CS:3620 Operating Systems

Mechanism of Address Translation

Recap: Virtual Address Space

- Virtual address space: every process assumes it has access to a large space of memory from address 0 to MAX
- Contains program code (and static data), heap (dynamic allocations), and stack (used during function calls)
  - Stack and heap grow during runtime
- CPU issues loads and stores to virtual addresses
  - Memory management unit (MMU) to translate VA to PA

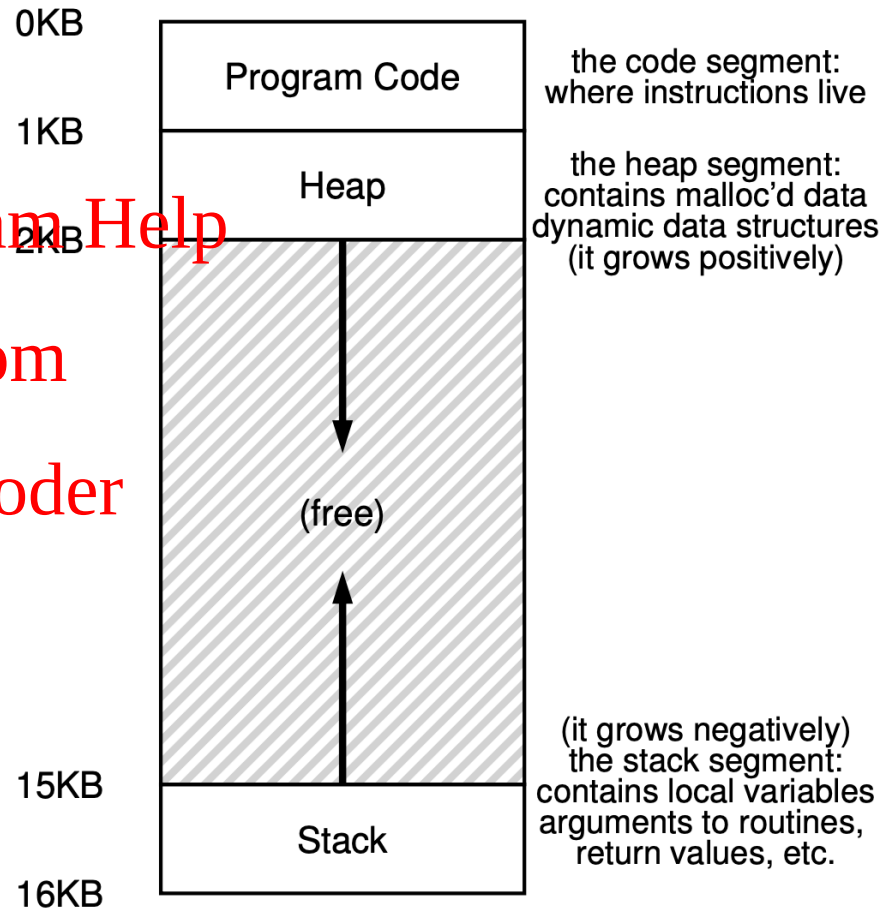


Figure 13.3: **An Example Address Space**

A simple example

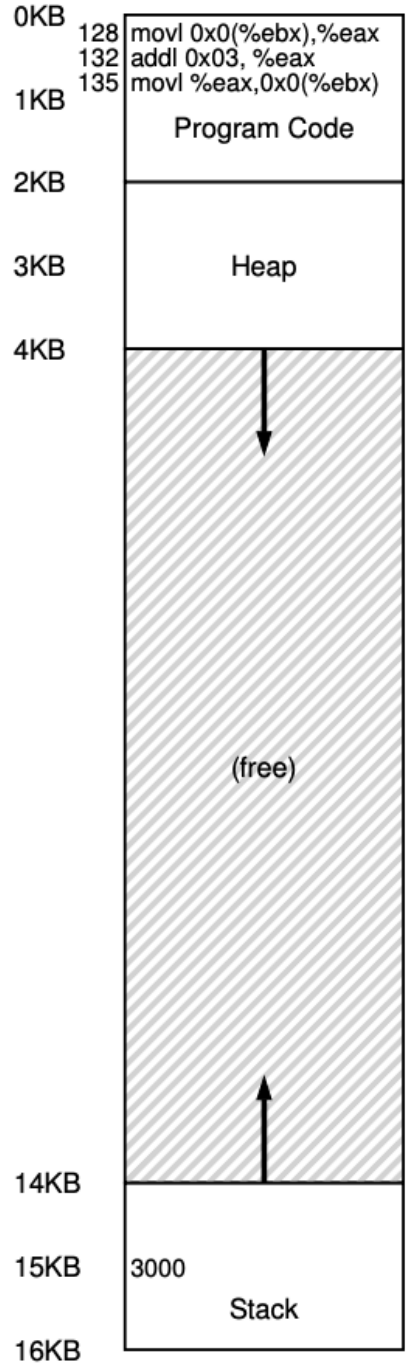
- Consider a simple C function

```
void func() {
    int x = 3000;
    x = x + 3;
    ...
```


