Virt Cache

COMP

Reviewing the big picture

Review 1/2

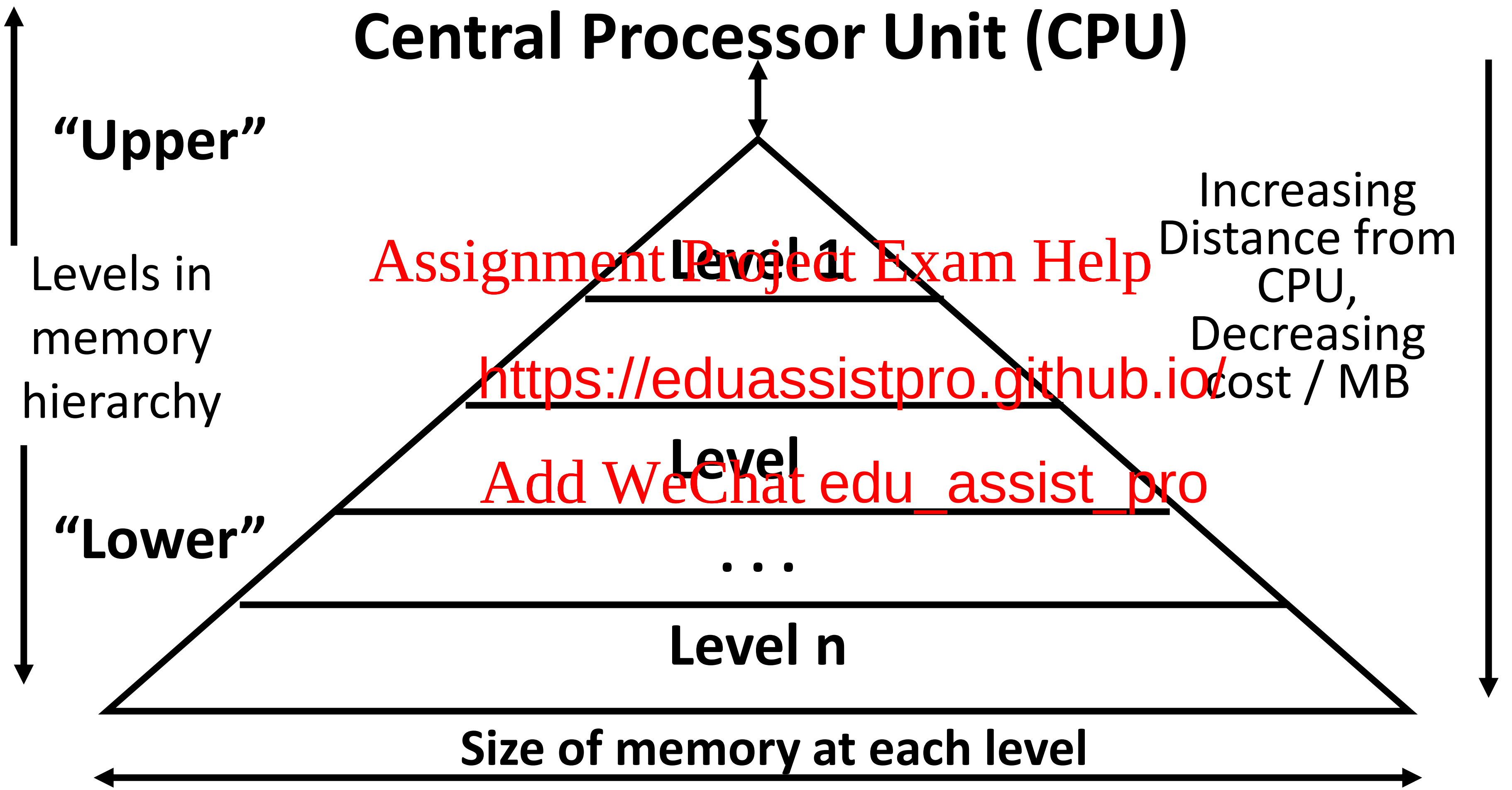
- Apply Principle of Locality Recursively
- Reduce Miss Penalty? add a (L2) cache

- Manage memory to e
  - Included protection
  - Use Page Table of mappings vs. tag/data in cache
- Virtual memory to Physical Memory Translation too slow?
  - Add a cache of Virtual to Physical Address Translations, called a TLB

Review 2/2

- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always-swap
- Spatial Locality mean [illegible]es is all that must be in memory for process to run fairly
- TLB to reduce performance cost of VM
- Need more compact representation to reduce memory size cost of simple 1-level page table (especially 32- ⇒ 64-bit address): 2-level page tables.

