بسم الله الرحمن الرحیم

Advanced Computer Architecture

Teacher

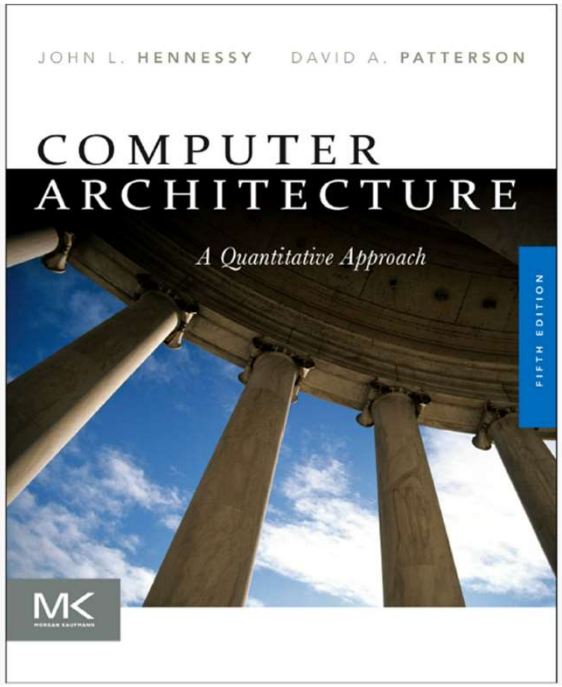
Fatemeh Daraee

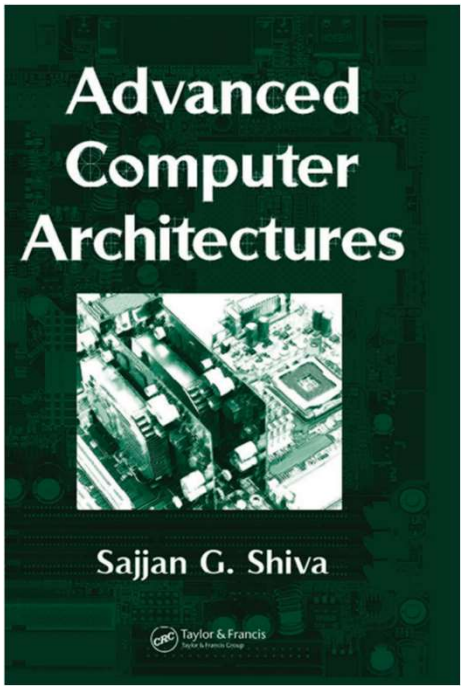
f_daraei@semnan.ac.ir

https://fdaraei.profile.semnan.ac.ir

Semnan University

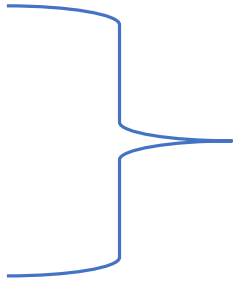
- **Course Textbook**
- Hennessy J. L & Patterson, D. A. (2011). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann 5th edition.
- Shiva, S. G. (2006). *Advanced Computer Architecture*. CRC Press.





Course Outline

- High-Speed Memory Systems
- Pipeline Architecture
- Vector Computers
- Multiprocessor and Multi-Computer Systems
- Parallel Programming
- Interconnection Networks
- Low-Cost Acceleration
- Distributed Computing, Grid Computing



Seminar

Grading

• **Midterm**	**6**
• **Final**	**10**
• **Seminar**	**5**
	21

- Your seminar is a research study on Interconnection Networks, Low-Cost Acceleration, distributed computing, or GRID computing.

Reminder from before

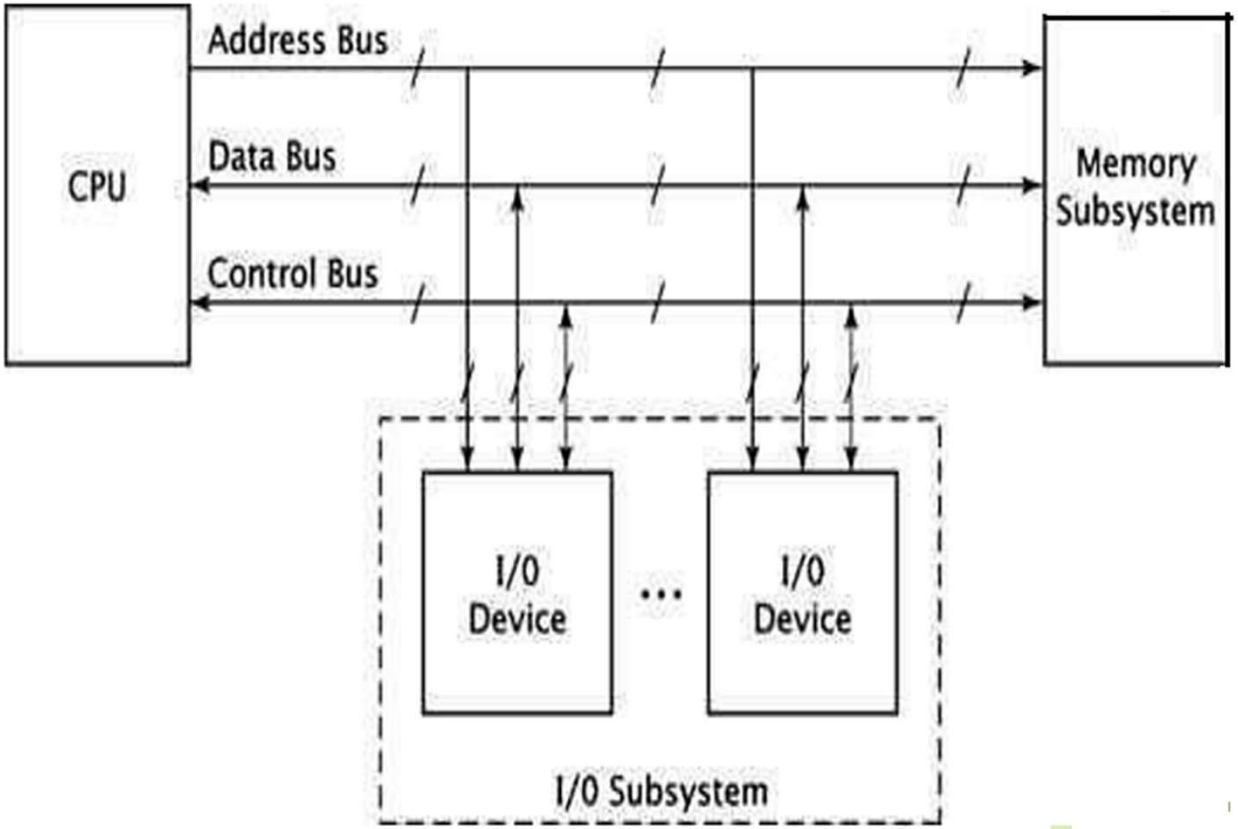
Semnan University

Basic Computer Organization

The basic computer organization has three main components:

- **CPU**
- **Memory subsystem**
- **I/O subsystem**

Generic Computer Organization



Semnan University

System bus

The system bus has three buses:

- **Address bus**
- **Data bus**
- **Control bus**

Address bus

The uppermost bus is the address bus. When the CPU reads data or instructions from or writes data to memory, it must specify the address of the memory location it wishes to access.

Data bus

Data is transferred via the data bus. When CPU fetches data from memory it first outputs the memory address on to its address bus. Then memory outputs the data onto the data bus. Memory then reads and stores the data at the proper locations.

