CS 219

Lecture 12.

CS 236

Lab 4

~ dive into xv6!

- xv6 riscV git
- xv6 book
- xv6 source code listing

memory (virtualization)

recap: the process abstraction

- decouple program logic encoding / development from program execution
- CPU virtualization (via context save & restore & scheduling).

process-view

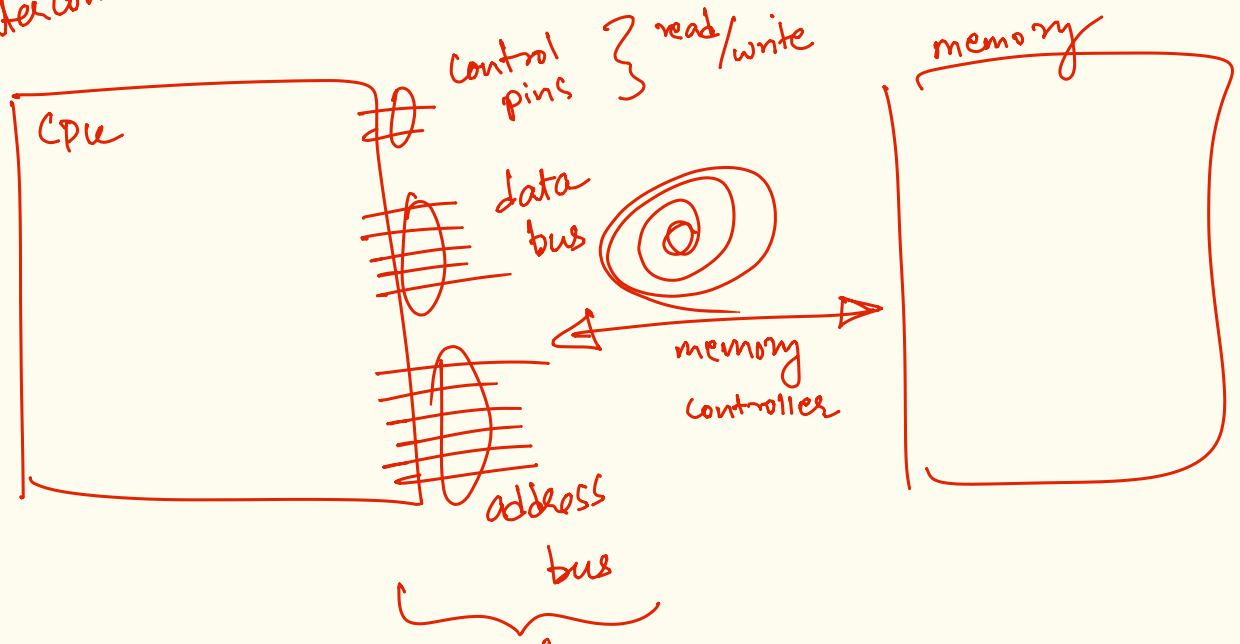
- data objects / variables
- code / functions
- stack ptr.

~ an (independent) / explicit CPU for execution

memory

- (virtualization)
  - addressing
  - mechanisms for allocation
- mgmt.
  - how much?
  - where to allocate?
  - eviction policy?

memory access / interconnect view



- identifying / addressing memory
- address-width — 8-bits, 16, 32, 64 ...
- └ address range ~ how much memory can the CPU address.

address: 0/1 encoding on the address-bus.

0000 0000 0000 1111

0x 000f — 16-bit address!

0x 0000 ... 0x ffff } 2^{16} addresses

(1) why do process need memory?

data — static

code — read only / shared

heap — dynamic

stack — dynamic

PC range

memory allocation/access for processes

requirements.

(i) isolation

(ii) "full" addressability

& zero-starting address range

(iii) transparent, correctness

(iv) efficient.

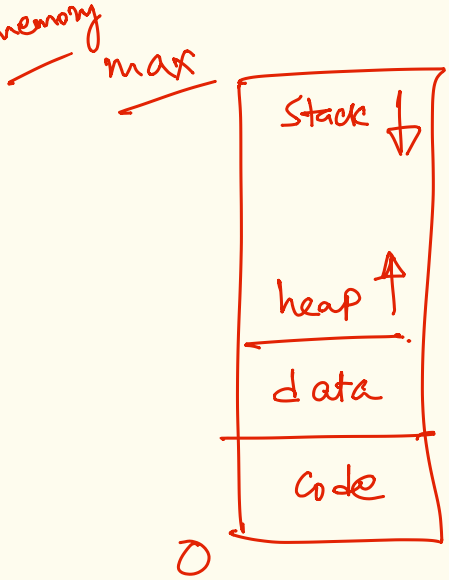
---

design 0

~ similar to the context switch idea

~ extend save & restore to memory!

~ all memory available to single executing process!



+ves: zero-starting

full range

-ves:

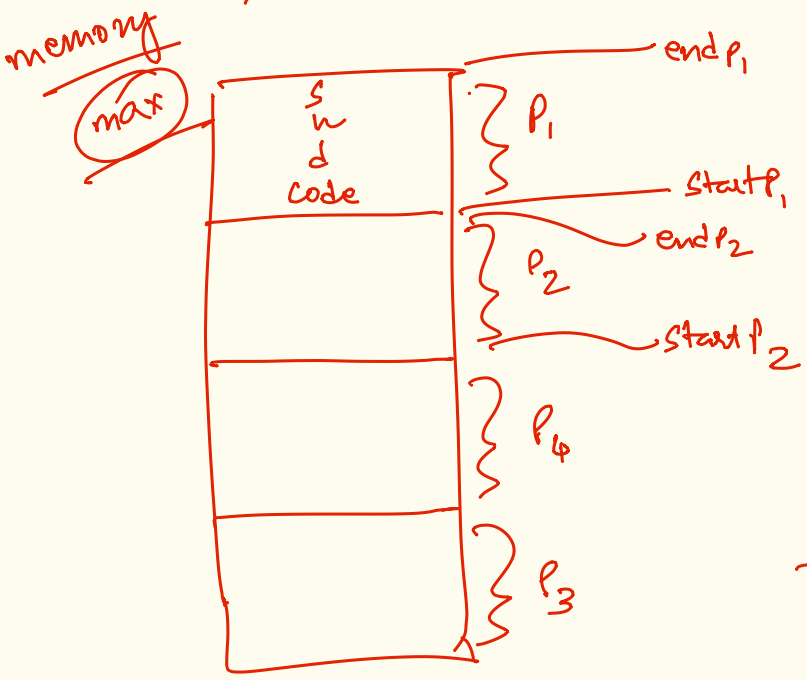
~ potential offload dest^n disk is slow, slow, slow

~ size of context is all memory!

design 1:

~ not all memory may be used at all times!

~ more from temporal-sharing to spatial sharing.



+ves

~ efficiency!

~ multiple processes in memory.

-ves:

- no zero-starting
- limited size
- no dynamic size change possibility.

---

design 2:

~ decoupling of memory addressing from allocation

└ via ISA feature.