Memory Hierarchy Pyramid

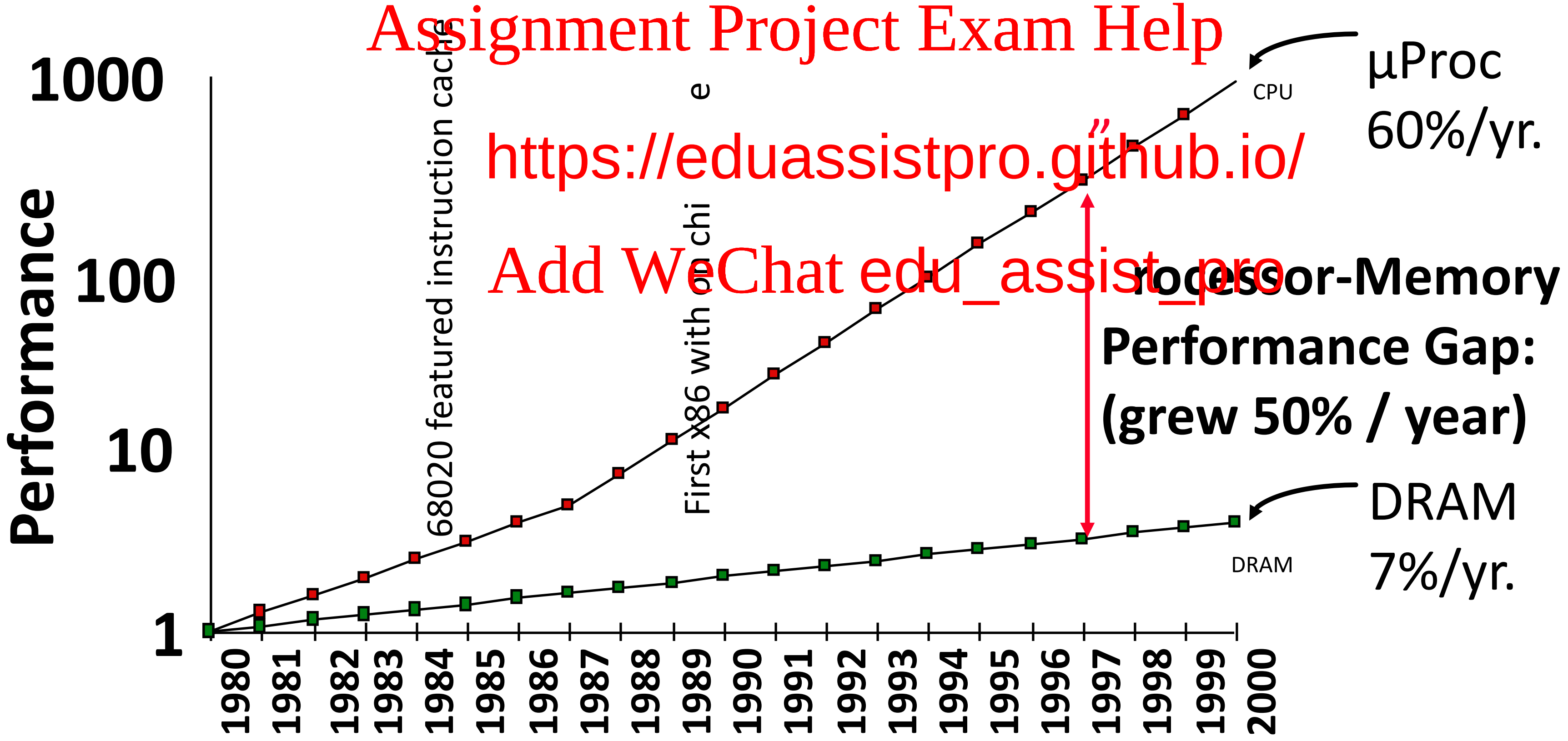

Central Processor Unit (CPU)
"Upper"
Levels in memory hierarchy
"Lower"
Level 1
Level
. . .
Level n
Increasing Distance from CPU, Decreasing cost / MB
Size of memory at each level

Principle of Locality (in time, in space) + Hierarchy of Memories of different speed, cost; exploit to improve cost-performance

Future changes to memory hierarchies?

Why Caches?