- It is compiled as follows


```
128: movl 0x0(%ebx), %eax     ;load 0+ebx into eax
132: addl $0x03, %eax         ;add 3 to eax register
135: movl %eax, 0x0(%ebx)     ;store eax back to mem
```

- Virtual address space is setup by OS during process creation

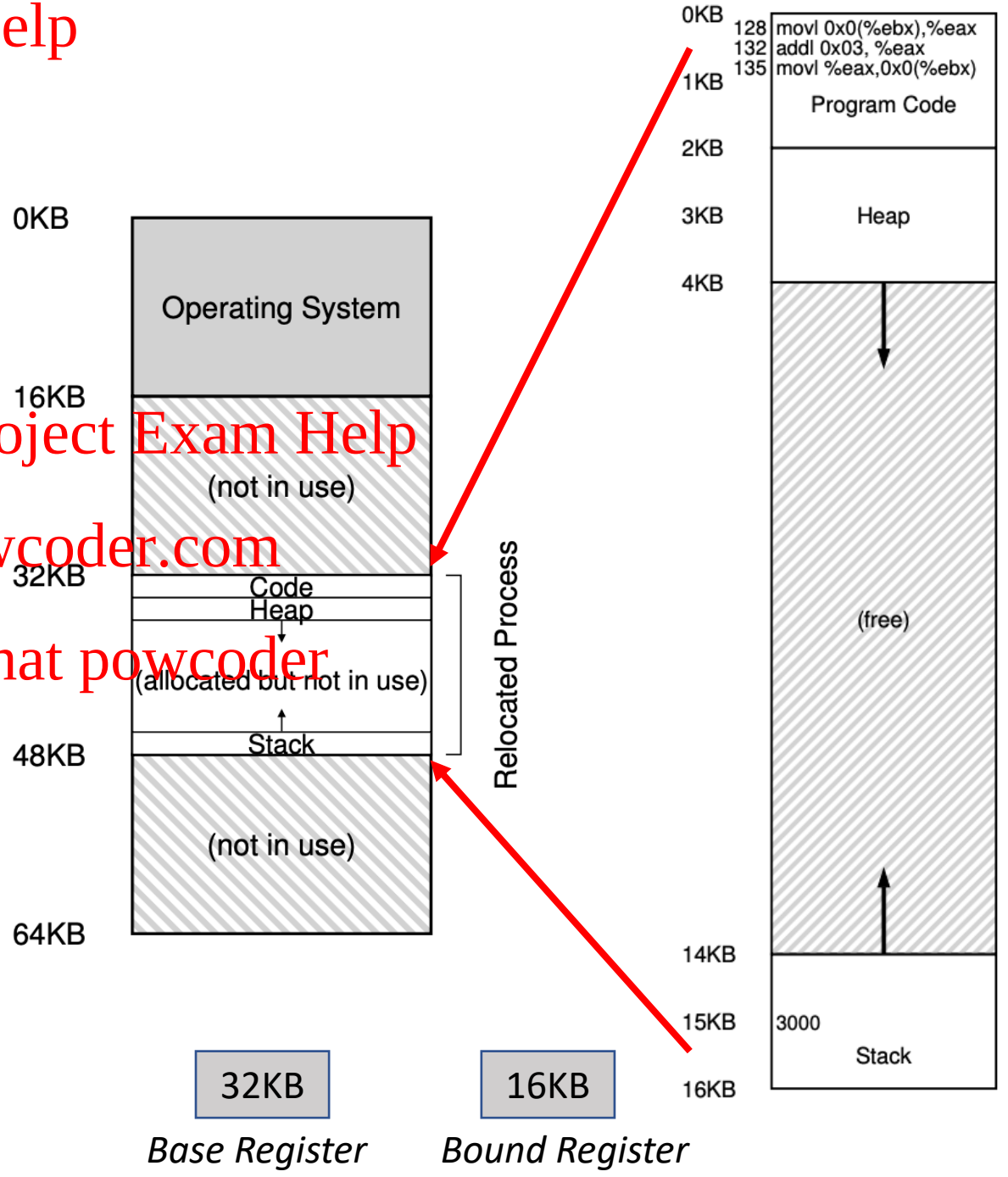


Address Translation

- An easy solution: Base and bound registers
  - Places entire memory image in one chunk


```
physical address = virtual address + base
```


- Need the following translation from VA to PA
  - 128 to 32896 (32KB + 128)
  - 1KB to 33 KB
  - 20KB? Error!



