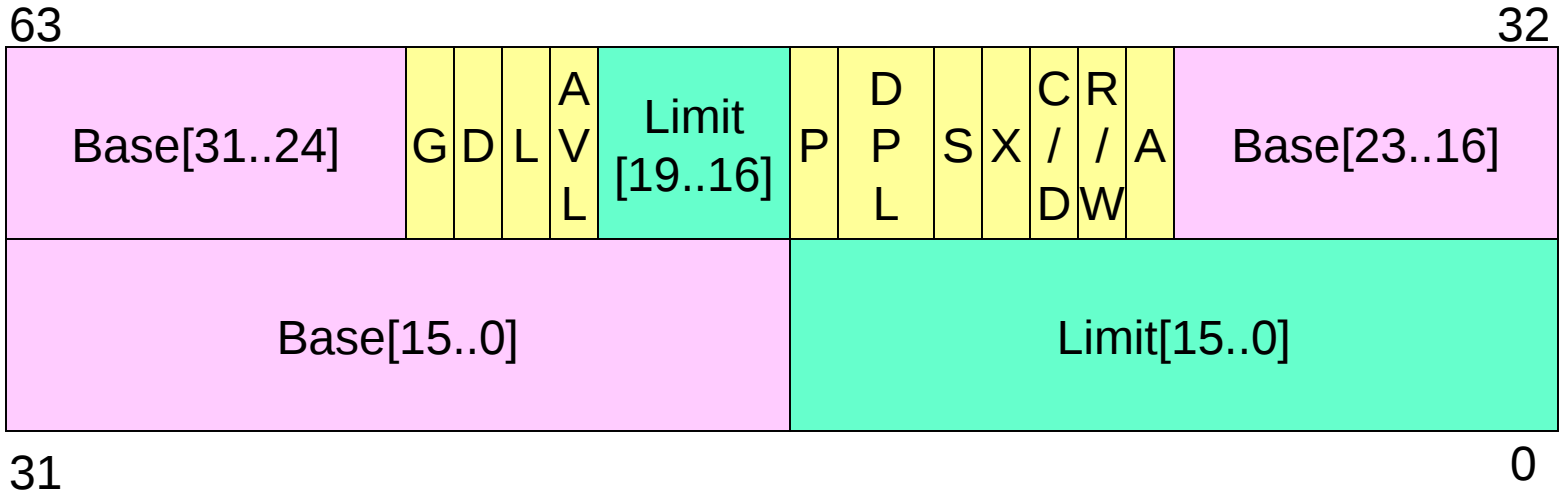
Special segment-registers

How to utilize registers FS and GS for addressing memory-operands while in IA-32e mode

Code/Data Descriptor-Format



Legend:

G = Granularity: (0=byte-granularity, 1=page-granularity)
D = Default operand and address size: (0=16-bits, 1=32-bits)
L = Long (0=compatibility mode code/data, 1=64-bit code)
AVL = Available (this bit can be used by programmers for any purpose)

NOTE: These descriptors can only store 32-bits as a segment's base-address

IA32_FS_BASE

- This Model-Specific Register is now a part of the official Intel Architecture
- It provides a “back door” to the hidden part of segment-register FS
- It allows use of a 64-bit base-address for segment-register addressed by FS when the CPU is executing in 64-bit mode
- Its MSR register-index is 0xC0000100

Loading register FS

- In “compatibility” mode the upper 32-bits of the “hidden” 64-bit segment base-address for FS will be disregarded by the CPU for computing memory-operand addresses
- But in 64-bit mode the full 64-bit value of the “hidden” FS segment’s base-address will be used in forming effective addresses
- How does FS segment-base get loaded?

It depends on cpu mode

- In “compatibility” mode, we continue to put segment-sectors into register FS:

```
mov $sel_fs, %ax
mov %ax, %fs
```

- This loads the bottom 32-bits into FS from the Global (or the Local) Descriptor Table
- The upper 32-bit are unmodified, and are disregarded, in “compatibility” mode

Use ‘wrmsr’ in 64-bit mode

- In 64-bit mode, the ‘wrmsr’ instruction can be used to load the full 64-bits of segment base-address into the “hidden” part of the FS segment-register:

```
mov    base_lo32, %eax
mov    base_hi32, %edx
mov    $0xC0000100, %ecx
wrmsr
```

GS is similar

- The foregoing remarks about register FS also apply to register GS
- There is a 64-bit IA32_GS_BASE register that is accessed with ‘rdmsr’ and ‘wrmsr’
- Its MSR register-index is 0xC0000101

The ‘swapgs” instruction

- There is a third Model Specific Register that gets used (in 64-bit mode) with the IA32_GS_BASE register, officially named the IA32_KERNEL_GS_BASE register
- Its MSR register-index is 0xC0000102
- A special instuction can be used by ring0 code to exchange the contents of these two Model-Specific Registers

In-class exercise

- Use our ‘newapp64.cpp’ development-tool to quickly create the boilerplate code for a boot-time program that takes the CPU into its IA-32e mode (where you can try some experiments with registers FS and GS, as well as the privileged ‘swapgs’ instruction)
- Can you “map” the topmost page-frame to video-memory, then use IA32_FS_BASE to write a message to screen-memory?