Control bus

Control bus carries the control signal. Control signal is the collection of individual control signals. These signals indicate whether data is to be read into or written out of the CPU.

Instruction cycle

The instruction cycle is the procedure a microprocessor goes through to process an instruction.

It has three phases:

- **Fetch**
- **Decode**
- **Execute**

Instruction cycle

- First the processor **fetches** or reads the instruction from memory.
- Then it **decodes** the instruction determining which instruction it has fetched.

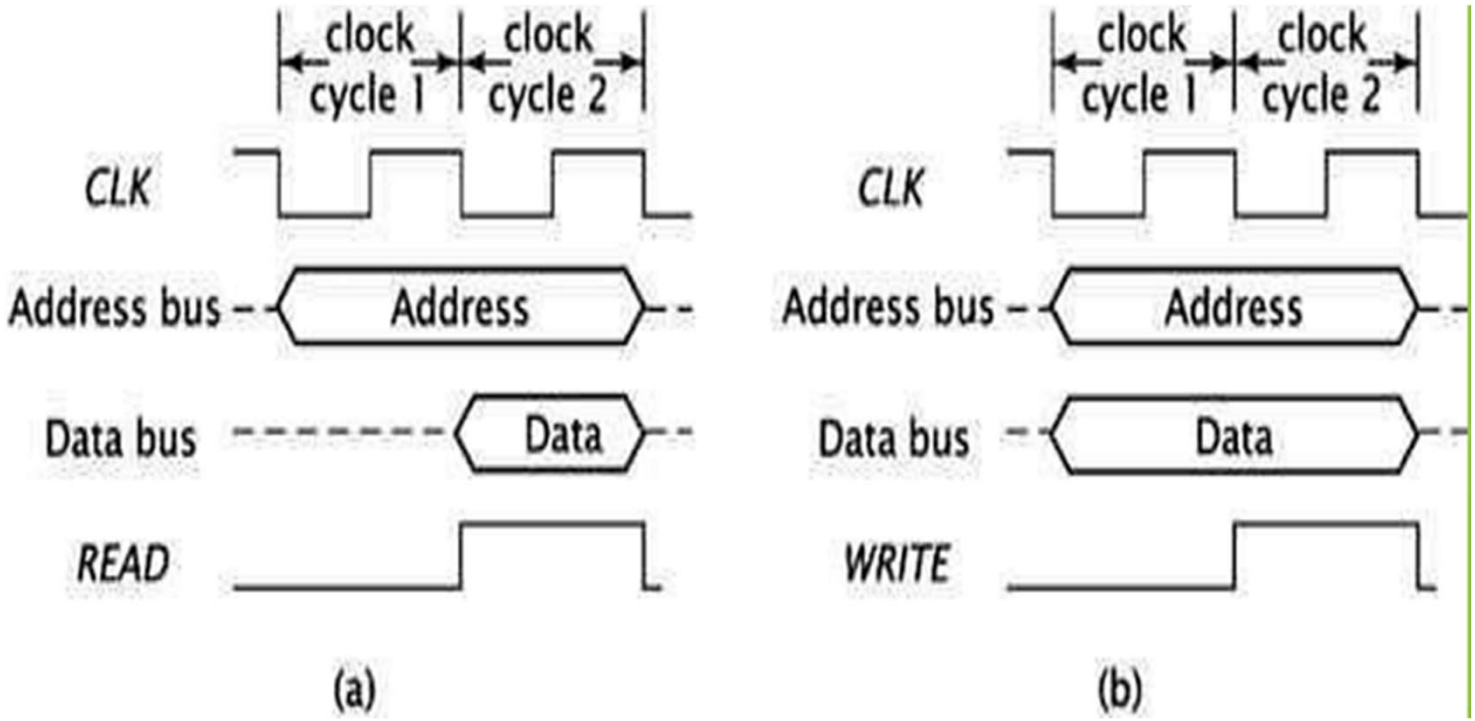
- Finally, it performs the operations necessary to **execute** the instruction.
- It performs some operation internally, and supplies the address, data & control signals needed by memory & I/O devices to execute the instruction.

Control signals

Semnan University

- The **READ** signal is a signal on the control bus which the microprocessor asserts when it is ready to read data from memory or I/O device.
- When **READ** signal is asserted, the memory subsystem places the instruction code to be fetched onto the computer system's data bus. The microprocessor then inputs the data from the bus and stores it in its internal register.
- **READ** signal causes the memory to read the data;
  the **WRITE** operation causes the memory to store the data.

Timing Diagram



Memory read operation

Semnan University

- In fig (a), the microprocessor places the address on to the bus at the beginning of a clock cycle, a 0/1 sequence of clock. One clock cycle later, to allow for memory to decode the address and access its data, the microprocessor asserts the **READ** control signal.
- This causes the memory to place its data onto the system data bus. During this clock cycle, the microprocessor reads the data off the system bus and stores it in one of the registers.
- At the end of the clock cycle, it removes the address from the address bus and desserts the **READ** signal. Memory then removes the data from the data bus, completing the memory read operation.

Memory write operation

- In fig (b), the processor places the address and data onto the system bus during the first clock pulse.
- The microprocessor then asserts the **WRITE** control signal at the end of the second clock cycle.
- At the end of the second clock cycle, the processor completes the memory write operation by removing the address and data from the system bus and deserting the **WRITE** signal.

CPU Organization

Central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control, and input/output (I/O) operations specified by the instructions.