- 1989 first Intel CPU with cache on chip
- 1998 Pentium III has two levels of cache on chip

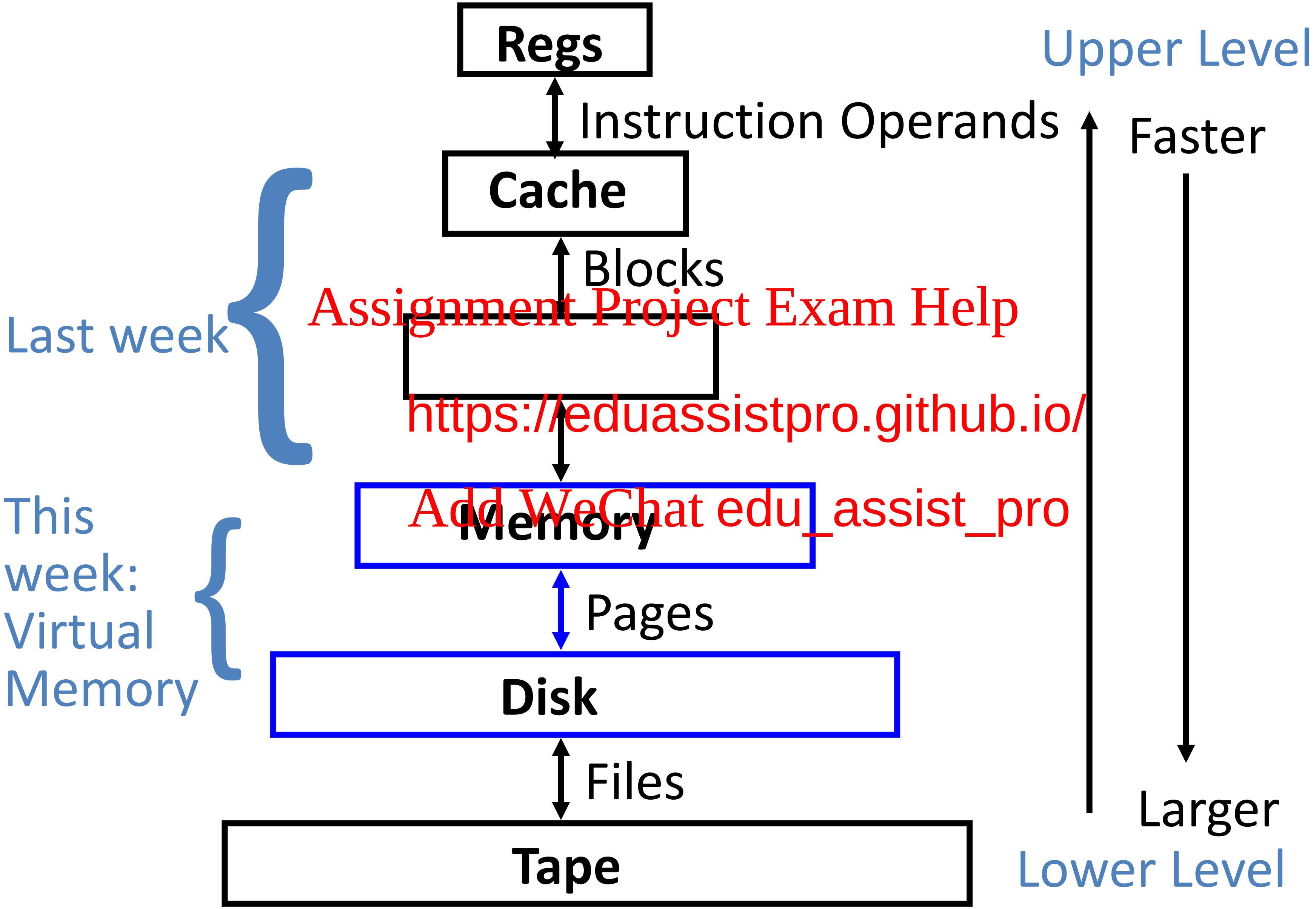


µProc 60%/yr.

Processor-Memory Performance Gap: (grew 50% / year)

DRAM 7%/yr.

Another View of the Memory Hierarchy



Regs

Instruction Operands

Cache

Blocks

Memory

Pages

Disk

Files

Tape

Upper Level

Faster

Larger

Lower Level

Last week

This week: Virtual Memory

Why virtual memory? (1/2)

- **Protection**
  - Regions of the address space can be read only, execute only, ...

- **Flexibility**
  - Portions of a program [illegible]e, without relocation

- **Expandability**
  - Can leave room in virtual address space for objects to grow
- **Storage management**
  - Allocation/deallocation of variable sized blocks is costly and leads to (external) fragmentation; paging solves this

Why virtual memory? (2/2)

- **Generality**
  - Ability to run programs larger than size of physical memory
- **Storage efficiency**
  - Retain only most impor am in memory
- **Concurrent I/O**
  - Execute other processes while loading/ e

Virtual Memory Overview (1/3)

- User program view of memory:
  - Contiguous
  - Start from some set address
  - Infinitely large
  - Is the only running pr
- Reality:
  - Non-contiguous
  - Start wherever available memory is
  - Finite size
  - Many programs running at a time

Virtual Memory Overview (2/3)

- Virtual memory provides:
  - Illusion of contiguous memory
  - All programs startin
  - Illusion of effectively (2^{32} or 2^{64} bytes)
  - Protection

