Credits:Asanovic @ MIT

Virtual Memory Overview
Handout 17

November 18, 2004
Shoukat Ali

shoukat@umr.edu

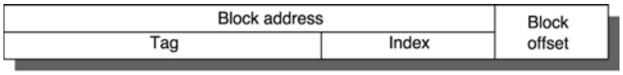
UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

How Have We Evolved?

- In Handout 5 about pipelines, I said:
- when the CPU asks for a word at address A, e.g., lw R1, 20(R2)
  - it gets it back in 5 clock cycles
- **the above is a naïve view of how microprocessors work**
  - because the memory request might not hit in the cache

How Have We Evolved?

- In Handout 7 about caches, I said:
- when the CPU asks for a word at address A, e.g., lw R1, 20(R2)
  - the bits of A are split in three fields:
    - block-offset$_A$
    - index$_A$
    - tag$_A$

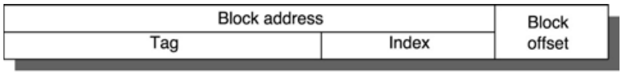


  - cache set at index$_A$ is accessed
  - tags of all frames in set are searched to see if tag$_A$ = stored tag
    - if yes and valid, the block sought by CPU is in cache

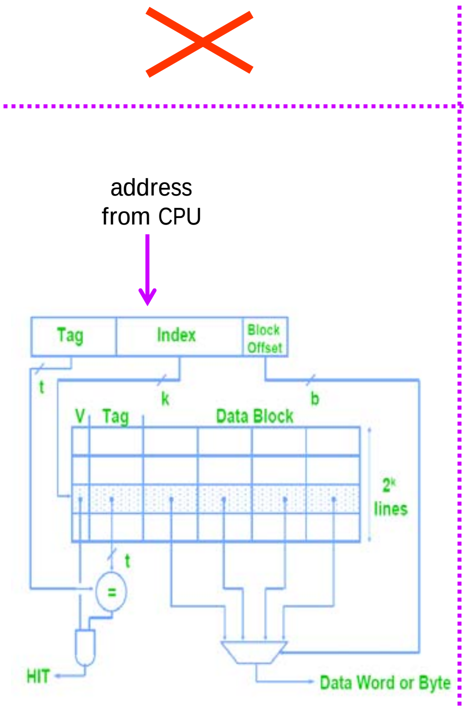
- **the above is a naïve view of how microprocessors work**
  - **because …**

How Have We Evolved?

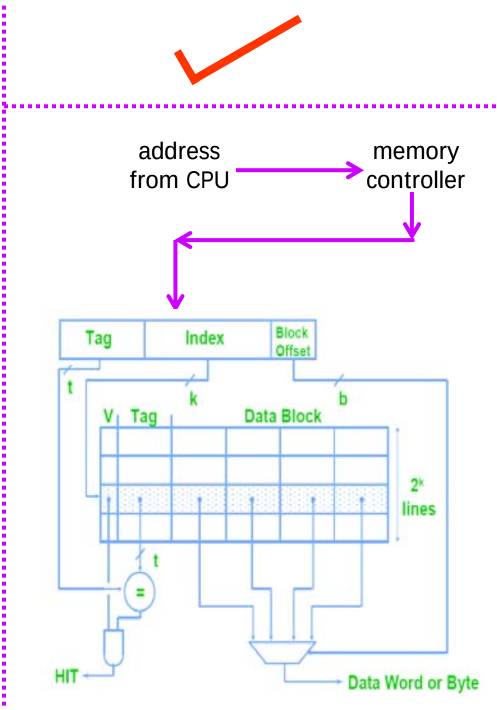
- **New Rules:**
- when the CPU asks for a word at address A, e.g., lw R1, 20(R2)
  - memory system converts A into what is known as a “physical address,” PA
  - the bits of A are split in three fields:
    - block-offset$_{PA}$
    - index$_{PA}$
    - tag$_{PA}$



  - cache set at index$_{PA}$ is accessed
  - tags of all frames in set are searched to see if tag$_{PA}$ = stored tag
    - if yes and valid, the block sought by CPU is in cache



address
from CPU



address
from CPU

memory
controller

Questions

- why does the memory system <u>convert A</u> into a “physical address?”
- what is a physical address?
- what is A?
  - non-physical address? spiritual address? virtual address?