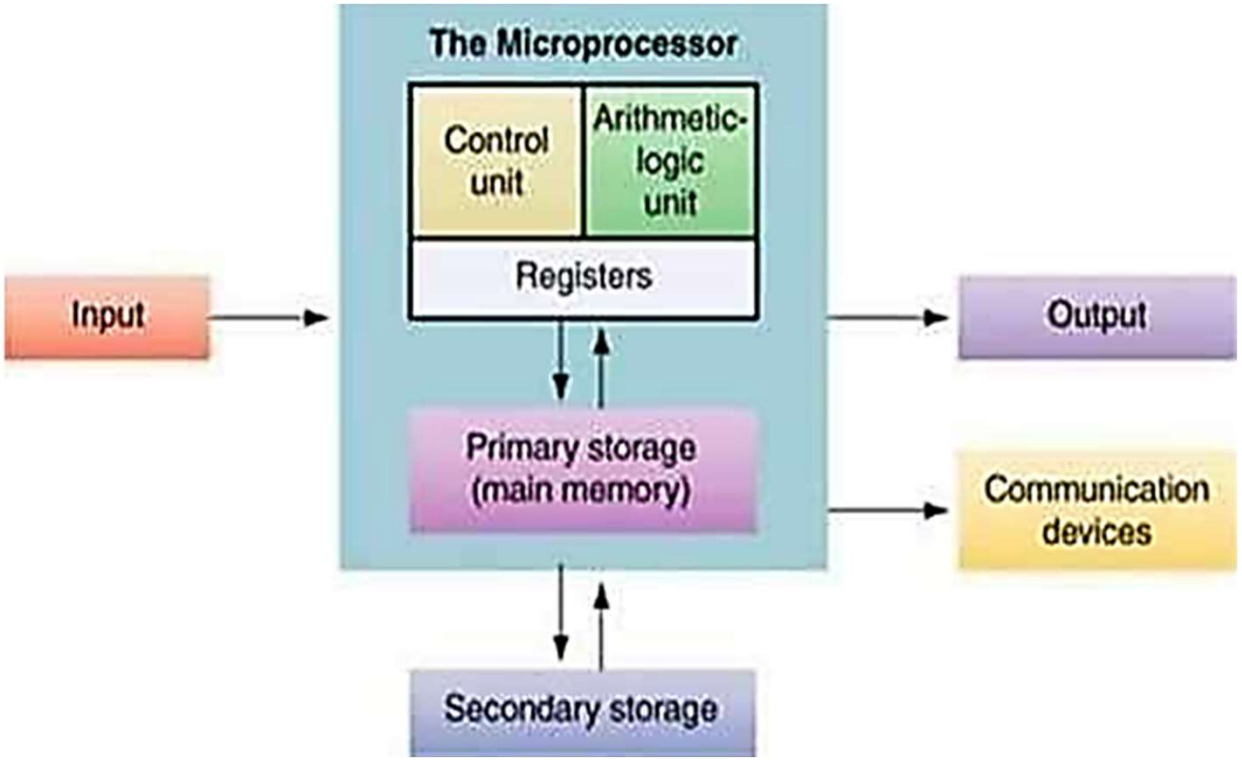
CPU Organization

Semnan University

- In the computer, all the major components are connected with the help of the **system bus**.
- **Data bus** is used to shuffle data between the various components in a computer system.
- When the software wants to access some particular memory location or I/O device, it places the corresponding address on the **address bus**.
- The **control bus** is an eclectic collection of signals that control how the processor communicates with the rest of the system.
  The **read** and **write** control lines control the direction of data on the data bus.

Semnan University

CPU Organization



CPU Organization

- The register section, as its name implies, includes a set of registers and a bus or other communication mechanism.
- The register in a processor's instruction set architecture are found in the section of the CPU.
- The system address and data buses interact with this section of CPU. The register section also contains other registers that are not directly accessible by the programmer.

CPU Organization

- The fetch portion of the instruction cycle, the processor first outputs the address of the instruction onto the address bus. The processor has a register called the **program counter**.
- At the end of the instruction fetch, the CPU reads the instruction code from the system data bus.
- It stores this value in an internal register, usually called the **instruction register**.

CPU Organization

Semnan University

- The **arithmetic / logic unit (or ALU)** performs most arithmetic and logic operations such as adding and ANDing values.

- CPU controls the computer, the **control unit** controls the CPU. The control unit receives some data values from the register unit, which it uses to generate the control signals.

- The **control unit** also generates the signals for the system control bus such as **READ**, **WRITE**, and **I/O** signals.

Memory Organization

Memory Unit

- An essential component in any general-purpose computer since it is needed to store programs and data.
- Memory unit that communicates directly with the CPU = main memory.
- Devices that provide backup storage = auxiliary memory.
- Auxiliary memory devices are used to store system programs, large data files, and other backup information. Only programs and data currently needed by the processor reside in main memory. All other information is stored in main memory and transferred to main memory when needed.

Semnan University

Cache Memory

- Memory that lies in between main memory and CPU.
- Holds those parts of the program and data that are most heavily used.
- Increases the overall processing speed of the computer by providing frequently required data to the CPU at a faster speed.

Semnan University

Main Memory

- Memory unit that communicates directly with CPU
- Programs and data currently needed by the processor reside here
- Also known as primary memory
- RAM and ROM

Semnan University

Auxiliary Memory

- Made of devices that provide backup storage
- Magnetic tapes, Magnetic disks
- At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files, whereas at the top level, magnetic disks used as backup storage

Memory Types

Sequential Access Memory

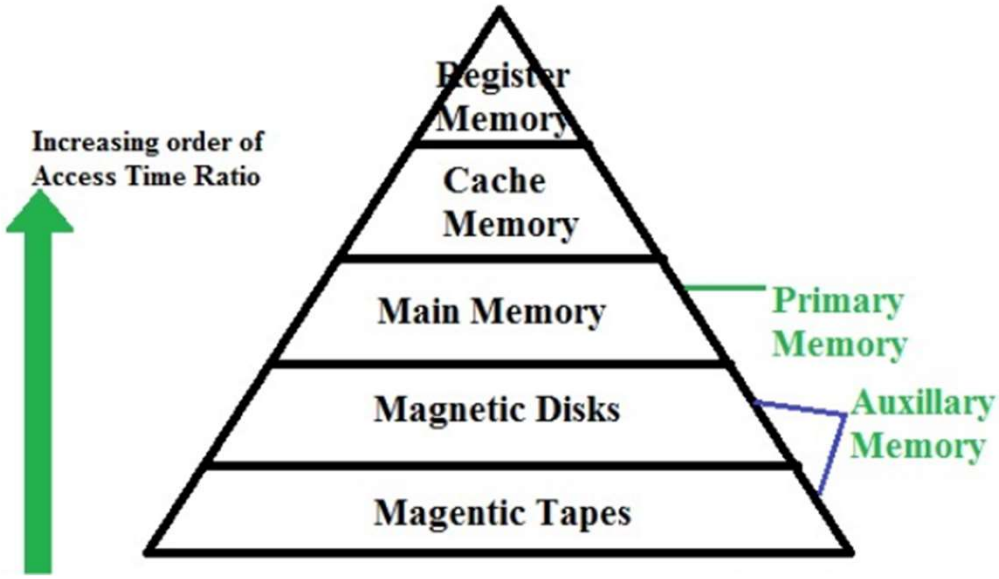
- A class of data storage device that read their data in sequence
- Are usually a form of magnetic memory
- Typically used for secondary storage in general-purpose computers due to their higher density, resistance to wear, and non-volatility
- Eg: hard disk, CD-ROMs, magnetic tapes, etc.

Random Access Memory

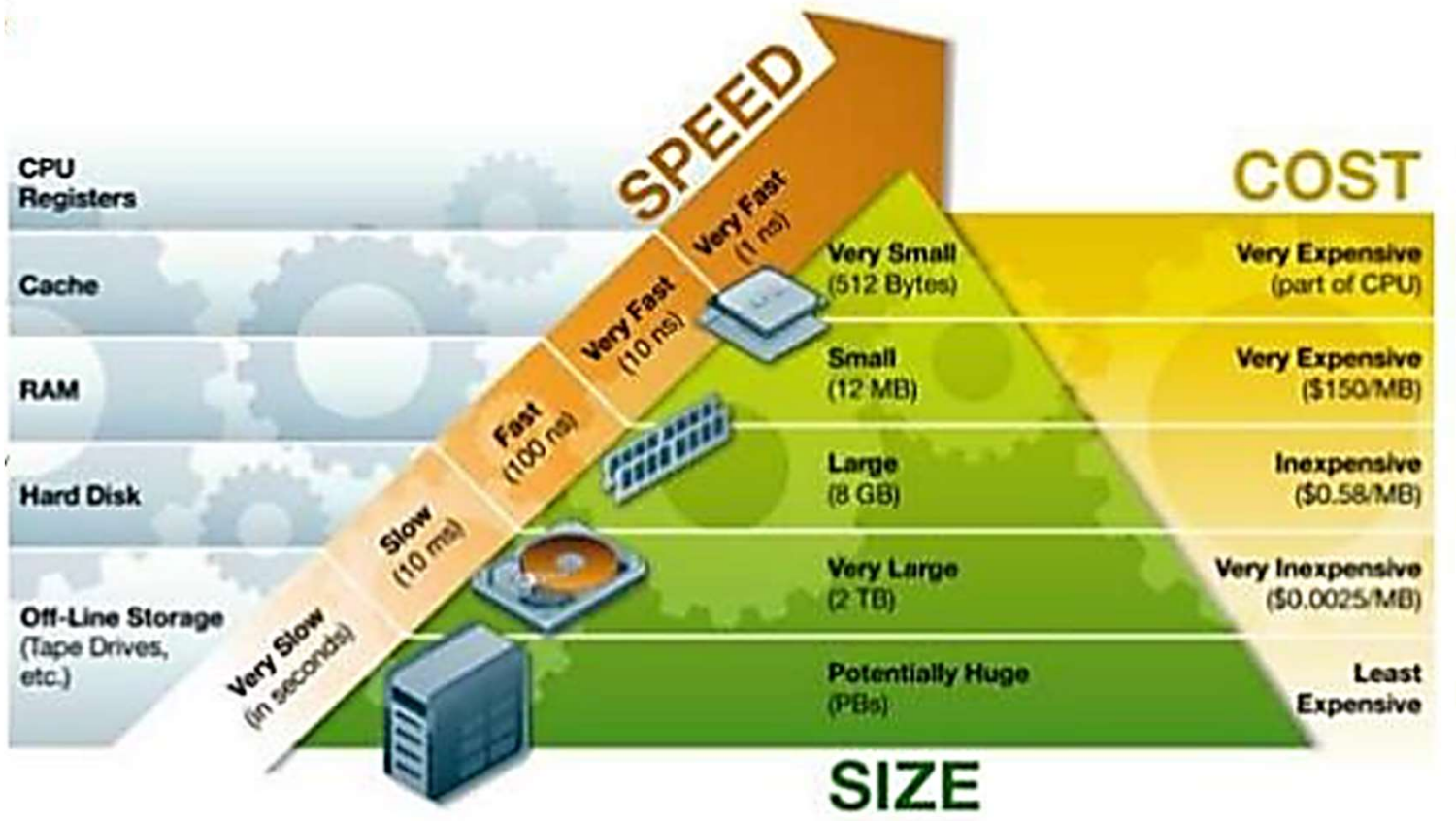
- Is a form of computer data storage
- Allows stored data to be accessed in any order
- Associated with volatile types of memory
- Type: SRAM and DRAM

