Address Translation

Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

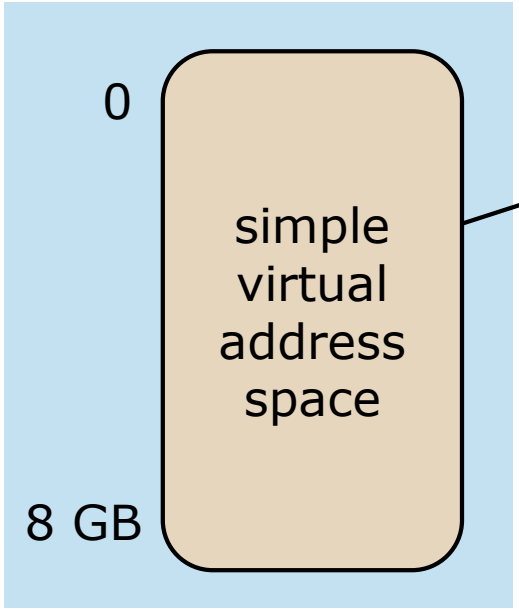
- Simulate the working of the base-and-bounds translation scheme
- Explain how base-and-bounds addresses the design goals of memory management

"Base-and-bounds" also known as "dynamic relocation"

Review: address translation

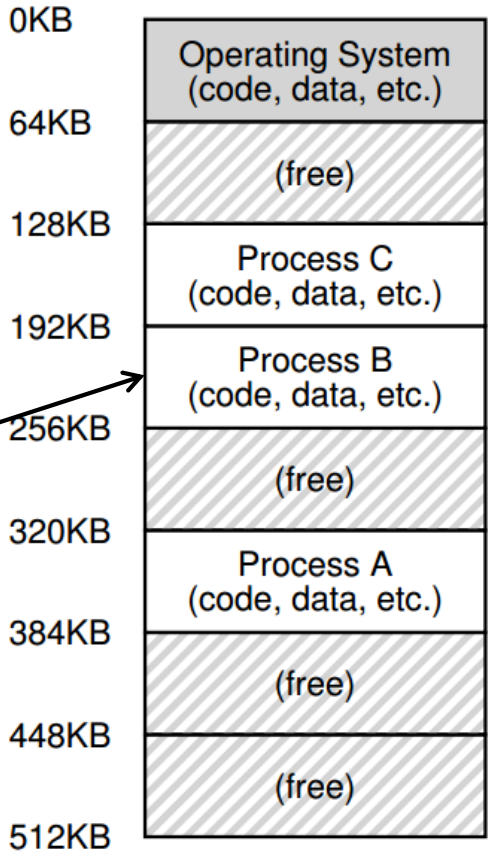
Physical memory is shared by the OS and many user processes

Each user process only sees its own simple address space

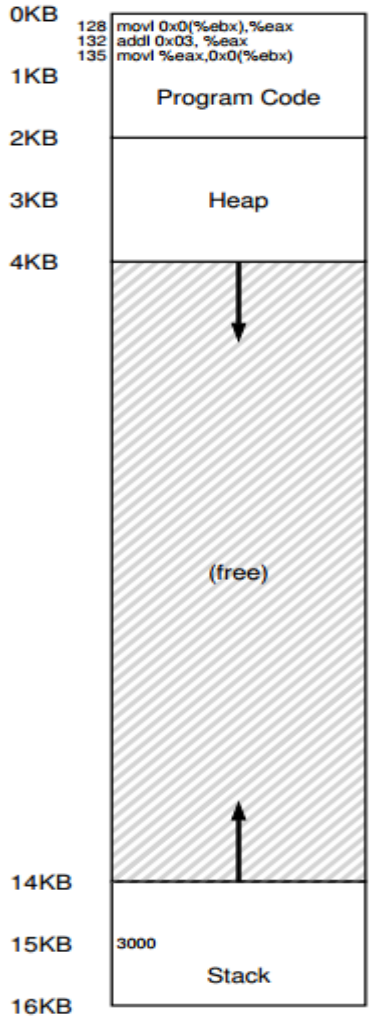


(8 GB just an example)