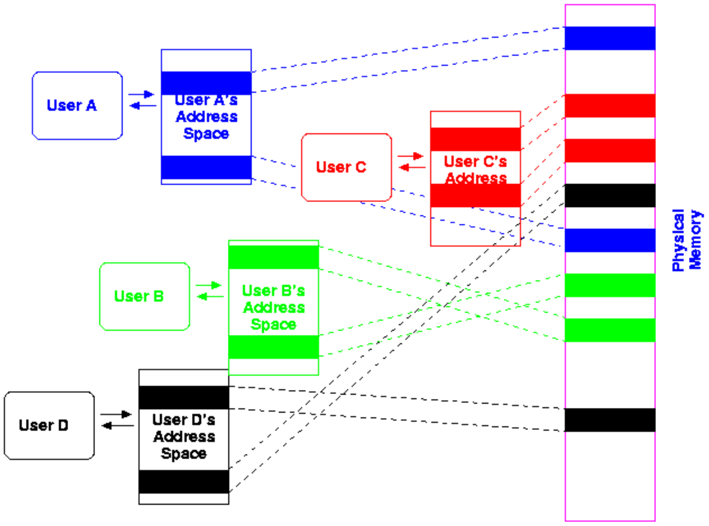
- <u>memory controller converts A in order to implement a scheme called “virtual memory”</u>

What is Virtual Memory?

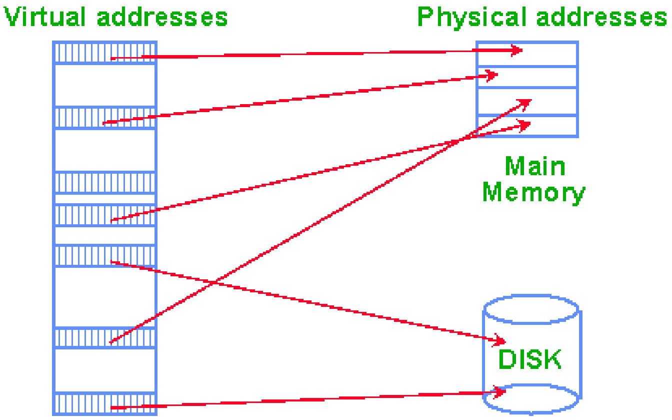
- virtual memory is a scheme for sharing at the same time physical memory among a number of programs
- to be able to execute, a program needs to be in a computer’s memory
  - being in the disk does not suffice
- busy computers alternate quickly among many programs
- to keep many programs in memory at the same time
  - physical memory is divided into pages
  - different programs are allocated different pages
  - currently inactive programs are moved to disk

tasks of a
memory controller

Different Users Share Physical Memory



Parts of a Program May Reside On Disk



Virtual page may not be in physical memory, but on disk.

Access Time Ratios

Access time ratios for various memories:

$$\frac{L2}{L1} = 5 \qquad \frac{\text{Main Memory}}{L2} = 5$$

$$\frac{\text{Disk}}{\text{Main Memory}} = 10^4$$

Huge penalty for miss in Main Memory

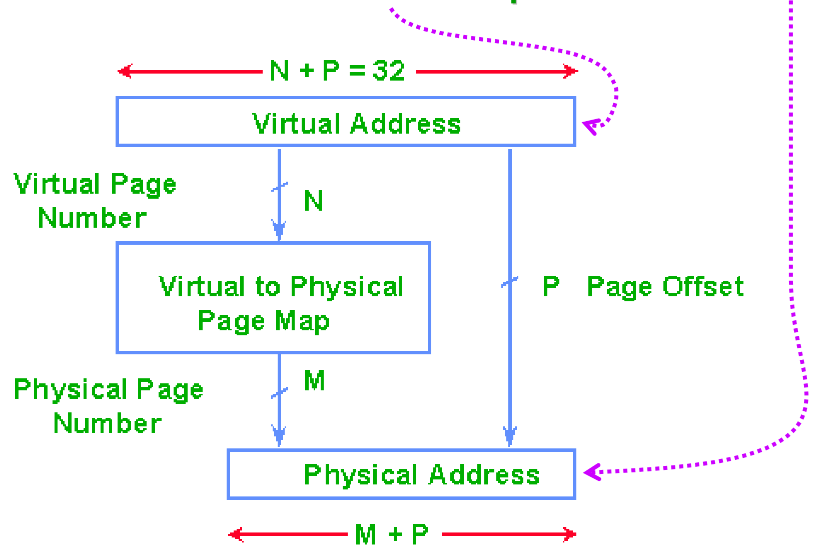
Miss Rate for Memory MUST Be Very Small

Main Memory characteristics

Called a page

- **Block size:** **4-16KB**
- **Associativity:** **Fully associative, implying a virtual page can be mapped to <u>any</u> physical page**
- **Replacement:** **LRU**
- **Write policy:** **Write-Back**

Virtual Memory Implementation OR How Do We Know Which VA Corresponds to Which PA?



N + P = 32

Virtual Address

Virtual Page Number

N

Virtual to Physical Page Map

P Page Offset

Physical Page Number

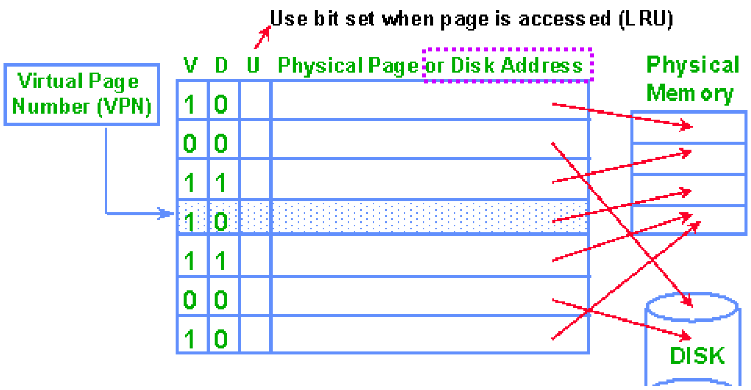
M

Physical Address

M + P

Structure of Virtual to Physical Page Map or Table

Page Table resides in Main Memory


Use bit set when page is accessed (LRU)
Virtual Page Number (VPN)
Physical Memory
DISK

V	D	U	Physical Page or Disk Address
1	0		
0	0		
1	1		
1	0		
1	1		
0	0		
1	0		

Ignore the Use bit.

Does Virtual Memory Hurt My Plan to Use Caches?

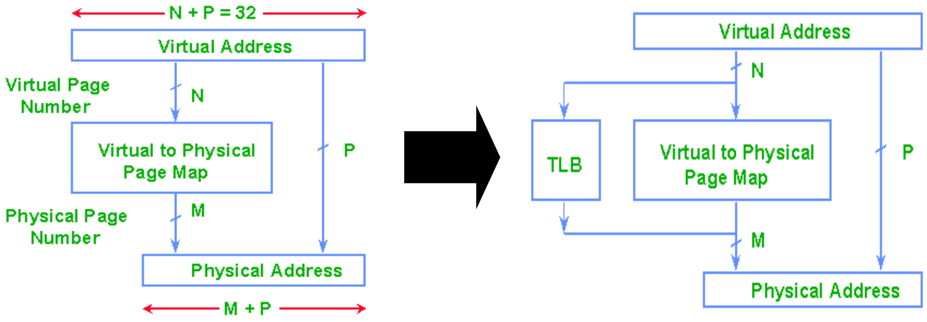
Wish to read a location in Main Memory. Steps are:

- Present a virtual address to Main Memory.
- Determine if corresponding page is in Main Memory by accessing Page Table.
- If page is in Main Memory, access Main Memory again to obtain contents of location in page.
- If page is not in Main Memory, interrupt program and call a Page Fault Handler to bring in the page from disk.
- Repeat read, this time page is in Main Memory

Even assuming 100% HIT rate in Main Memory, the penalty of using virtual memory is 2X.

