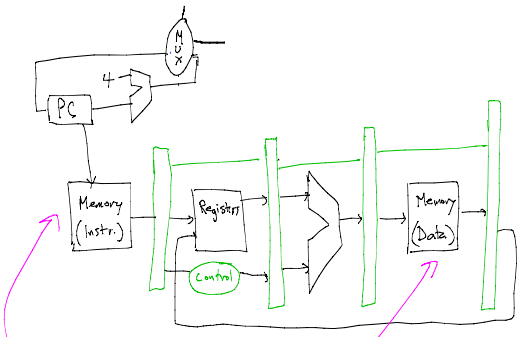
lecture 16

virtual vs. physical memory

- types of physical memory
- paging

Wed. March 9, 2016

MIPS Memory



next three lectures

virtual address

physical address

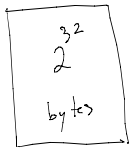
514 - 398 - 3740

cell tower

cell phone

virtual memory
(program addresses)

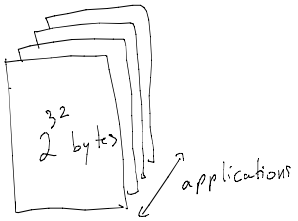
physical memory



- RAM
- disk
- flash
- etc...

virtual memory
(program addresses)

physical memory



"process" (running program)

- RAM
- disk
- flash
- etc...

How do multiple programs share the same (finite) address space?

How to reconcile different sizes of program vs. physical memory ?

Sizes of Memory

$2^{10} \approx 1$ KB (kilobyte)

$2^{20} \approx 1$ MB (megabyte)

$2^{30} \approx 1$ GB (gigabyte)

$2^{40} \approx 1$ TB (terabyte)

$2^{50} \approx 1$ PB (petabyte)

$2^{60} \approx 1$ EB (exabyte)

⋮

Floppy disk (1.4 MB)

(obsolete)

magnetic

CD ~1 GB

DVD ~10 GB

optical (laser)

Hard Disk Drive (HDD)

~1 TB

magnetic

HDD access time

- 100 rotations per second

⇒ average access $\frac{1}{2 \times 100}$ sec = 5 ms

⇒ $2 \times 10^9 \frac{\text{clock cycles}}{\text{sec}} \times \frac{5 \text{ ms}}{\text{access}} = \frac{10^7 \text{ clock cycles}}{\text{access}}$

SLOW!

Flash (SSD - solid state drive)

semiconductor - very fast access 10^{-6} sec

(although it varies between technologies)

USB

Card (digital camera, cell phone)

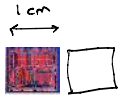
phone, tablet

(flash can replace HDD)

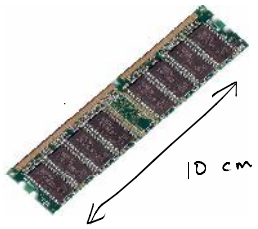
RAM

Semiconductor (silicon)

SRAM
- fast and expensive
- access in one clock cycle (10^{-9} sec)



DRAM
- slower but less expensive
- access in ~10 clock cycles



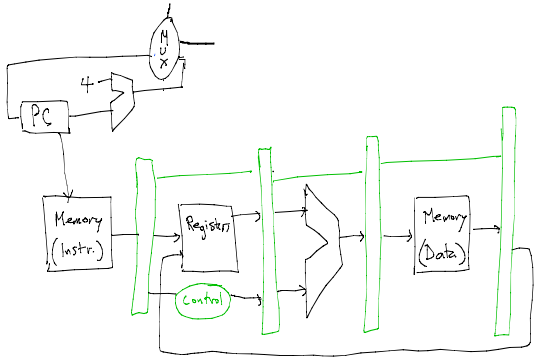
volatile vs. non-volatile

- RAM (SRAM, DRAM)
- flash (SSD)
- disk (HDD, CD, DVD)

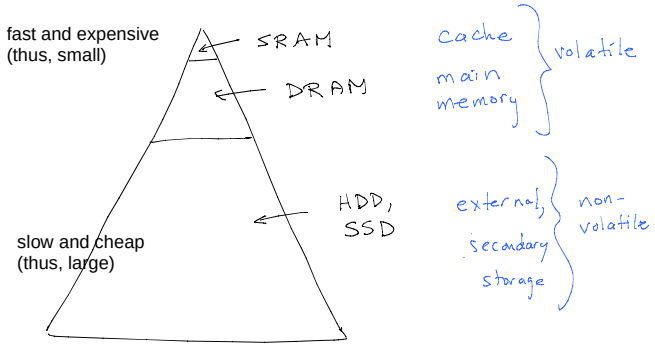
Does storage vanish when power is off?

YES NO

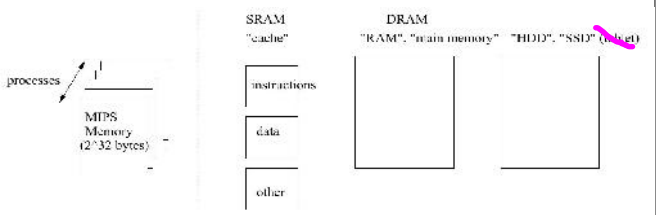
We would like to access Memory in one clock cycle. However, there is a tradeoff between the speed and size of physical memory (can't be large and fast).