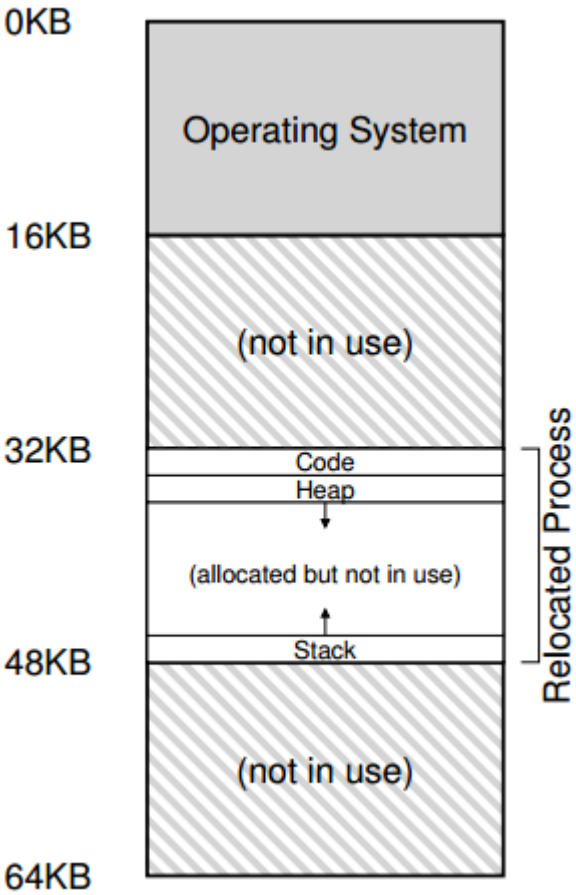
shared physical address space



Detail on virtual vs. physical memory



a process's virtual memory



the system's physical memory

Example address translation

Example code: (initialize a variable x to 3000, then add 3 to x)

```
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
```

Initially the PC (program counter) register has value 128, and the ebx register has value 15KB.

"virtual execution":

1. Instruction is fetched from virtual address 128
2. As this instruction is executed, the value 3000 is loaded from the virtual address 15KB
3. …

Example address translation, cont'd.

```
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
```

Initially the PC register has value 128, and the ebx register has value 15KB.

execution with memory translation:

1. Address 128 in PC is translated to a physical address, say 32896 – then instruction is fetched from there
2. During execution of the instruction, the virtual address 15KB is translated to a physical address, say 47KB, and the value 3000 is loaded from this physical address
3. …