Memory Hierarchy

- To obtain the highest possible access speed while minimizing the total cost of the memory system
- Consists of all storage devices in a computer system (auxiliary, cache, main, high-speed registers, and processing logic)



Semnan University



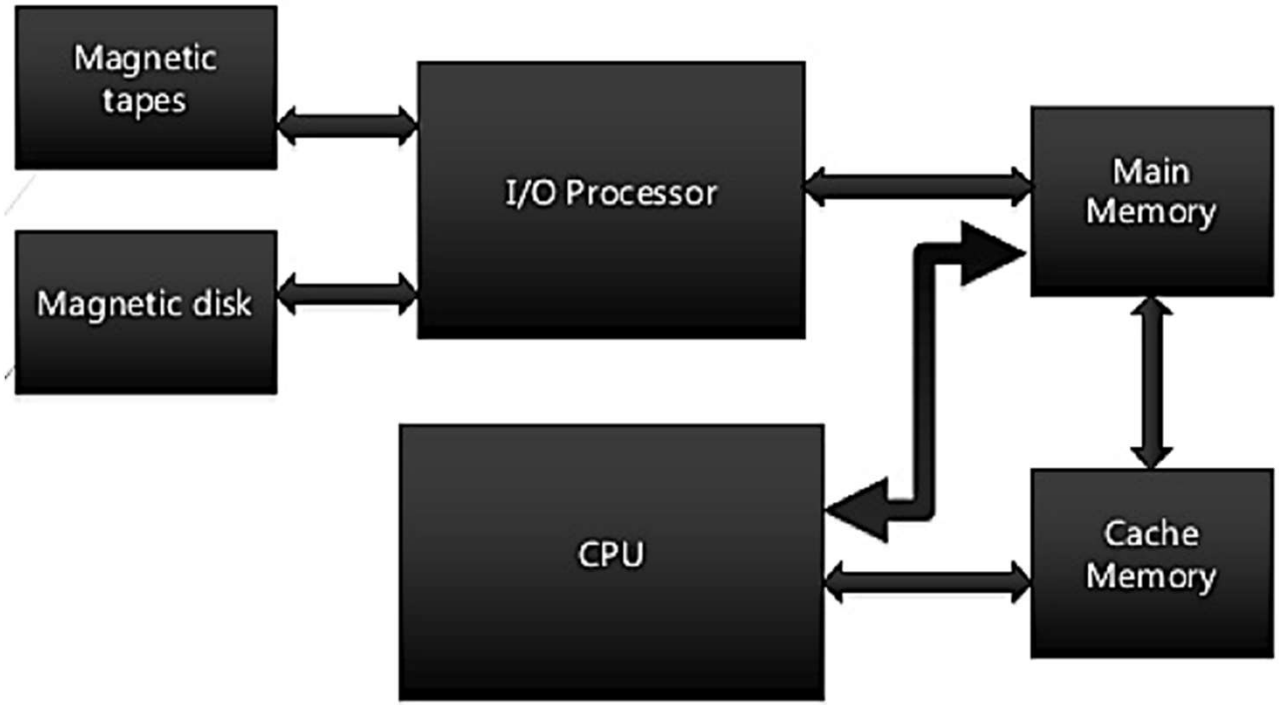


Fig: Memory Hierarchy in a Computer System

Main Memory

- Basic memory of the computer
- Temporary memory except ROM
- Faster for read/write operations
- Expensive internal memory, so not portable.

RAM

- Volatile memory.
- Stores information required during processing.
- Two types of random: Static Ram (SRAM) and Dynamic Ram (DRAM).

TYPES OF RAM

Semnan University

SRAM

- Does not lose its content until the computer is turned off
- Information is stored in the form of voltage
- Faster

DRAM

- Loses its content after a few seconds
- Information is stored in the form of charge
- Slower

Semnan University

ROM

- Permanent memory
- Stores information required for computer operations
- Types of ROM (PROM, EPROM, EEPROM)

Semnan University

Auxiliary Memory (SECONDARY MEMORY)

- The most common auxiliary device used in the computer system is magnetic disk and magnetic tape.
- Store large amounts of data permanently.
- Portable.

Types of Auxiliary Memory

- Magnetic disk
- Magnetic tape

Semnan University

Magnetic Disk

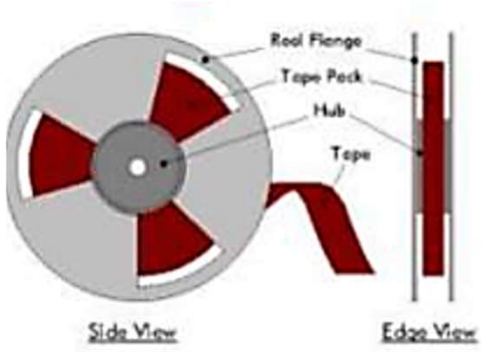
- Circular plate made of metal or plastic coated with magnetized material. High speed of rotation.

- Bits are stored in concentric circles called tracks.

- Division of tracks are called sectors.

Magnetic Tape

- Sequential access memory used for storing, backup, audio, video data, etc.
- Highly reliable memory.
- Slower for read/write operations.