Memory Hierarchy



virtual memory physical memory



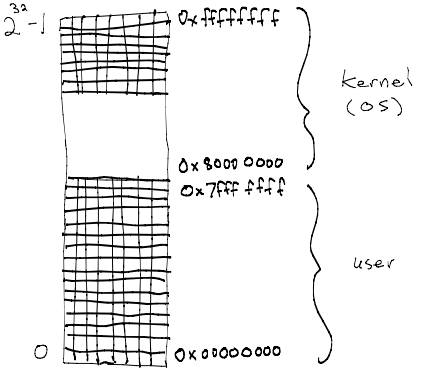
size	~MB	~GB	~TB
number of clock cycles / access	1	~10	10^6

lecture 16

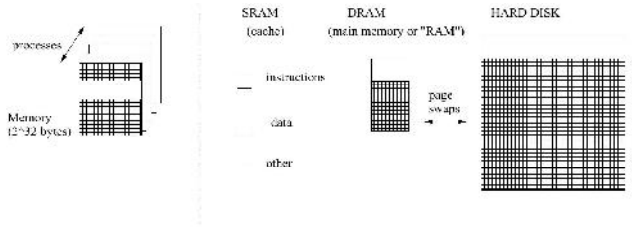
virtual vs. physical memory

- types of physical memory
- paging
  - how to translate (map) virtual to physical ?
  - page tables
  - page fault and page swap

Paging



Q: How to translate a virtual address to a physical address?



Note that both the user part of Memory and (part of the) kernel part of Memory is paged.

Example: suppose 1 page = 2^12 bytes

How many pages?

Virtual Memory

(4 GB = 2^32 bytes)

--> 2^32 / 2^12 = 2^20

Physical Memory

RAM
(e.g. 1 GB = 2^30 bytes)

--> 2^30 / 2^12 = 2^18 pages

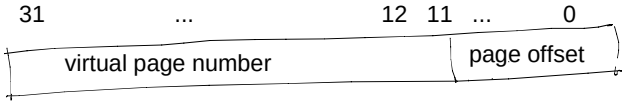
HDD
(e.g. 1 TB = 2^40 bytes)

--> 2^40 /2^12 = 2^28 pages

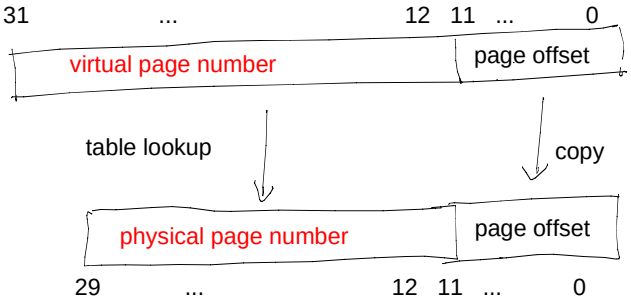
How to translate (map) a virtual address to a physical address ?

A virtual address is 32 bits. These are the program addresses we have been talking about for the last few weeks.

Again, take example that a page is 2^12 bytes

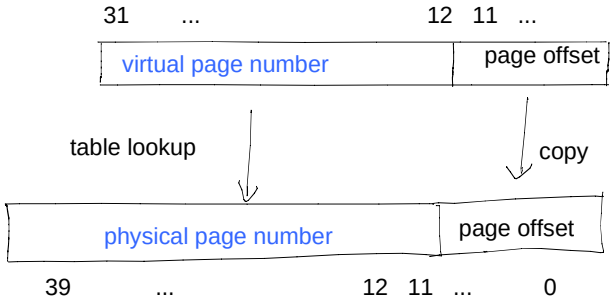


virtual address



physical address (RAM) e.g. 1 GB = 2^30 bytes

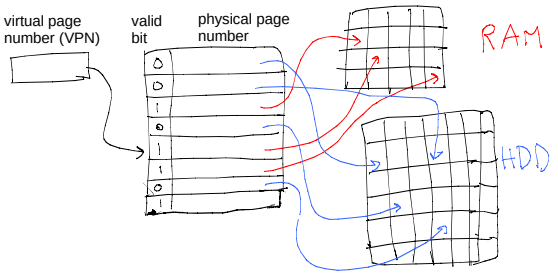
virtual address



physical address (HDD) e.g. 1 TB = 2^40 bytes

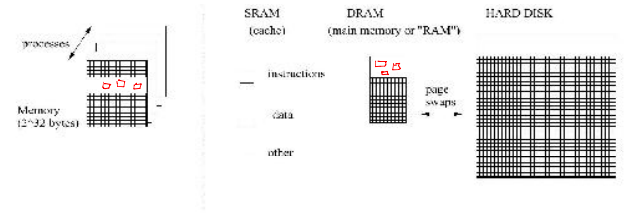
"Page table"

Data structure in kernel that translates (maps) a virtual page number (address) to a physical page number (address)



"Valid bit" says whether page is in RAM (1) or on HDD (0).

Where are the page tables ?



Page tables are in a reserved data region *in the kernel part of MIPS Memory*. Note that they have both a virtual and a physical address. The page table region is not partitioned into pages. Rather this region has a fixed mapping from virtual to physical memory.

Page Fault and Page Swap

- When a MIPS program tries to access an address whose physical page is on disk (HDD), we say that a "page fault" occurs. The page first must be brought into main memory (RAM) before the program can access that address.

- If there is no page available in main memory, then some page first must be moved out of main memory, and then the desired page can be moved in main memory. This is called a page swap.

- The page table must be updated (regardless of whether a page is swapped out).

Page swaps are done by a kernel program (OS) called the page fault handler (return to this in lecture 21 -- interrupts).

Next week's lectures

- more on page tables (we need a cache for them too !)

- how do caches work ?