

selector
segment base
segment limit
access rights
The “invisible” parts of a segment-register
The programmer-visible part of a segment-register

Real-Mode Addresses

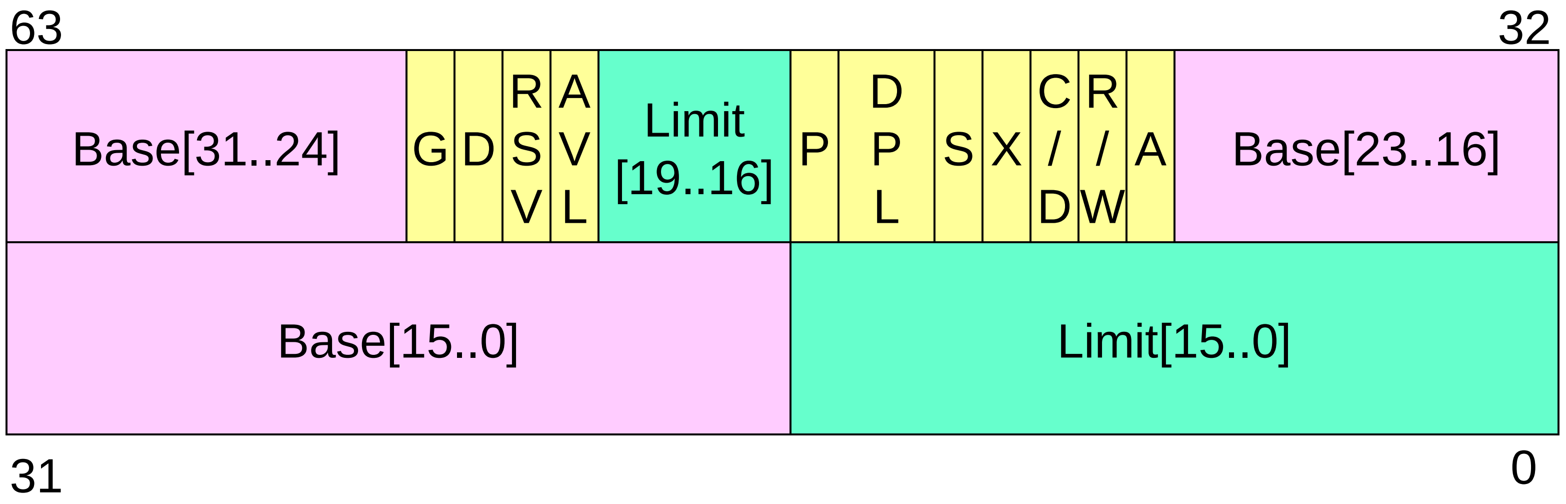


While in Real-Mode, the memory-segments are all 64-kilobytes in size (and readable/writable)

