Lecture 4:

Intro to Mem Mgmt

(Reloc, Segment)

Last Time: CPU virtualization
(private registers)
mechanism: Limited Direct Execution
(efficient control)
Policies: many possible → [MLFQ]

} continue building our illusion

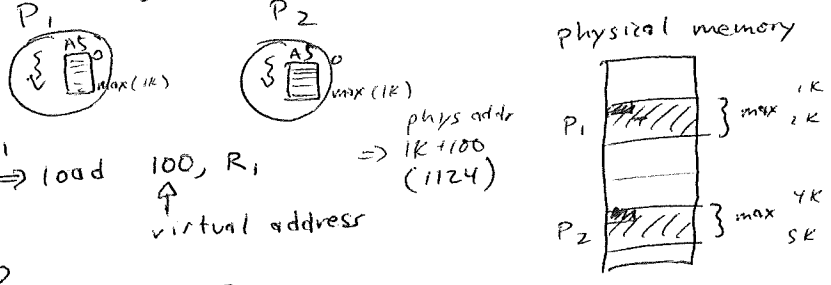
Today: Memory Virtualization
extend LDE to include memory references
⇒ mech: address translation
goal: private address space per process
(efficient but controlled) (also, transparent)

motivate AS
(heap ↓ stack ↑ get them to figure this out)

(virt addr ⇒ phys addr)

Address Translation: need h/w again
start simple, grow

Technique: Dynamic Relocation (Base/Bounds)



P1
⇒ load 100, R1
↑ virtual address
⇒ phys addr = 1K + 100 (1124)

P2
⇒ load 100, R1
↑ virt. addr ⇒ PA: 4K + 100 (4196)

2 new h/w registers (per CPU)
base: holds physical address of beginning of AS
bounds (limit): holds size of AS

e.g. (P1) load 100, R1
base: 1024
bounds: 1024
} (in CPU's MMU)

to execute: h/w
VA (100) + Base (1024) ⇒ PA (1124)
issues load to 1124,
puts result into R1

thus: h/w translates every mem reference (from virt → phys)

what is bounds for? protection
h/w: check if addr w/in bounds
if not, raise exception ⇒ (kill process)

When is OS involved?
new process creation: e.g. add P3
growth of addr space:
} must track free mem ⇒ free list

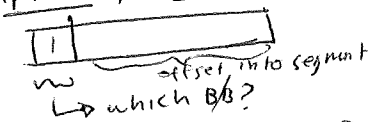
on context switch:
save base/bounds of P1, restore b/b of P2

Pros/Cons:
+ fast + simple
+ little mem overhead (2 regs)
- not flexible
- wastes memory (large AS)

Segmentation: general base/bounds
base/bounds pair per segment of AS

segment: contiguous region of AS, defined by (ptr, length)
usually logical portion of AS
example: code, stack, heap
each can be relocated independently!

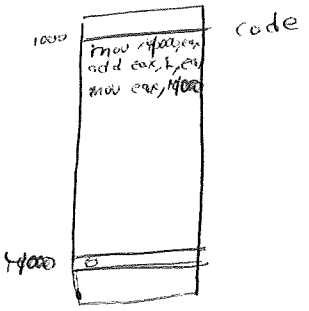
problem: how to determine which base/bounds pair to use?
⇒ implicit, explicit



When is OS involved?
⇒ new process
⇒ growth of seg
⇒ ctxt switch

Pros/Cons:
+ supports sparse AS
+ code sharing
- external fragmentation
(must compact, rearrange mem)
+ fine-grained protection

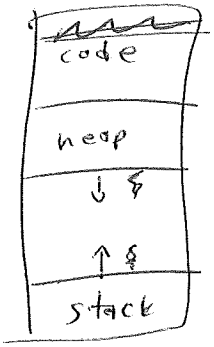
Example: canonical address space

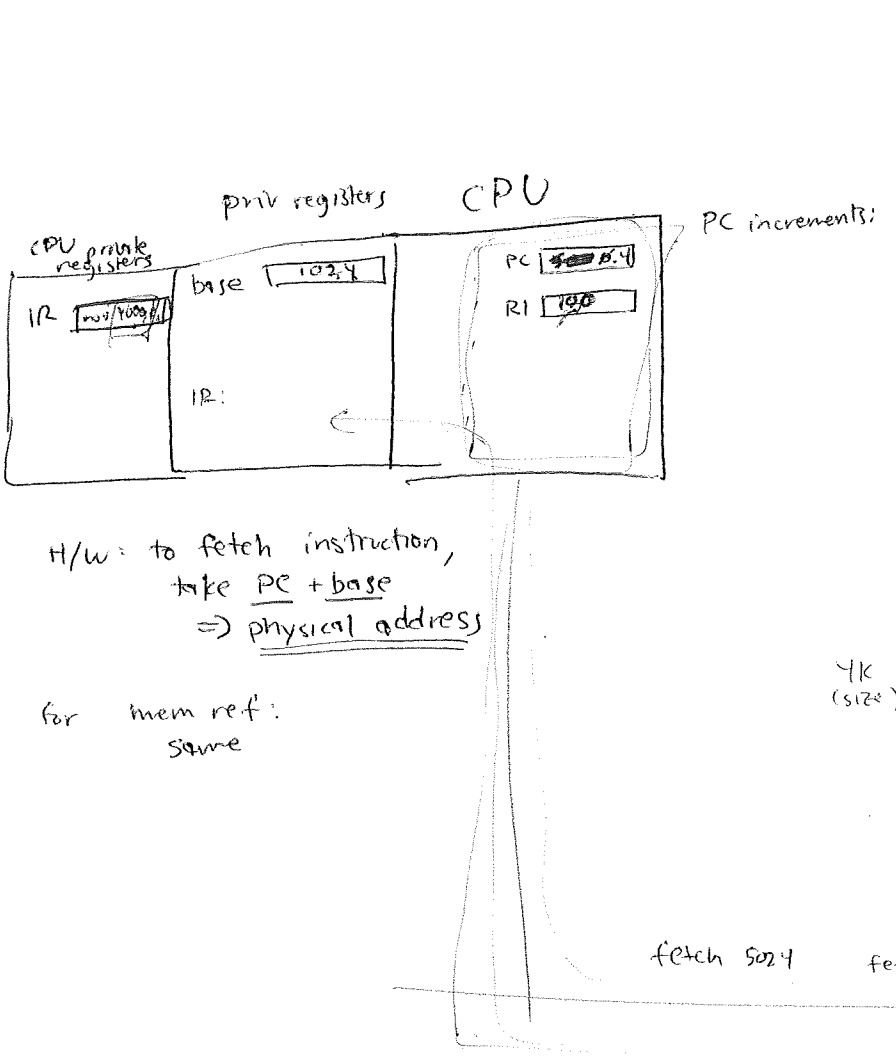


what is stream of memory refs?

fetch 1000
load 4000
fetch 1004
fetch 1008
store 4000

} 5 mem refs





H/W: to fetch instruction, take PC + base
⇒ physical address

for mem ref: same