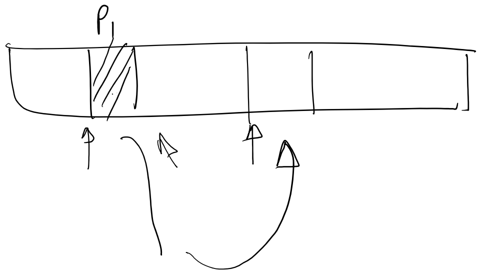
Lecture 16. Memory virtualization contd.

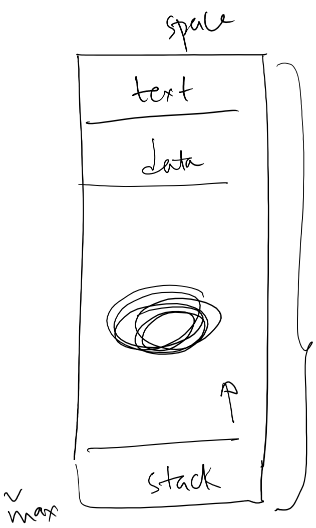
- design 0 : one-to-one v2p mapping
- design 1a. static relocation
  - 1b. dynamic relocation (base + bound) regs.
- design 2. ________

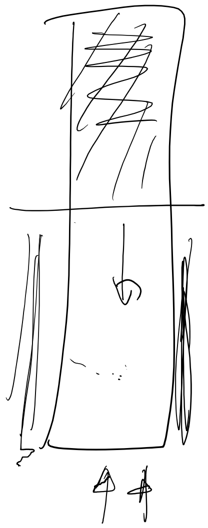
support the address space abstraction w/ an efficient implementation soln.

* assumptions

- VA size < PA size ✓
- all process have equal sizes ✓
- all processes get contiguous allocations ⊛ (virtual)
-



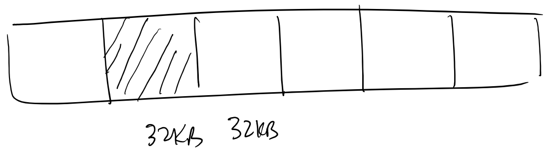




32-bit 4GB

16-bit : ~~16~~ 32 KB

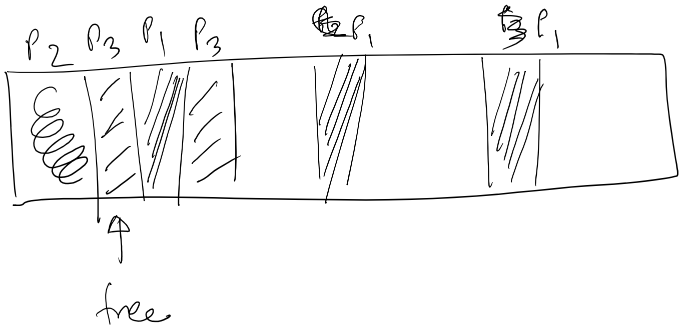
PA range



PA = (VA + base) ——— assumption that this address is valid & belongs to the process.

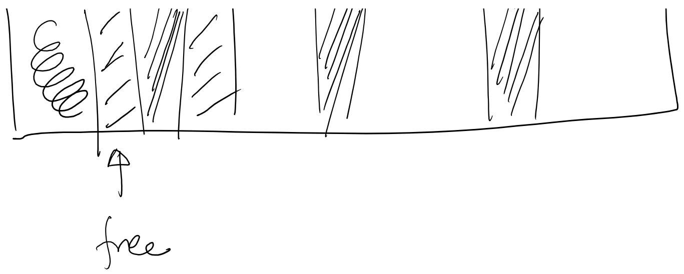
segmentation

PA range



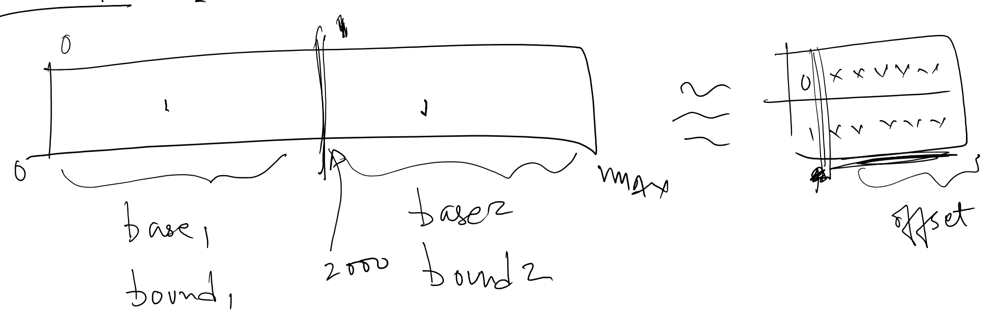
non-contiguous imp. for varying sized programs!

PA range



imp. for varying sized programs!

- use multiple base & bound regs.



PA = (VA + base) ~~ if bound ≥ VA

if VA < bound1
  PA = VA + base1
else if VA ≤ bound2
  PA = VA + base2

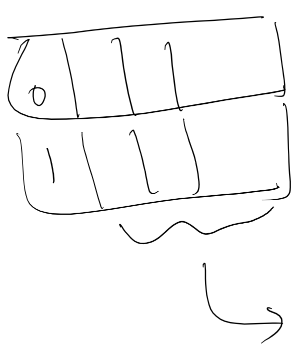
Hw, MMU
- needs to know which VA region
- know the base regs. & bound values.

VA = 2000
↓
PA = 1000 ?

0 — 1999 VA1
2000 — 3999 VA2

b1 = 4000
b2 = 1000

simpler example.



4-bit
↕
16 addresses
0 | 1 ← MSB
8 addr | 8 addr

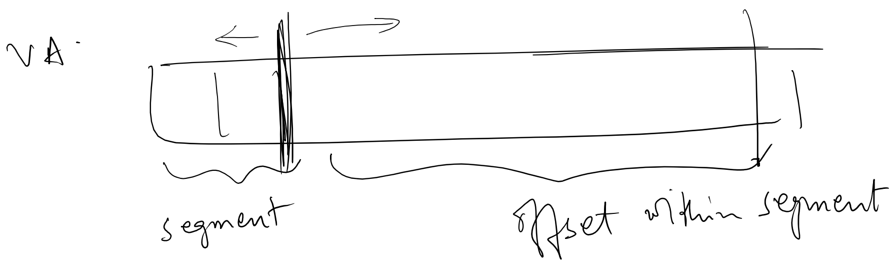
MMU
- lookup MSB bit
- select Base

- select Base
- compute offset
  - (base + offset) for memory access

segmentation

- break / divide VA space into <u>logical</u> segments
  (continuous VA region)

code segment
data segment
stack segment
heap segment
} base & bound <u>regs.</u>



00 — cs
01 — ds
11 — ss
10 — es

choose a segment register