Protected-Mode Addresses

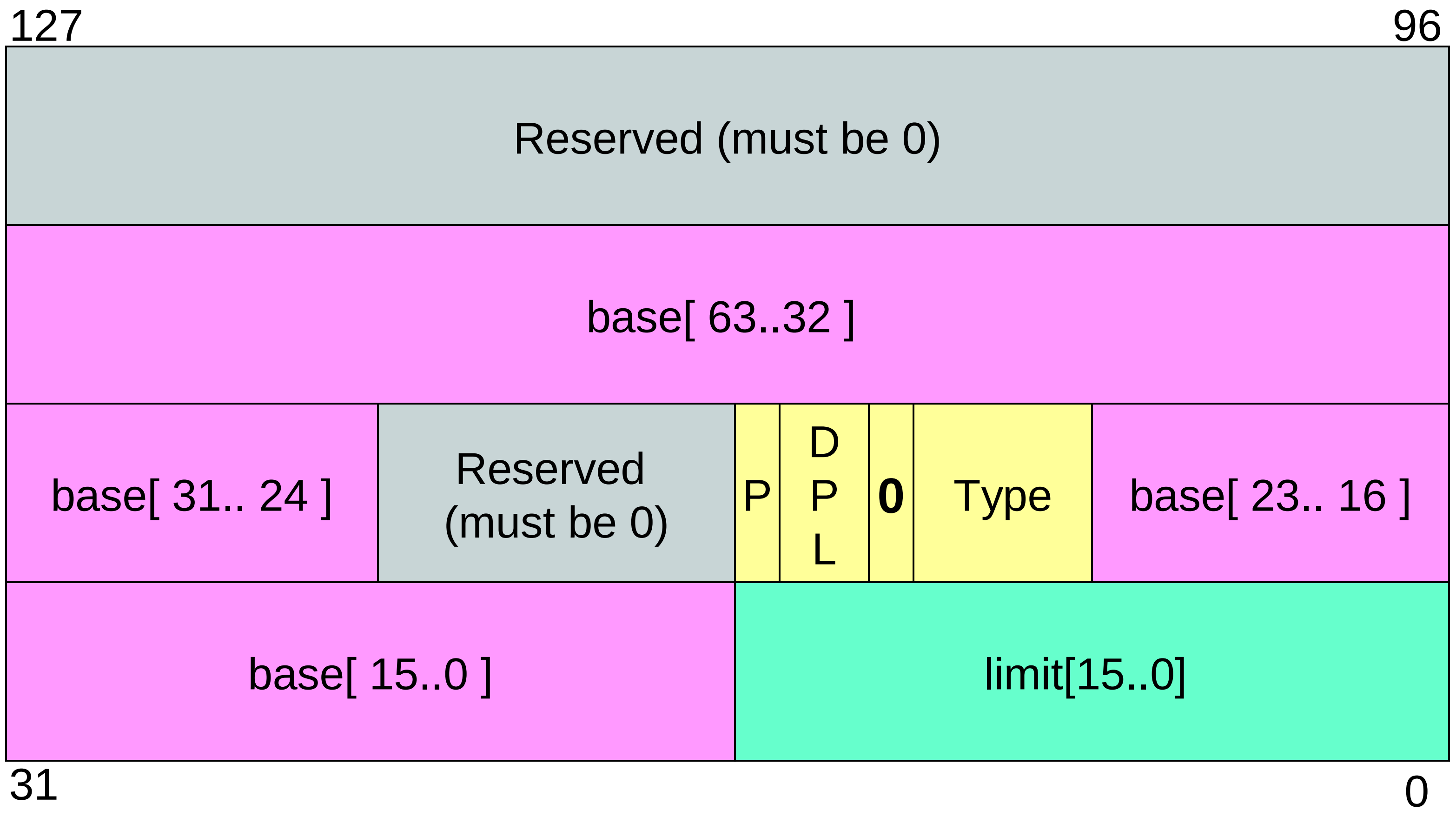


Segment-Descriptor Format



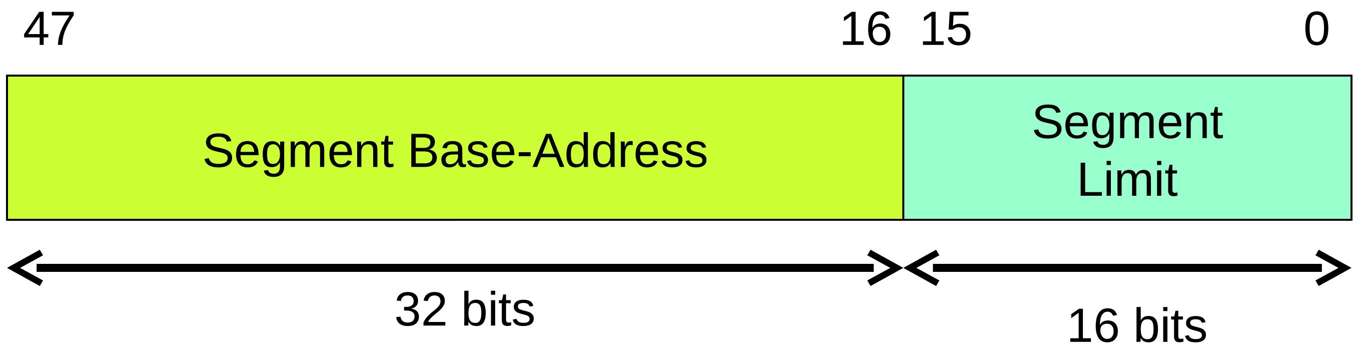
Several instances of this basic **'segment-descriptor'** data-structure will occur in the **Global Descriptor Table** (and maybe also in some **Local Descriptor Tables**)