Virtual Memory Overview (3/3)

- Implementation:
  - Divide memory into "chunks" (pages)
  - Operating system co [illegible] maps virtual addresses into physical addres [illegible]
  - TLB is a cache for the page table
  - Can think of memory as a cache for disk

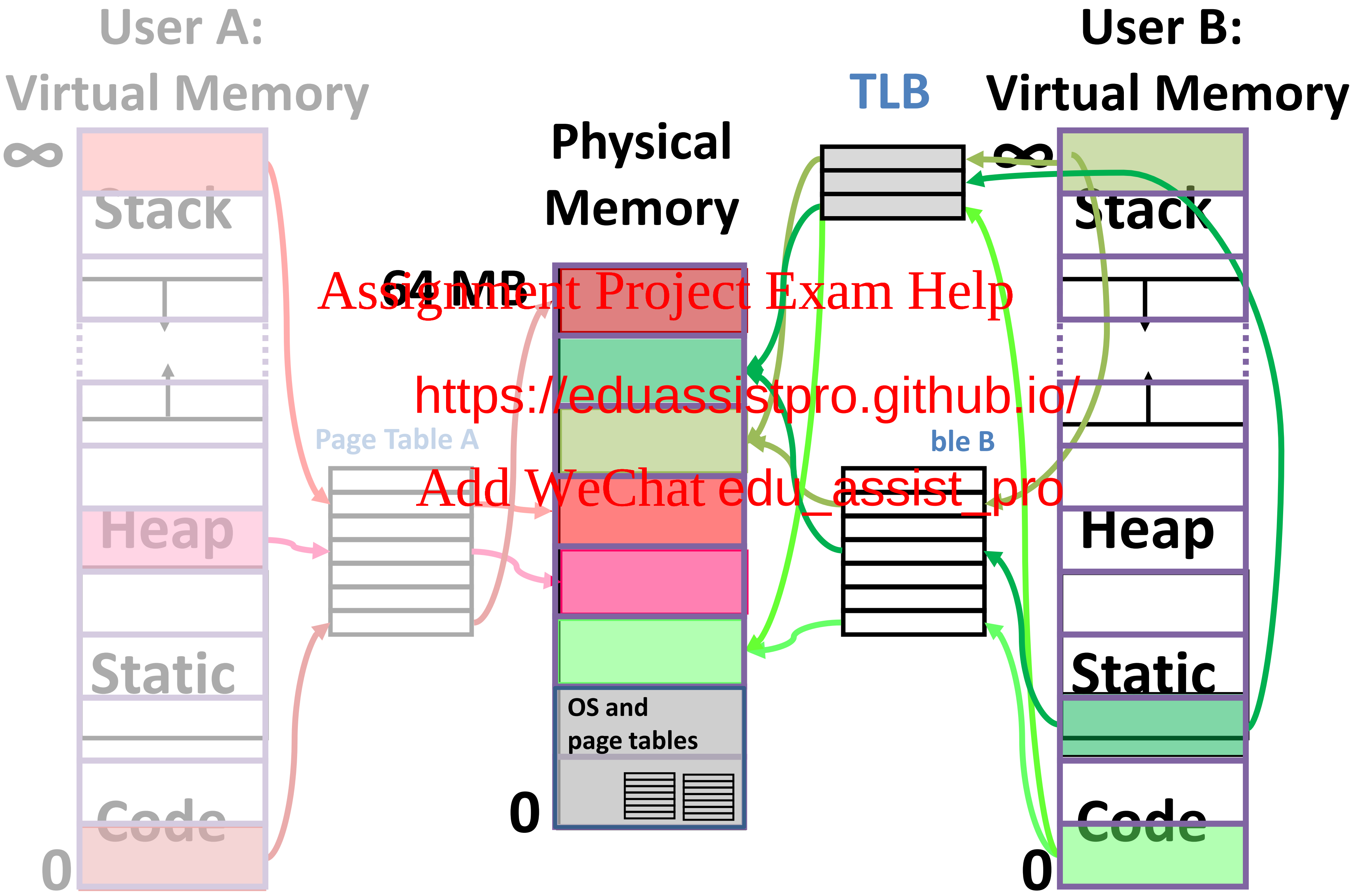
Why Translation Lookaside Buffer (TLB)?