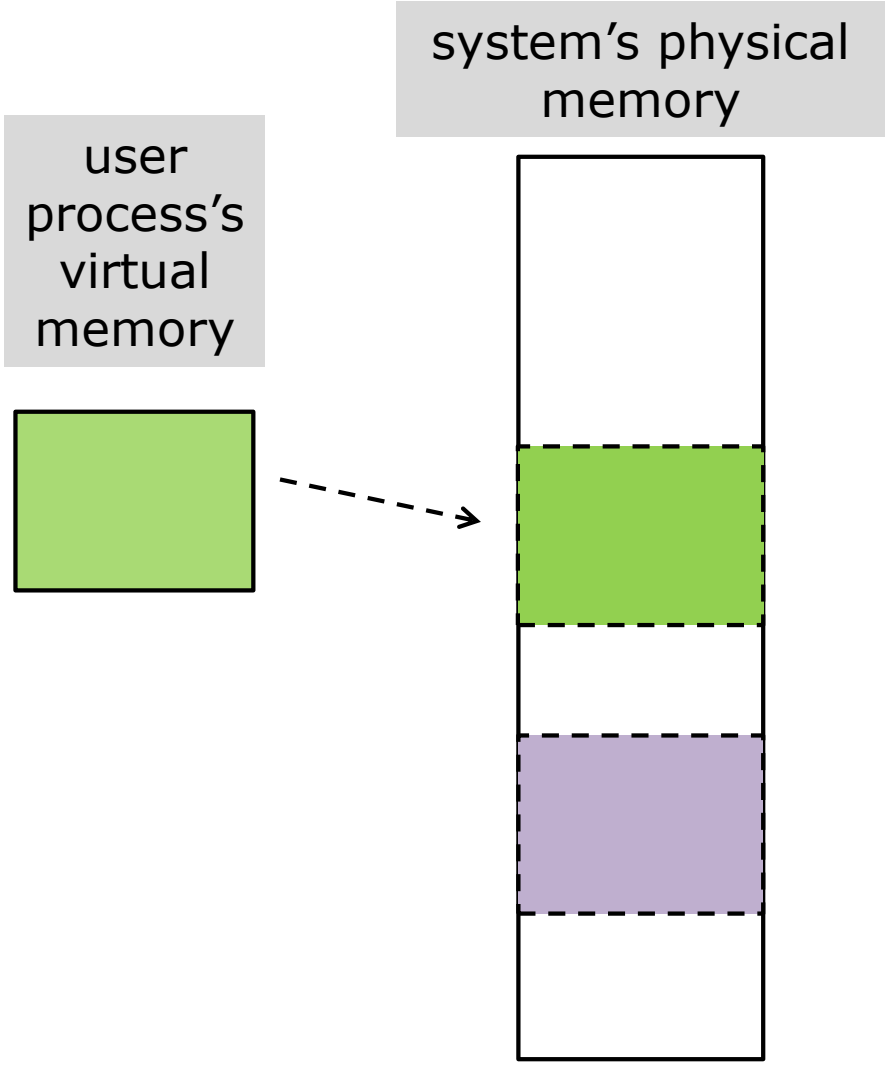
Design goals for address translation

- efficiency
- protection
- clean memory abstraction

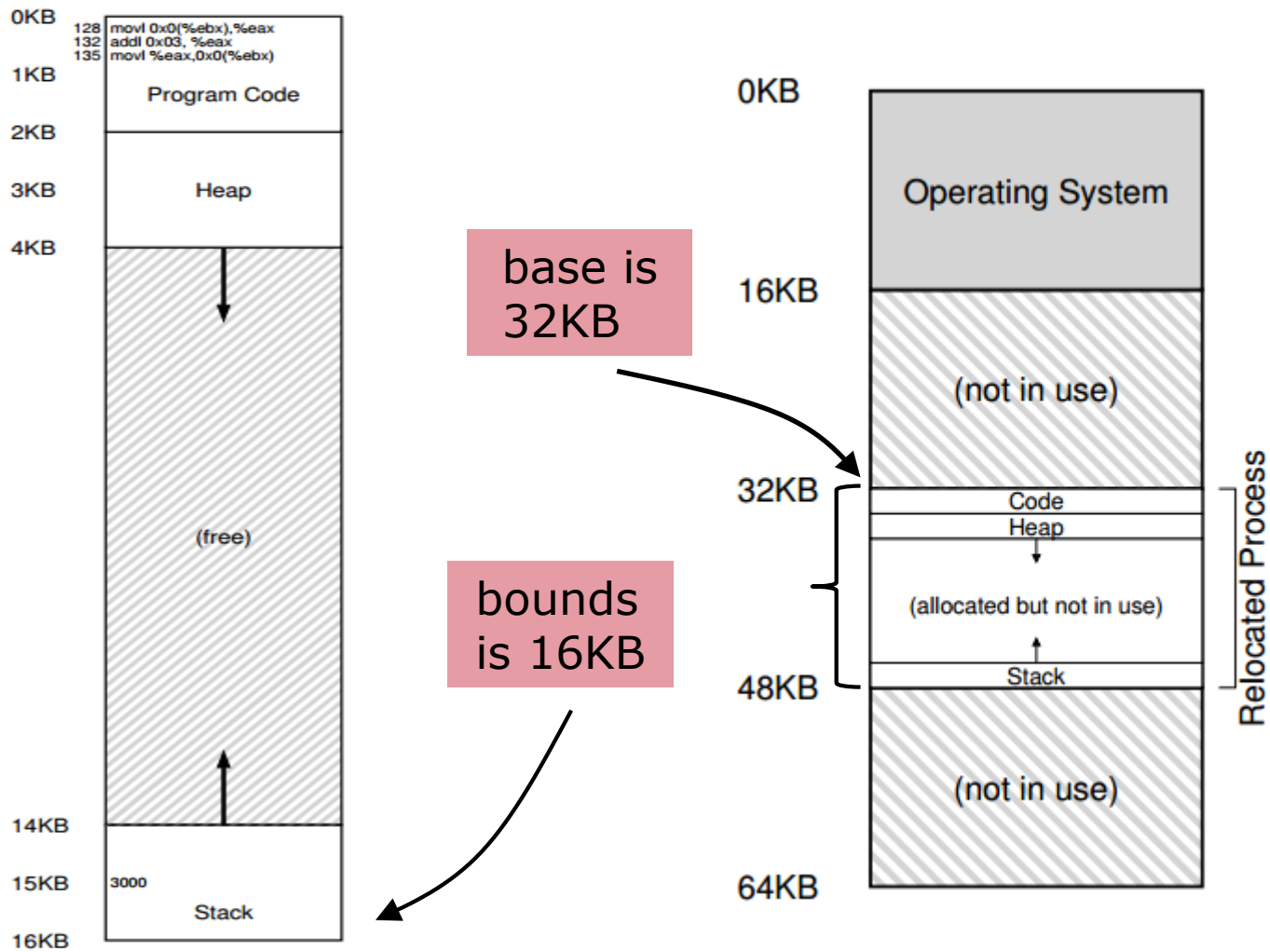
Assumptions

For now, we'll assume:

1. each process's address space is contiguous in physical memory
2. each process's address space is smaller than physical memory
3. all user virtual address spaces are the same size