Memory Technology

Static RAM (SRAM)

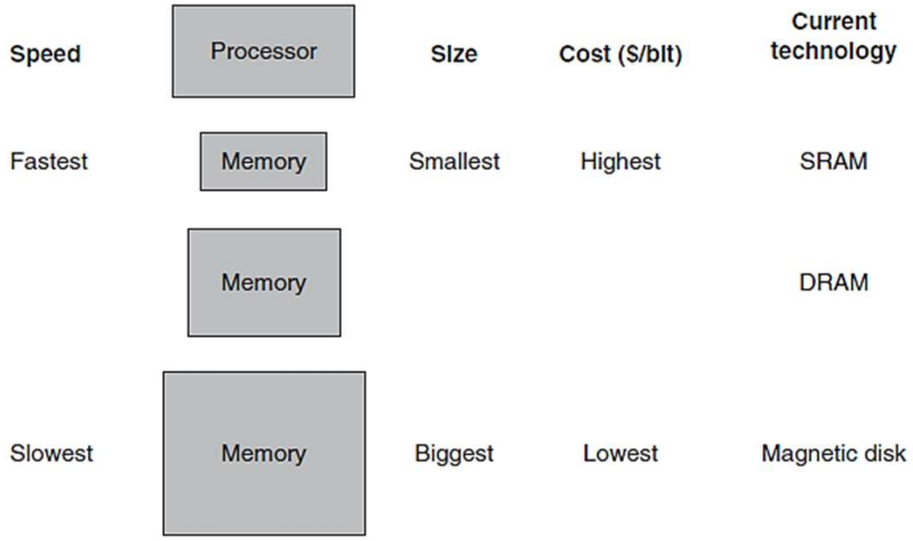
- Access Time: 0.5 – 2.5 ns
- Cost: $2000 – $5000 per GB

Dynamic RAM (DRAM)

- Access Time: 50 – 70 ns
- Cost: $20 – $75 per GB

Magnetic Disk

- Access Time: 5 – 20 ms
- Cost: $0.20 – $2 per GB



Ideal Memory

Access Time of SRAM
Capacity and cost per GB of disk

Principle of Locality

Programs access a small proportion of their address space at any time.

Temporal locality

- Items accessed recently are likely to be accessed again soon.
- e.g., instructions in a loop, induction variables.

Spatial locality

- Items near those accessed recently are likely to be accessed soon.
- e.g., sequential instruction access, array data.

Taking Advantage of Locality

Semnan University

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

Memory Hierarchy Levels

Semnan University

Block : unit of copying

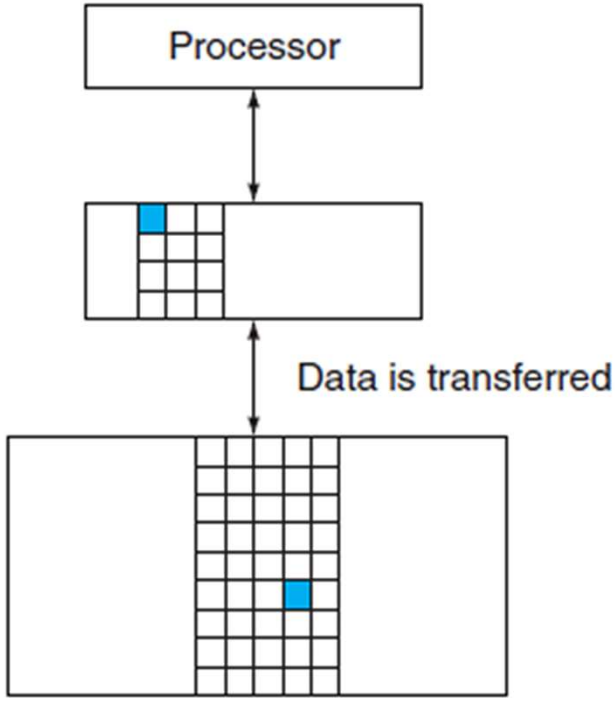
- May be multiple words

If accessed data is present in upper level

- **Hit:** access satisfied by upper level
  - Hit ratio: hits/accesses

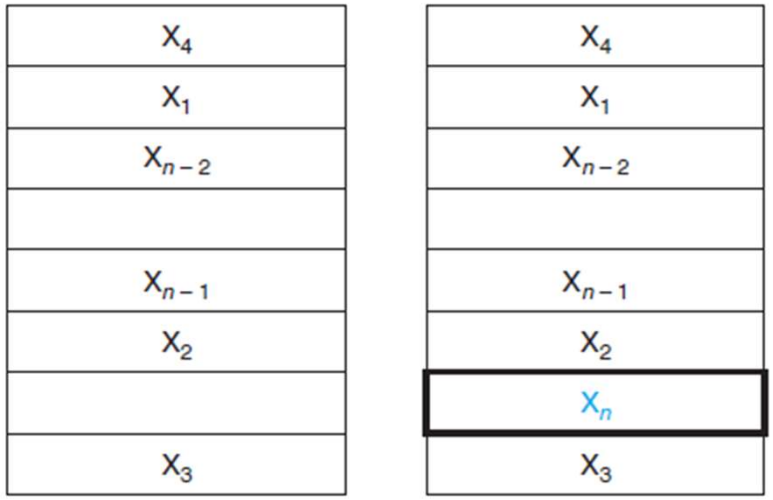
If accessed data is absent

- **Miss:** block copied from lower level
  - Time taken: miss penalty
  - Miss ratio: misses/accesses = 1 – hit ratio
- Then accessed data supplied from upper level



Cache Memory

- **Cache memory**
  - The level of the memory hierarchy closest to the CPU
- **Given accesses** X1,…,Xn−1,Xn



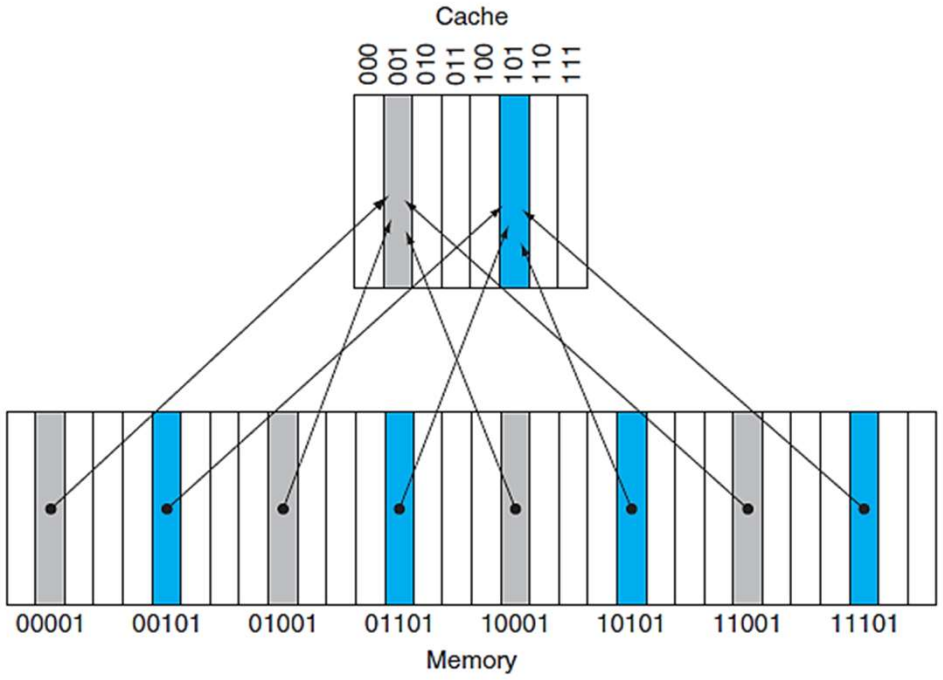
a. Before the reference to X_n b. After the reference to X_n

How do we know if the data is present?

Where do we look?

Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

Tags and Valid Bits

- **How do we know which particular block is stored in a cache location?**
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag

- **What if there is no data in a location?**
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

Cache Example

Semnan University

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Semnan University

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Semnan University

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Semnan University

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Semnan University

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

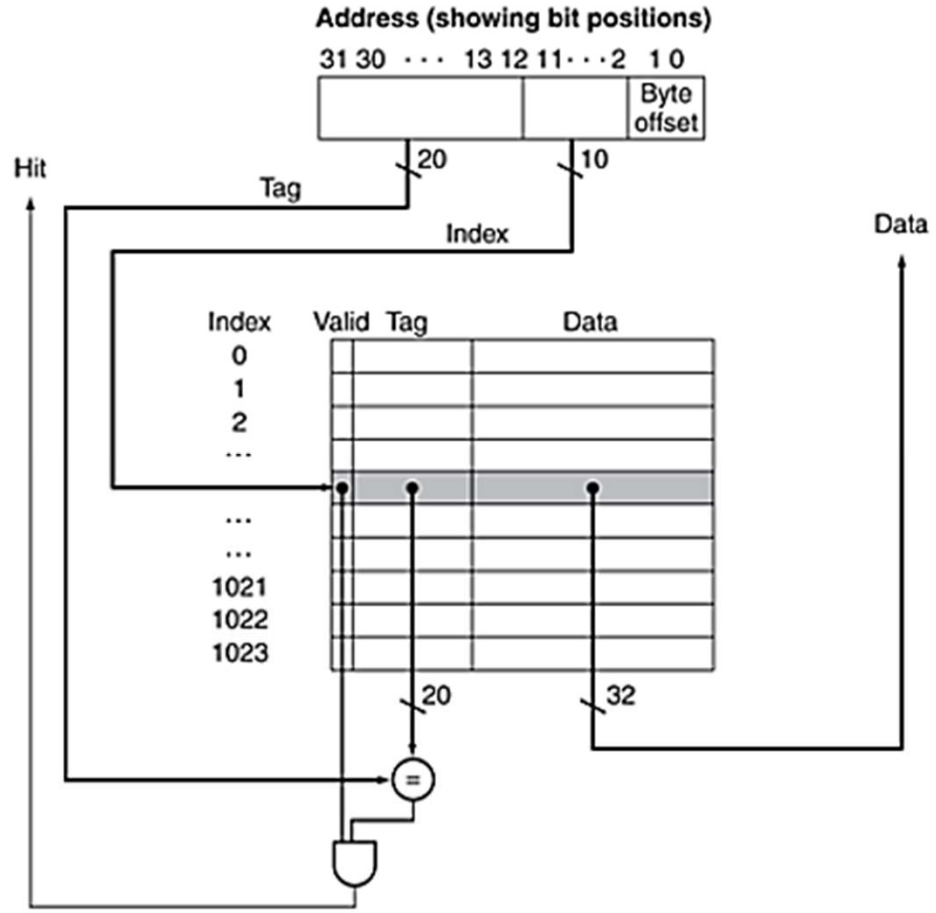
Cache Example

Semnan University

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

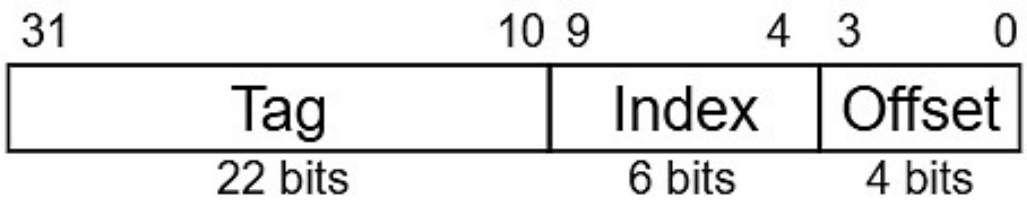
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Address Subdivision



Example: Larger Block Size

- **64 blocks, 16 bytes/block**
  - To what block number does address 1200 map?
- **Block address** = [1200/16] = 75
- **Block number** = 75 modulo 64 = 11



Address = 1200 = 0b10010110000

Explanation

- Address = 1200 = 0b10010110000
- Byte addressing so each block covers 16 addresses
  - 16 addresses represented by 4 bits
- Remove lower 4 bits from 0b10010110000
  - 0b1001011 = 75 = block address
- Next 6 bytes are the index
  - 0b001011 = 11 = cache block number
- Remaining bits are the tag
  - 0b1 = tag

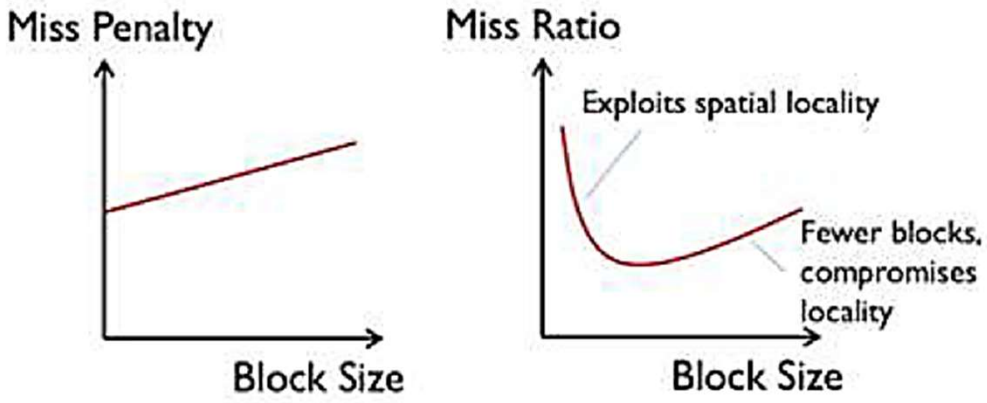
Block Size Considerations

Semnan University

- **Larger blocks should reduce miss rate**
  - Due to spatial locality
- **But in a fixed-sized cache**
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
  - Larger blocks ⇒ pollution
- **Larger miss penalty**
  - Can override benefit of reduced miss rate
  - Early restart and critical-word-first can help

Block Size Tradeoffs

- Larger block sizes...
  - Take advantage of spatial locality
  - Incur larger miss penalty since it takes longer to transfer the block into the cache
  - Can increase the average hit time and miss rate



Direct-Mapped Cache Problem: Conflict Misses

	Word Address	Cache Line index	Hit/ Miss
Loop A: Pgm at 1024, data at 37:	1024	0	HIT
	37	37	HIT
	1025	1	HIT
	38	38	HIT
	1026	2	HIT
	39	39	HIT
	1024	0	HIT
	37	37	HIT
	...		
Loop B: Pgm at 1024, data at 2048:	1024	0	MISS
	2048	0	MISS
	1025	1	MISS
	2049	1	MISS
	1026	2	MISS
	2050	2	MISS
	1024	0	MISS
	2048	0	MISS

Assume:
- 1024-line DM cache
- Block size = 1 word

Consider looping code, in steady state

Assume WORD, not BYTE, addressing

Inflexible mapping (each address can only be in one cache location) → Conflict misses!