- Paging is most popular implementation of virtual memory
- In a paged implementation, every virtual memory access must corresponding entry of the stored in physical memory) to prov ction
- Cache of Page Table Entries (TLB) makes address translation possible without memory access (to read page table)
- TLB exploits temporal and spatial locality, making the common case memory accesses fast

Load data example

- Suppose we are fetching (loading) some data:
  - Check TLB (input: VPN, output: PPN)
    - hit: fetch translation
    - miss: check page tab
      - **Page table hit: f**
      - **Page table miss: page fault, fetc disk to memory, return translation to TLB**
  - Check cache (input: PA, output: data)
    - hit: return value
    - miss: fetch value from memory

Paging/Virtual Memory Review



Three Advantages of Virtual Memory

1) Translation

- Program can be given **consistent view of memory**, even though physical memory is scrambled

- Makes **mult** ble

- Only the most important gram, i.e., the "**Working Set**", must be i memory

- Contiguous structures (like stacks) **use only as much physical memory as necessary** yet still grow later

Three Advantages of Virtual Memory

2) **Protection**:

- – Different processes protected from each other

- – Different pa al behaviour
  - (Read Onl

ms, etc).

- – Kernel data protected fro

ograms

- – Very important for protection from malicious programs (viruses)
- – Special Mode in processor ("**Kernel mode**") allows processor to change page table/TLB

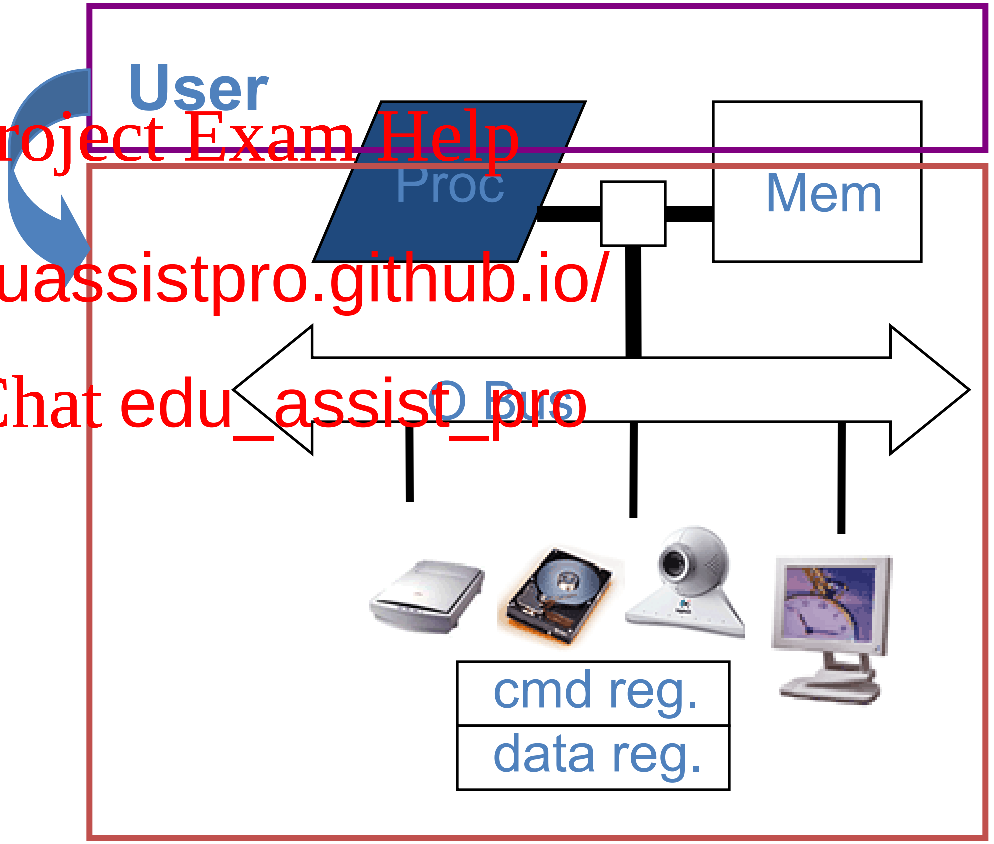
Three Advantages of Virtual Memory

3) **Sharing**:

– Can map same physical page to multiple users (“**Shared memory”**)

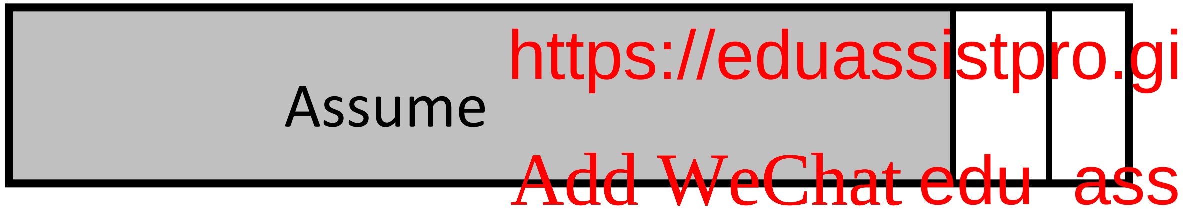
Crossing the System Boundary

- System loads user program into memory and “gives” it use of the processo
- Switch back
  - SYSCALL
    - request service
    - I/O
  - TRAP (overflow)
  - Interrupt