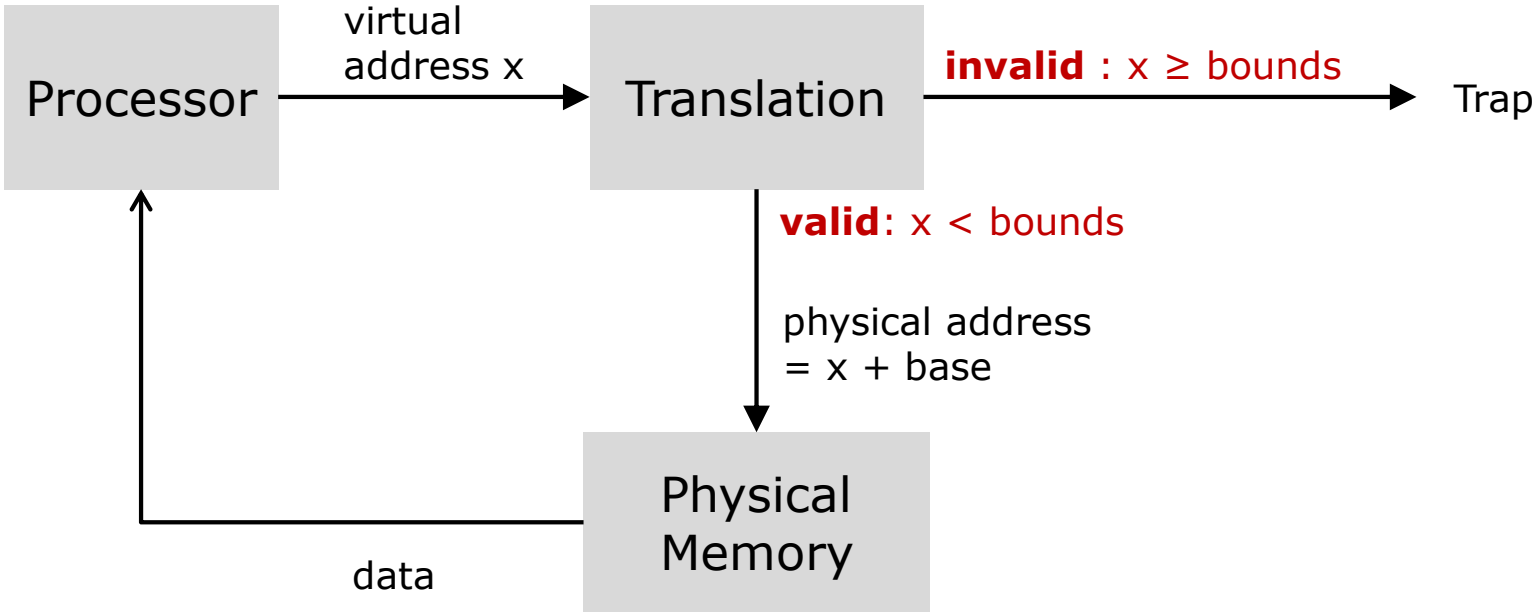
Base and bounds addressing



a process's virtual memory

the system's physical memory

Base and bounds address translation



- Each process has base and bounds values
- base: physical address of virtual address 0
- bounds: size of the virtual address space
- MMU has base and bounds registers
- before OS runs a process: it puts the process' base and bounds values into the registers

Base and bounds example

Suppose base is 16KB and bounds is 4KB

Virtual address	Physical address
0	16 KB
1 KB	?
3 KB	?
5 KB	?

physical address = virtual address + base

Exercise

How does base-and-bounds addressing the design goals of:

- efficiency?
- protection?

Hardware support

- base and bounds registers
  - part of the Memory Management Unit (MMU)
- privileged CPU instructions to set base and bounds registers
- CPU must be able to generate exceptions (traps)
  - memory faults
  - attempts by user programs to modify base/bounds registers
- privileged CPU instructions to set exception-handling code

OS duties

- at process creation: find physical memory for the process
- at process termination: reclaim physical memory used by the process
- when a context switch occurs:
  - save base and bounds register values of process that is stopped (save in process control block)
  - restore base and bounds register values of process to be run
- optionally: move a process' virtual address space to a different RAM location
- at boot time: install exception handlers

this is a memory allocation problem, like malloc() has to solve

Questions

- Does address translation happen in software or hardware?
- Can the location of a process' address space in memory change after the process starts running?
- What is the name of the hardware component used to implement virtual memory?

Summary

- Speed and protection are key requirements for memory management
- Base-and-bounds (aka dynamic relocation) is a simple scheme for memory management
- Hardware support in base-and-bounds includes a base register and a bounds register