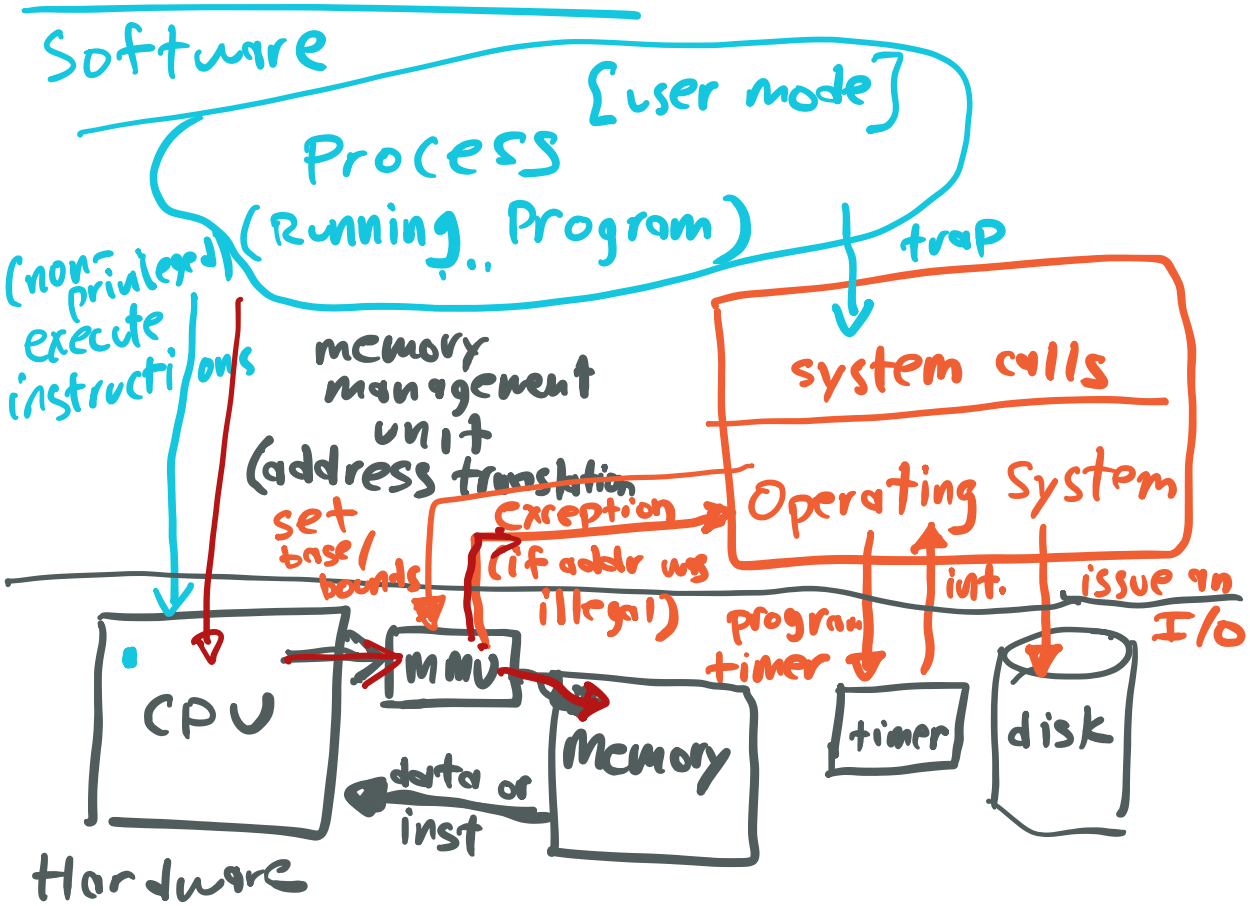
Lecture 9/23



MMUs : simple, more complex, painfully complex

(dynamic relocation)

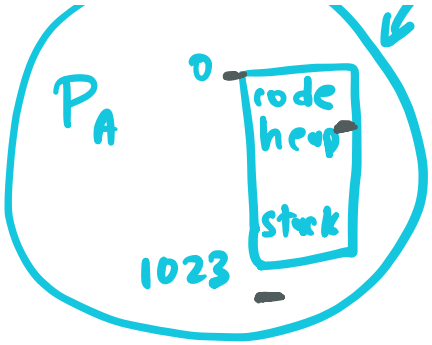
base/bounds:

2 registers

multi-level page table

MMU:
base
bounds

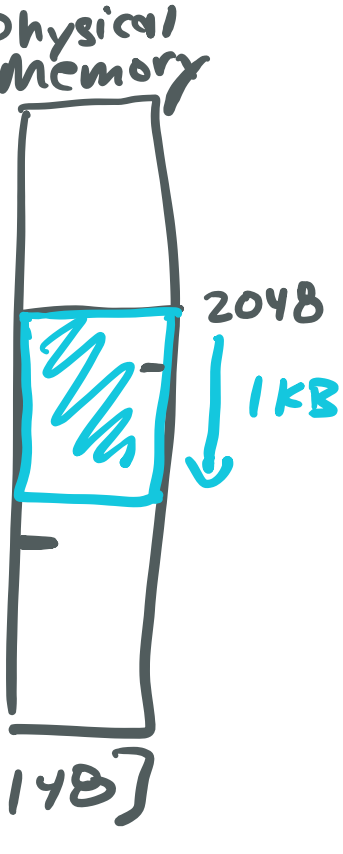
1KB → virtual



address space (illusion)

base: 2048

bounds: 1024 (size)



address translation:

process => virt. addr.
e.g. 100

physical address:
base + virt addr
=> 2048+100 => [2148]

bounds: protection

VA: 1100

check: if (VA < bounds) and (≥ 0) => OK PA, do access

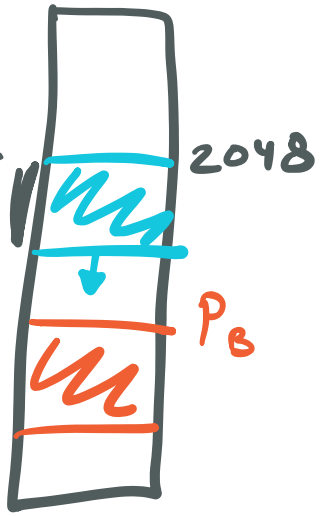
if not, raise exception

OS role:

→ when process is created:
→ allocate phys. memory for AS (contiguous)

(Process List:
P_A: base 2048) (and bounds)

Free List:
which parts of phys mem are free



when process runs:
before, set base/bounds on MMU
then, run process

when context switch:
P_A running
Timer interrupt
want to run P_B [need to set base/bounds for P_B]

when process wants to grow:
system address space
→ call: ask OS for more memory
→ sure:
→ update its free list
→ change the bounds to new size

Pros / Cons

Pros

fast

simple

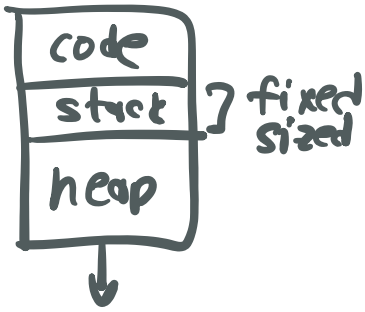
protection ✓

Cons

Inflexible

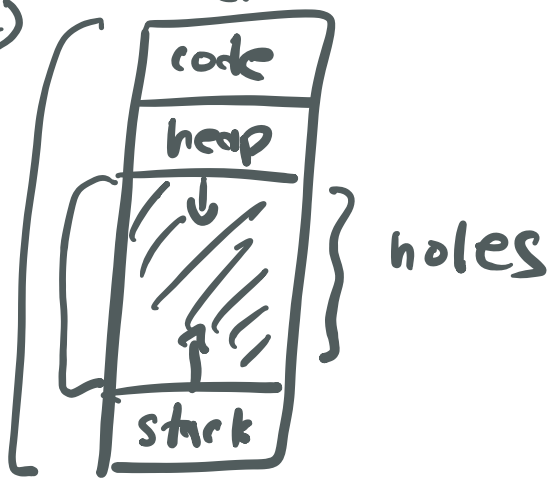
Doesn't support large "sparse" virt. address space

reality:



⇒

desire:



Discussion: Boo

AMA (about class project, whatever)

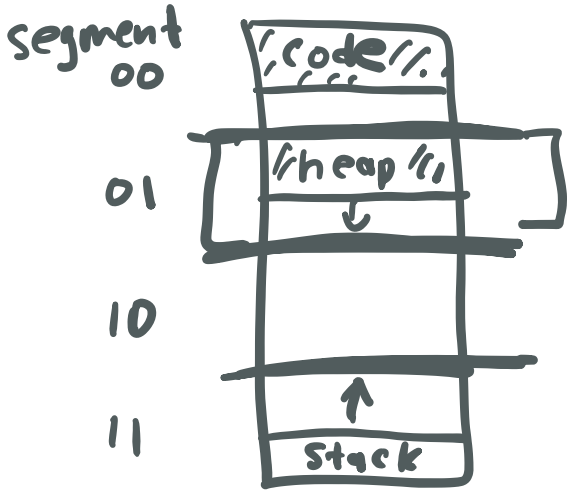
⇒ Projects? [A lot]

Thanks Jim!