Instruction Set Support for VM/OS

- How to prevent user program from changing page tables and go anywhere?
  - Bit in Status Register determines whether in user mode or OS (kernel) mode:



Kernel/User bit (KU) (0 ⇒ kernel, 1 ⇒ user)

  - On exception/interrupt disable interrupts (IE=0) and go into kernel mode (KU=0)
- Only change the page table when in kernel mode (Operating System)

Syscall

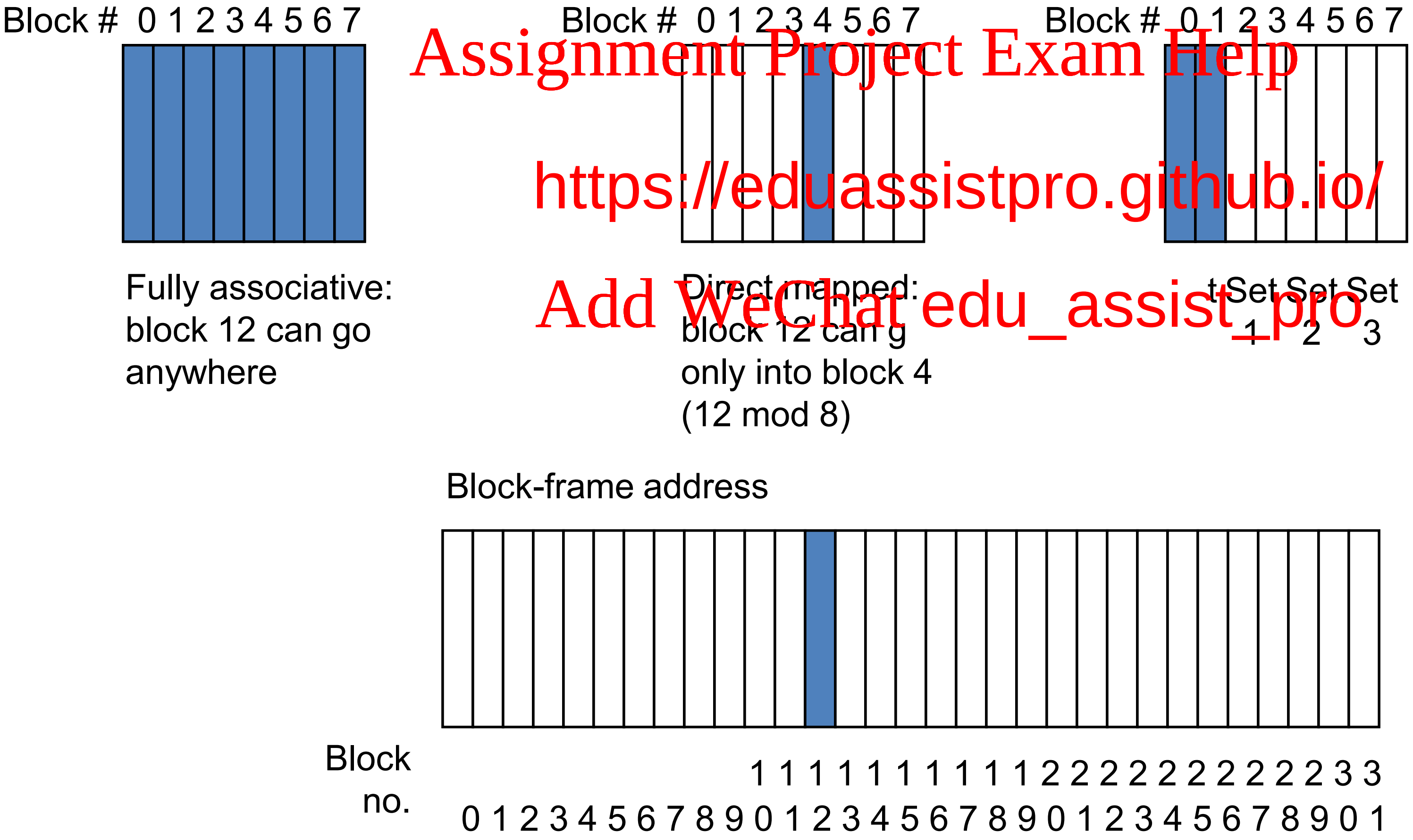
- How does user invoke the OS?
  - `syscall` instruction: invoke the kernel (Go to 0x80000080, change to kernel mode)
  - By software convent ervice requested: OS performs request

4 Questions for Memory Hierarchy

- Q1: Where can a block be placed in the upper level? *(Block placement)*
- Q2: How is a block found if it is in the upper level? *(Block identific*
- Q3: Which block should be r a miss? *(Block replacement)*
- Q4: What happens on a write? *(Write strategy)*

Q1: Where block placed in upper level?