EM64T TSS/LDTR descriptors

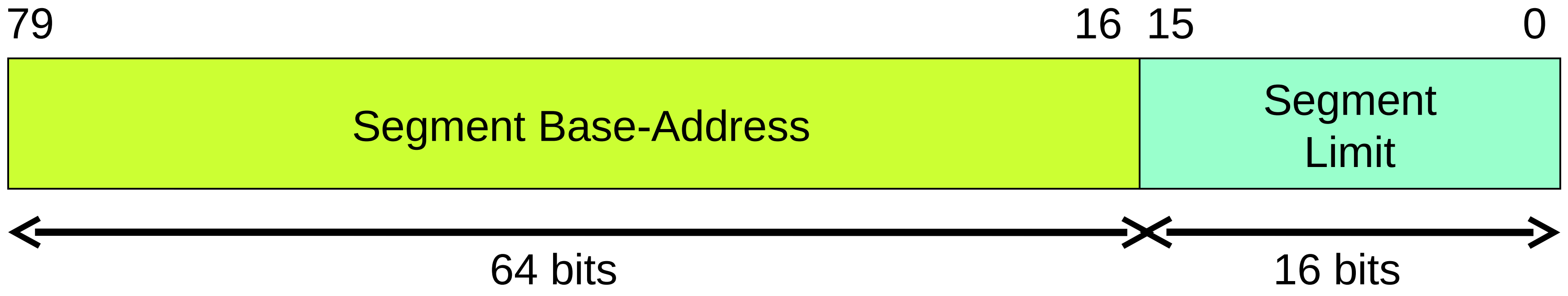


Type: 1010 = LDT descriptor, 1001 = available TSS, 1011 = busy TSS

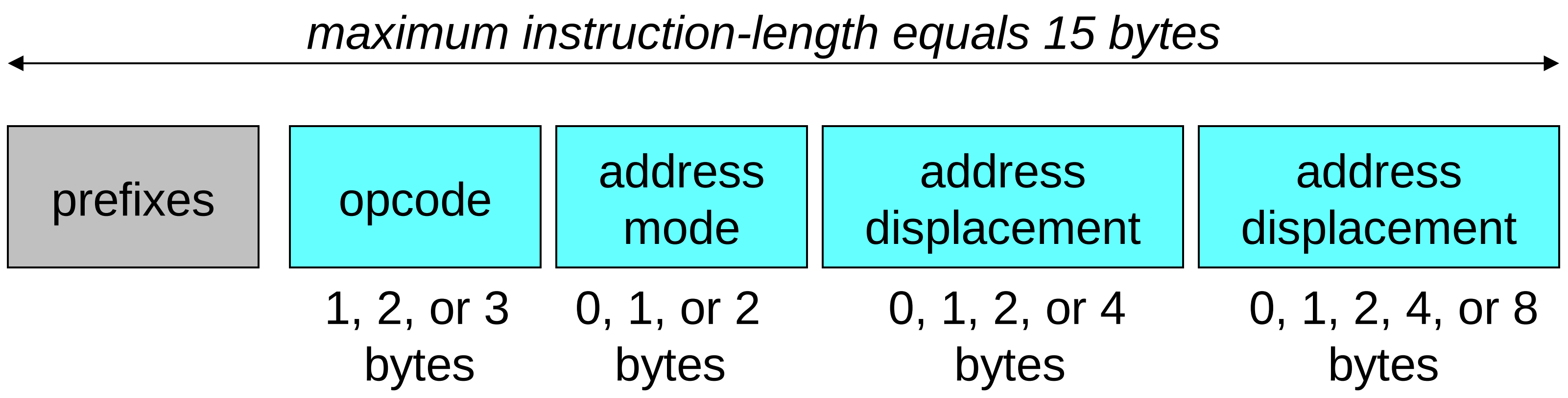
48-bit Register-Format



EM64T 80-bit Register-Format



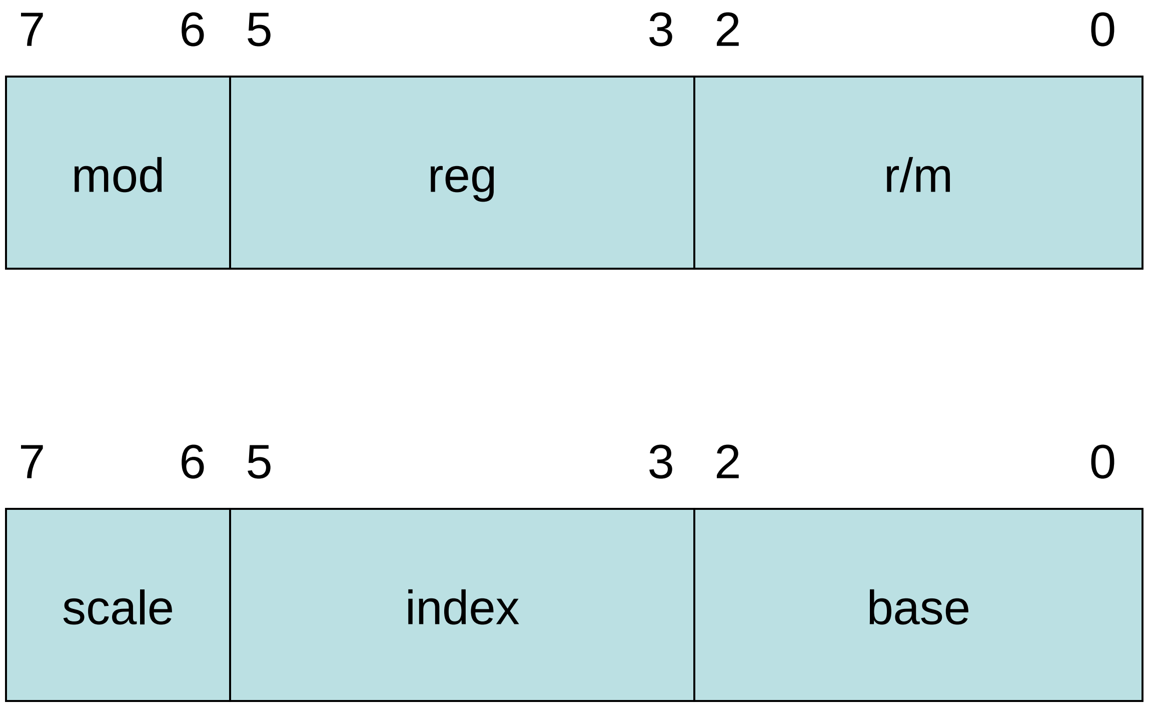
Intel instruction-format



Prefix bytes are used to ‘override’ the processor’s current default settings:
- (1) selection of memory-operand segment-register
- (2) current default setting of operand’s width
- (3) current default setting of address’s width

Or to modify the way in which the subsequent instruction will execute:
- (4) lock the bus for duration of instruction’s execution
- (5) conditionally reexecute a string-instruction
- (6) provide extra operand-selection bits (IA-32e mode)

Mode-r/m and SIB bytes



‘hangdemo.s’

- This program shows why ‘illegal’ values left in the FS and GS segment-registers can cause our demo-programs to ‘hang’ when leaving ‘protected-mode’ and then attempting to execute some ROM-BIOS service-functions