MMU: segmentation, paging

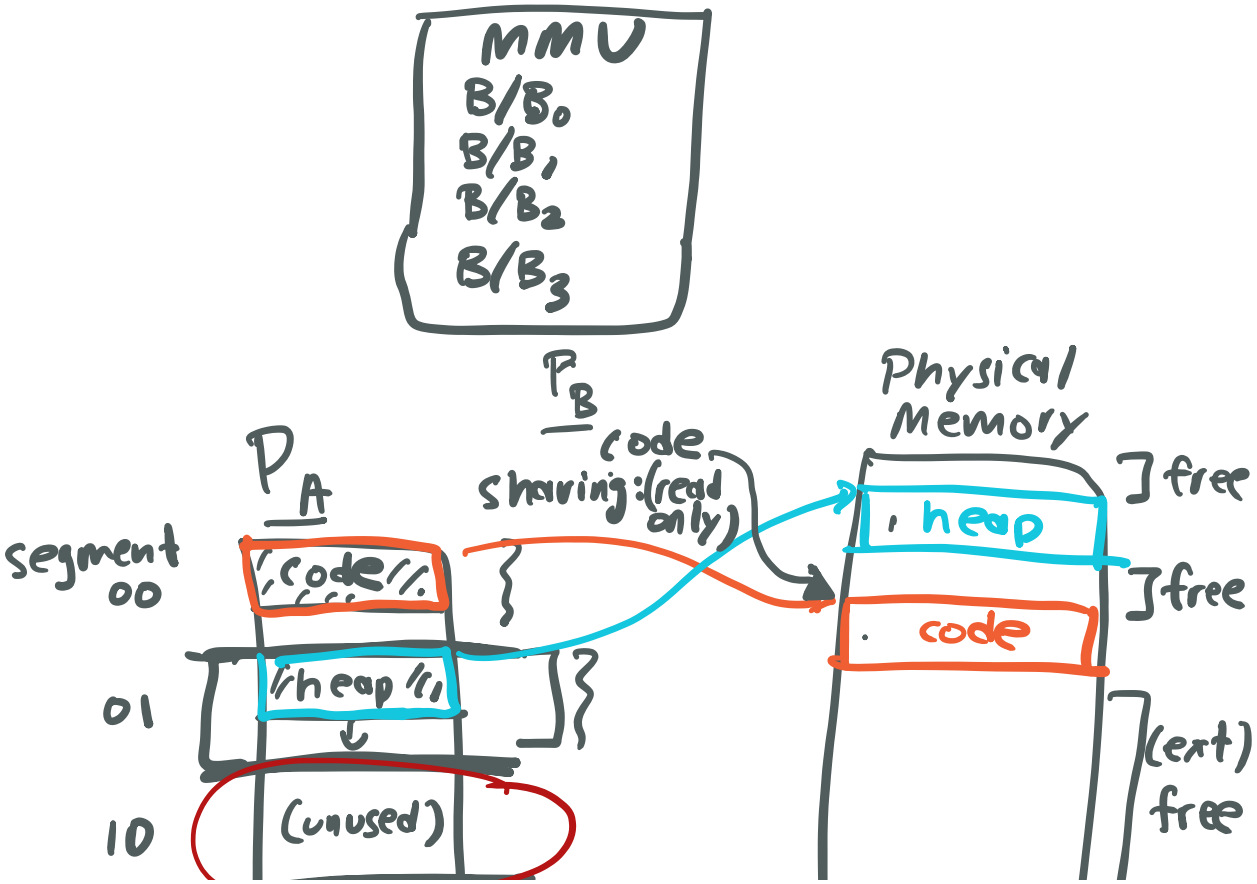
(generalization of base/bounds)

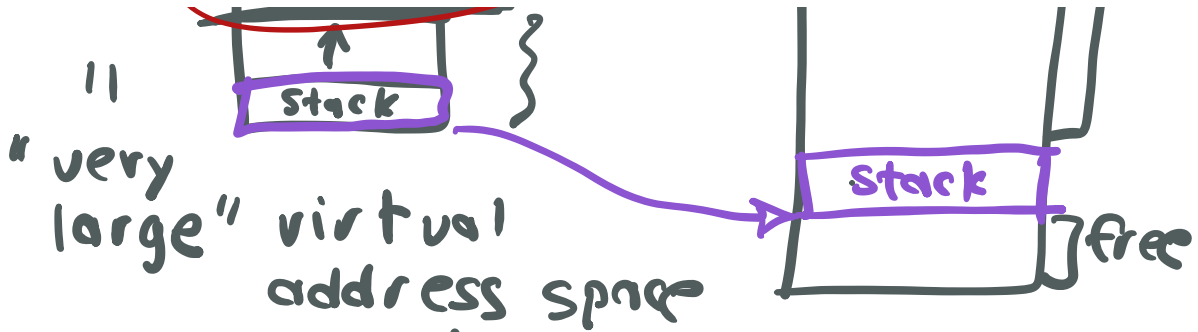
virtual address space (VAS)