Who performs address translation?

- In this simple example, OS tells the hardware the base (starting address) and bound (total size of process) values
- Memory hardware, Memory Management Unit (MMU), calculates PA from VA
- MMU also checks if address is beyond bound
- OS is not involved in every translation

Role of hardware in translation

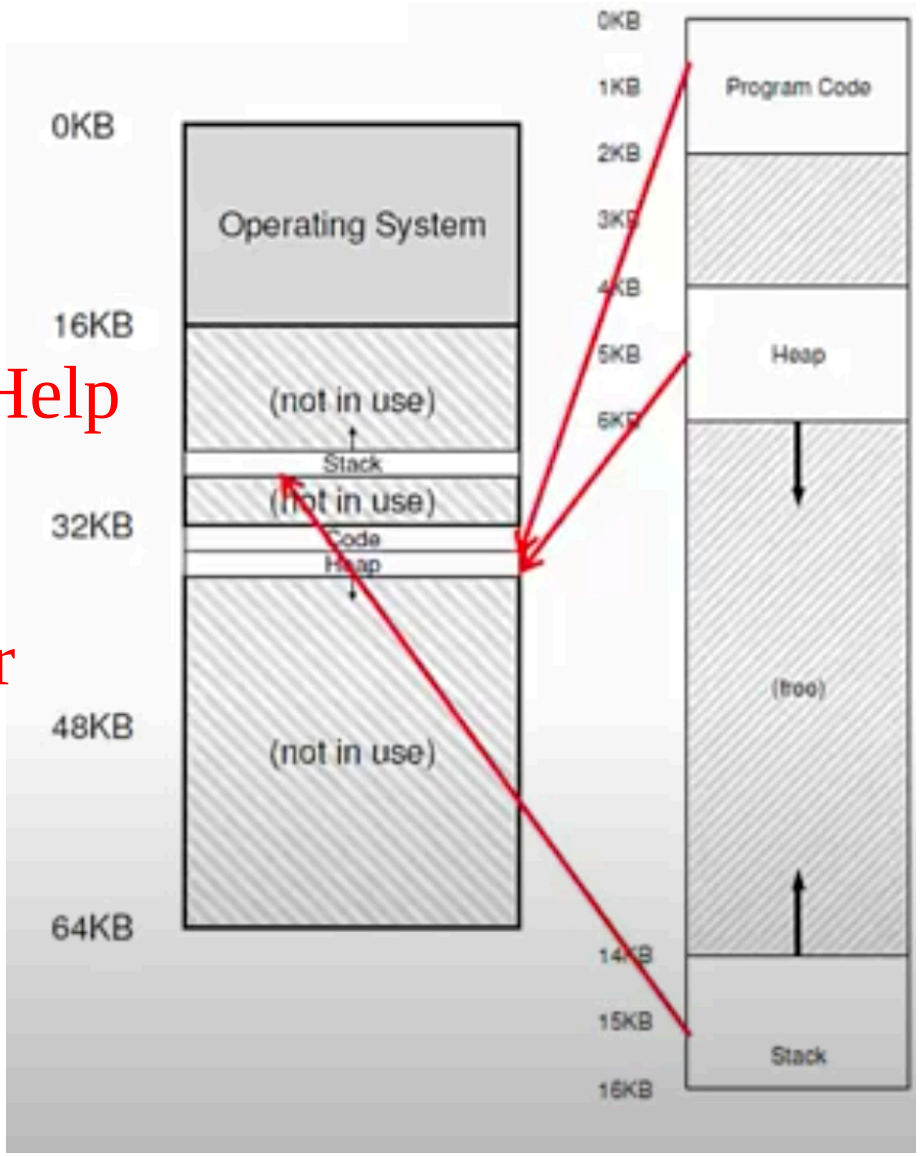
- CPU provides privileged mode of execution
- Instruction set has privileged instructions to set translation information (e.g., base, bound)
- Hardware (MMU) uses this information to perform translation on every memory access
- MMU generates faults and traps to OS when access is illegal (e.g., VA is out of bound)

Role of OS in translation

- OS maintains free list of memory
- Allocates space to process during creation (and when asked) and cleans up when done
- Maintains information of where space is allocated to each process (in PCB)
- Sets address translation information (e.g., base & bound) in hardware
- Updates this information upon context switch
- Handles traps due to illegal memory access

Segmentation

- Generalized base and bounds
- Each segment of memory image placed separately
- Multiple (base, bound) values stored in MMU
- Good for sparse address spaces
- But variable sized allocation leads to external fragmentation
  - Small holes in memory left between segments



Disclaimer

- *These lecture slides are based on a slide set by Youjip Won (Hanyang University) and Mythili Vutukuru (IIT Bombay)*