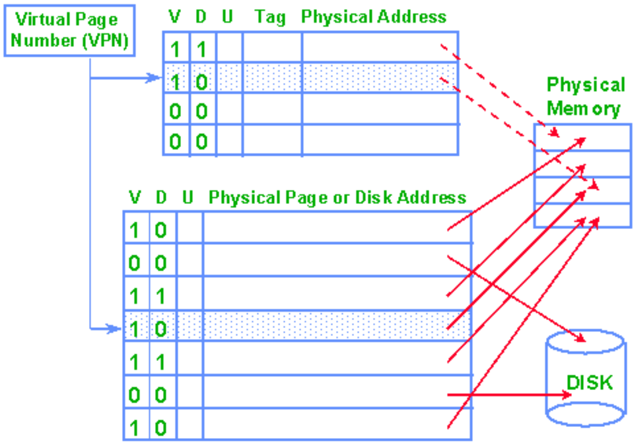
TLB: A Scheme To Allow Both VM and Caches

- TLB = translation lookaside buffer
- it’s a cache of recently used page table entries
- why have it?: if the memory accesses have locality, then address translations must also have locality!



Structure of a TLB

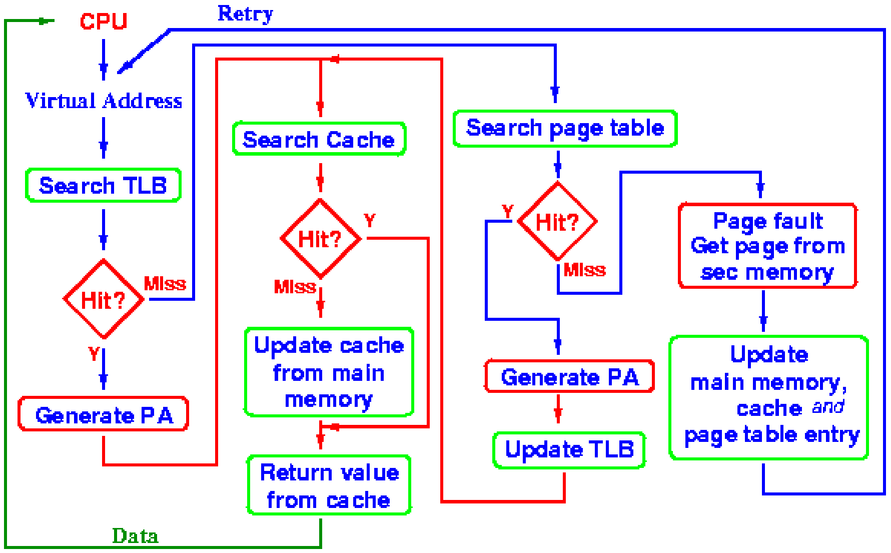
- same as that of a page table
  - much smaller size



V	D	U	Tag	Physical Address
1	1			
1	0			
0	0			
0	0			

V	D	U	Physical Page or Disk Address
1	0		
0	0		
1	1		
1	0		
1	1		
0	0		
1	0		

The Big Picture



CPU

Retry

Virtual Address

Search TLB

Hit?

Miss

Y

Generate PA

Search Cache

Hit?

Y

Miss

Update cache from main memory

Return value from cache

Data

Search page table

Hit?

Y

Miss

Page fault Get page from sec memory

Generate PA

Update TLB

Update main memory, cache *and* page table entry