segment

for each segment, <base, bounds> pair

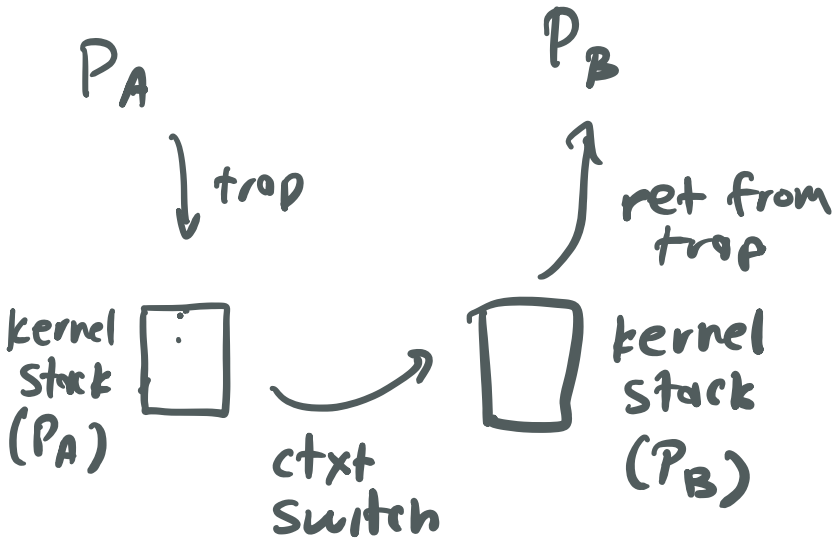




"very large" virtual address space now possible

(external) fragmentation: => memory is hard to manage

internal fragmentation:



Discussion : Begin!

Project

Plb:
lock?

control-a
x
=> exit
qemu

Lecture

~~Test Questions~~
~~(All old are available)~~
~~=> internet~~
→ instructor
→ broad
4xx, 5xx, 6xx

=> malloc

Life

~~Favorite Candy Bar~~ : Reese's

~~Pros/Cons~~
~~Grad School~~

~~Classes~~

Emacs

~~VIM~~

---

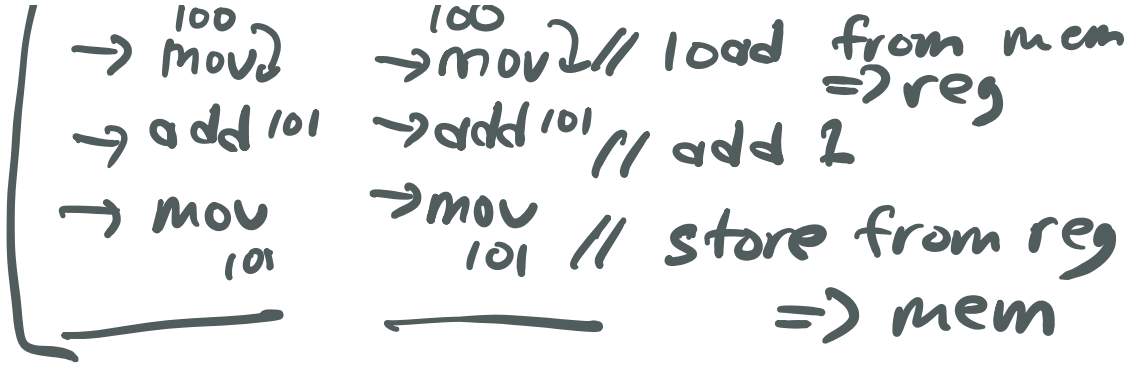
int counter = 0;

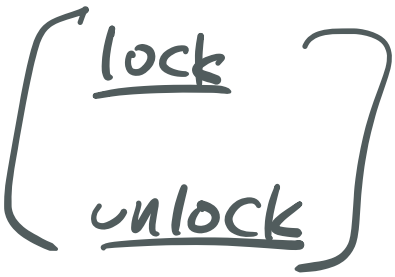
100

P_1 CPU_0

P_2 CPU_1

counter++;


100
mov
100
mov // load from mem
=> reg
add 101
add 101 // add 1
mov
101
mov
101 // store from reg
=> mem


lock
unlock