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Mod Number Sets

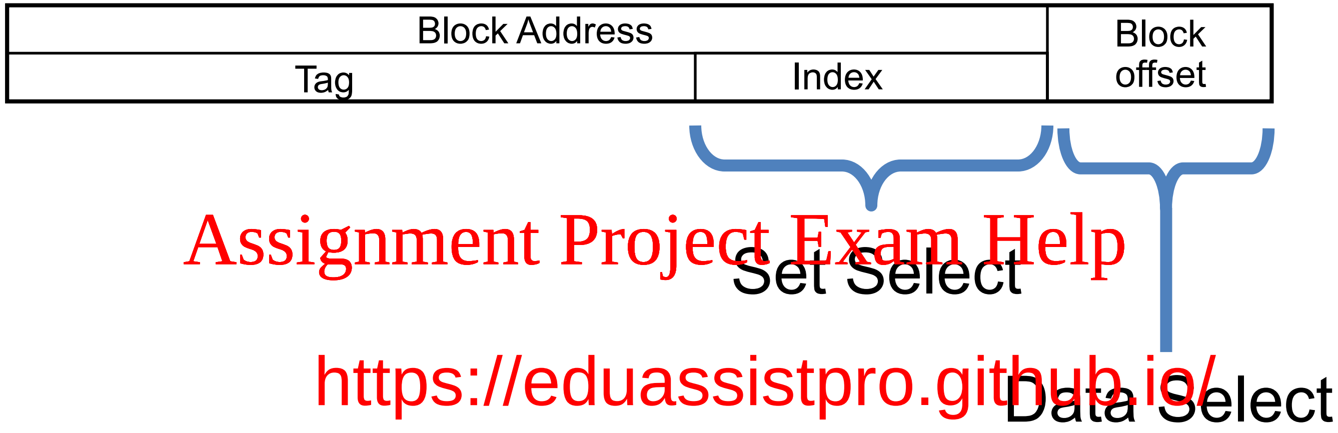


Fully associative: block 12 can go anywhere

Direct mapped: block 12 can g only into block 4 (12 mod 8)

Set associative: block 12 can go anywhere in set 0 (12 mod 4)

Q2: How is a block found in upper level?



- Direct indexing (using in block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

Q3: Which block replaced on a miss?

- Easy for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least R

Miss Rates Example

Associativity:	2-way				8-way	
Size	LRU	Ran	LRU	Ran	LRU	Ran
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

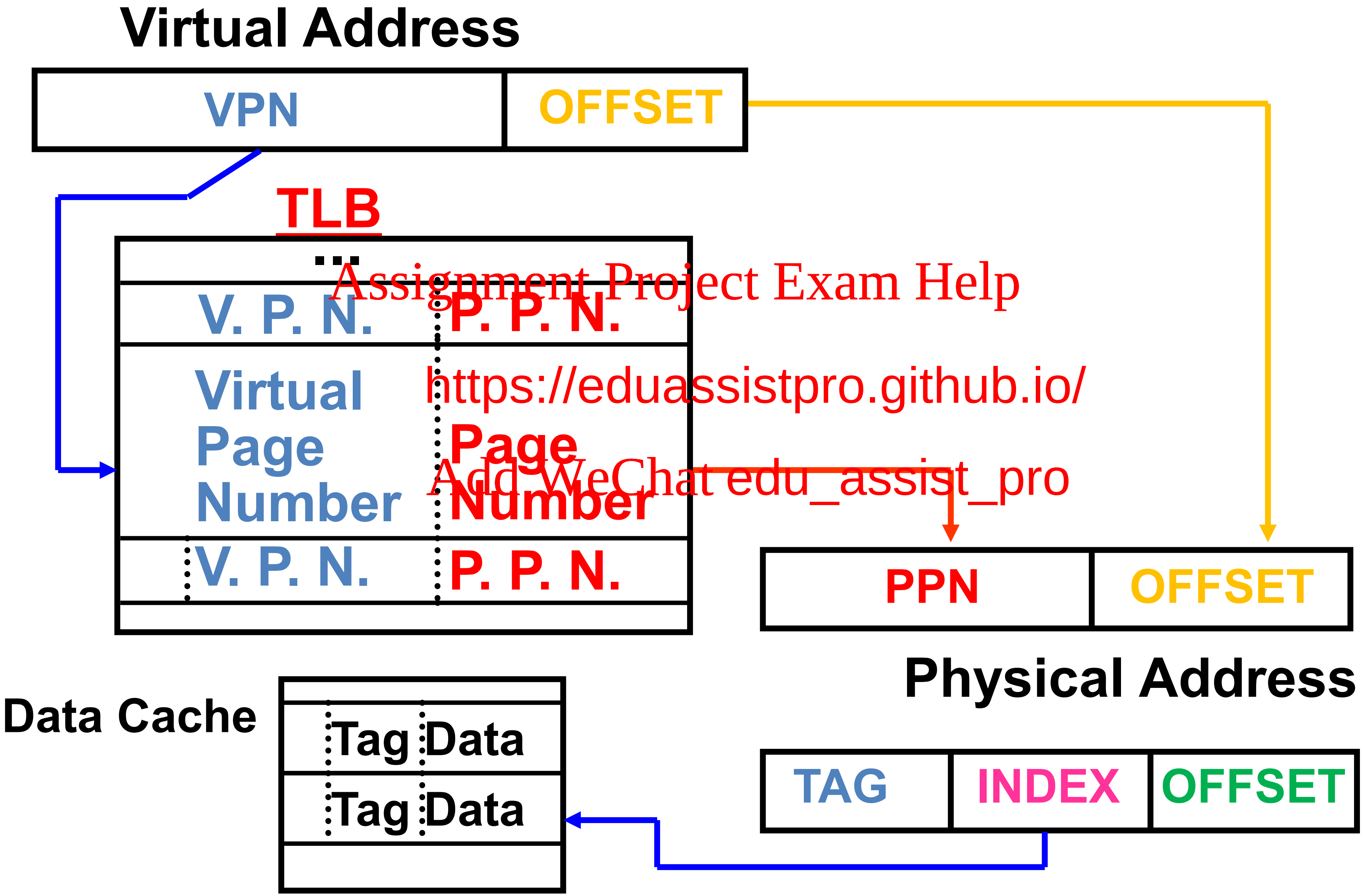
Q4: What to do on a write hit?

- Write-through
  - update the word in cache block and corresponding word in memory
- Write-back
  - update word in cache bl
  - allow memory word to

  => add ‘dirty’ bit to each line indicating th e updated when block is replaced

  => OS flushes cache before I/O !!!
- Performance trade-offs?
  - WT: read misses cannot result in writes
  - WB: no writes of repeated writes

Address Translation & 3 Concept tests



Cache and Virtual Memory

- Virtual memory and cache work together
- Hierarchy must be preserved
  - When a page is migr ill flushing the contents of the page from the c
  - Also modifies page table and TLB mpts to access data on migrated page will produce a fault.

Question

- A memory reference can encounter three different types of misses:
  - TLB miss, p
- Consider all ese events with one or more occur ssibilities).
- State if each event can actually occur and under what circumstances

Answer

TLB	PAGE TABLE	CACHE	POSSIBLE? HOW?
Hit	Hit	Miss	Possible, though page table not checked if TLB hits
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is
Miss	Hit	Miss	ge table; after retry, data misses in cache
Miss	Miss	Miss	TLB misses and is fol e fault; after retry, data ***must*** miss cache
Hit	Miss	Miss	impossible: cannot have a translation in TLB if page is not present in memory
Hit	Miss	Hit	impossible: cannot have a translation in TLB if page is not present in memory
Miss	Miss	Hit	impossible: data not allowed in cache if the page is not in memory

Understanding Program Performance

- Virtual memory allows a small memory to look like a large one
- A process that routinely accesses more virtual memory than it has physical memory will run slowly… It will continuously be swapping pages between memory and disk, called ***thrashing***
- Easiest solution: buy more
- Better solution: examine algorithms and d s to see if you can change the locality, and reduce the number of pages y ***working set***
- TLB misses a more common problem, and can be alleviated with larger page sizes (most computer architectures support variable page sizes, but not necessarily the OS).

Cache/VM/TLB Summary: #1/3

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
    - Temporal L
    - Spatial Loca
- Caches, TLBs, Virtual Memor stood by examining how they deal with 4 questions:
  1) Where can block be placed?
  2) How is block found?
  3) What block is replaced on miss?
  4) How are writes handled?

Cache/VM/TLB Summary: #2/3

- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always-swap or base/bound

- Three Problems:

  1) Not enough memory: Spatial Localit all Working Set of pages OK

  2) TLB to reduce performance cost of VM

  3) Need more compact representation to reduce memory size cost of simple 1-level page table, especially for 64-bit address space *(beyond scope of this course)*

Cache/VM/TLB Summary: #3/3

- Virtual memory was controversial at the time: can software automatically manage 64KB across many programs?
  - 1000X DRAM growth removed controversy

- Today VM allow are single memory without having t disk;

  - **VM protection today is more i an memory hierarchy**

- Today CPU time is a function of #operations and cache misses, rather than just a function of #operations.
  - What does this mean to Compilers, Data structures, Algorithms?

Review and More Information

- Textbook 5.7 – Virtual Memory
- See also 5.8