Write-Through

Semnan University

- **On data write hit:**
  - Could just update the block in cache
  - But then cache and memory would be inconsistent
- **Write through:**
  - Also updates memory
- **But makes writes take longer:**
  - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
    - Effective CPI = 1 + 0.1 × 100 = 11
- **Solution: write buffer:**
  - Holds data waiting to be written to memory
  - CPU continues immediately
    - Only stalls on write if write buffer is already full

Write-Back

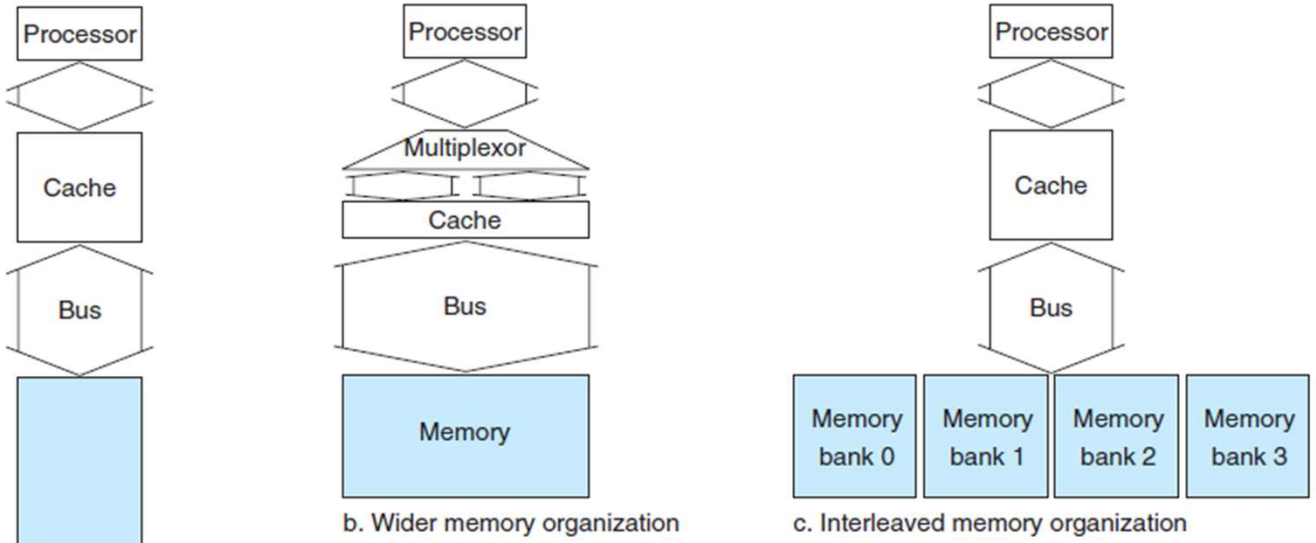
Semnan University

- **Alternative**: On data-write hit, just update the block in cache
  - Keep track of whether each block is dirty

- **When a dirty block is replaced**:
  - Write it back to memory
  - Can use a write buffer to allow replacing block to be read first

Main Memory Supporting Caches

- **Use DRAMs for main memory**
  - Fixed width (e.g., 1 word)
  - Connected by fixed-width clocked bus
    - Bus clock is typically slower than CPU clock
- **Example cache block read**
  - 1 bus cycle for address transfer
  - 15 bus cycles per DRAM access
  - 1 bus cycle per data transfer
- **For a 4-word block, 1-word-wide DRAM**
  - Miss penalty = 1+4×15+4×1=65 bus cycles
  - Bandwidth = 16 bytes/65 cycles=0.25 B/cycle



4-word Wide Memory

- **Miss Penalty:** 1+15+1=17 bus cycles
- **Bandwidth:** 16 bytes/17 cycles=0.94 B/cycle

4-bank Interleaved Memory

- **Miss Penalty:** 1+15+4×1=20 bus cycles
- **Bandwidth:** 16 bytes/20 cycles=0.8 B/cycle

Advanced DRAM Organization

Semnan University

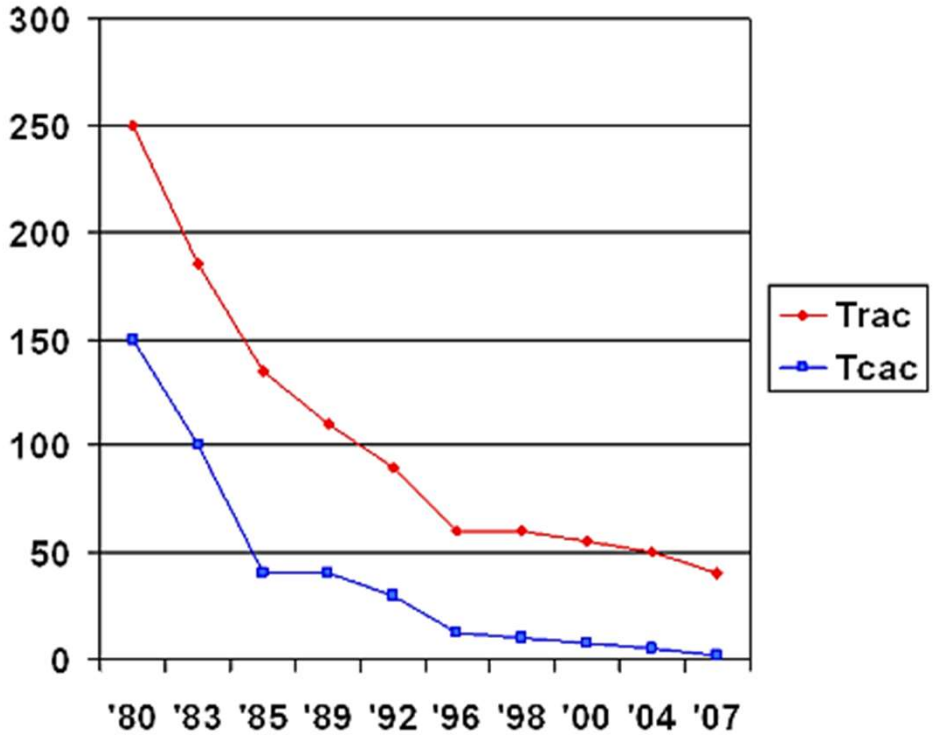
- **Bits in a DRAM are organized as a rectangular array**
  - DRAM accesses an entire row
  - **Burst mode:** supply successive words from a row with reduced latency

- **Double data rate (DDR) DRAM**
  - Transfer on rising and falling clock edges

- **Quad data rate (QDR) DRAM**
  - Separate DDR inputs and outputs

DRAM Generation

Year	Capacity	$/GB
1980	64Kbit	$1500000
1983	256Kbit	$500000
1985	1Mbit	$200000
1989	4Mbit	$50000
1992	16Mbit	$15000
1996	64Mbit	$10000
1998	128Mbit	$4000
2000	256Mbit	$1000
2004	512Mbit	$250
2007	1Gbit	$50



Measuring Cache Performance

Semnan University

- **Components of CPU time**
  - Program execution cycles
    - Includes cache hit time
  - Memory stall cycles
    - Mainly from cache misses
- **With simplifying assumptions:**

$$\text{Memory stall cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

Cache Performance Example

Semnan University

Given

- I-cache miss rate = 2%, D-cache miss rate = 4%
- Miss penalty = 100 cycles, Base CPI (ideal cache) = 2, Load & stores are 36% of instructions

Miss cycles per instruction

- I-cache: 0.02×100=2 D-cache: 0.36×0.04×100=1.44
- **Actual CPI**
- Actual CPI=2+2+1.44=5.44
- **Ideal CPU**
- Ideal CPU=5.44/2=2.72 times faster

Average Access Time

Semnan University

- **Hit time is also important for performance**
- **Average Memory Access Time (AMAT)**:

$$AMAT = Hit\ time + Miss\ rate \times Miss\ penalty$$

  - **Example**:
  - CPU with 1 ns clock
    - Hit time = 1 cycle
    - Miss penalty = 20 cycles
    - I-cache miss rate = 5%
  - Calculation:

- $AMAT = 1 + 0.05 \times 20 = 2$ ns
- **Result**: 2 cycles per instruction

Performance Summary

- **When CPU performance increased**
  - Miss penalty becomes more significant

- **Decreasing base CPI**
  - Greater proportion of time spent on memory stalls

- **Increasing clock rate**
  - Memory stalls account for more CPU cycles

Can’t neglect cache behavior when evaluating system performance

Associative Caches

Semnan University

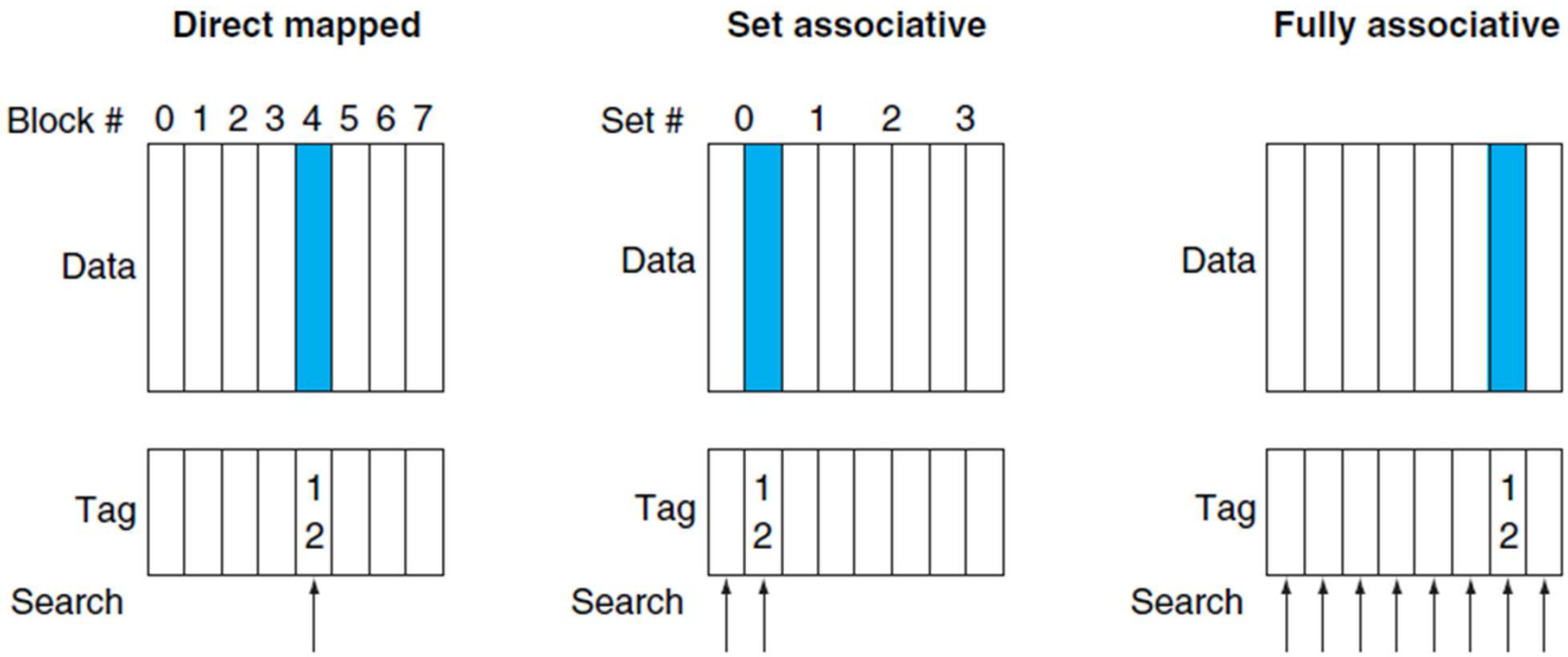
Fully Associative

- Allow a given block to go in any cache entry.
- Requires all entries to be searched at once.
- Comparator per entry (expensive).

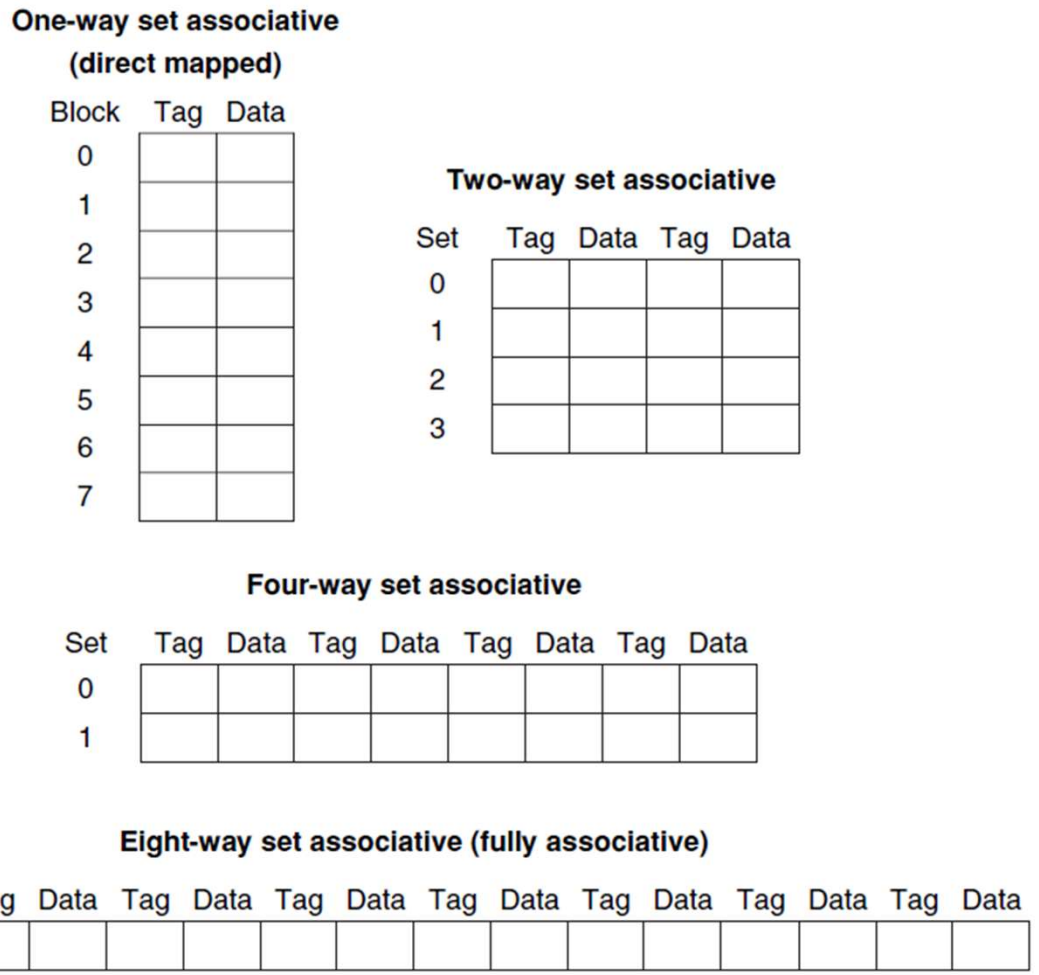
n-way Set Associative

- Each set contains n entries.
- Block number determines which set:
  - (Block number) modulo (#Sets in cache).
- Search all entries in a given set at once.
- n comparators (less expensive).

Associative Cache Example



Spectrum of Associativity



Associativity Example

Semnan University

- **Compare 4-block caches**
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- **Direct mapped**

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	Mem[0]			
8	0	miss	Mem[8]			
0	0	miss	Mem[0]			
6	2	miss	Mem[0]		Mem[6]	
8	0	miss	Mem[8]		Mem[6]	

Semnan University

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0	miss	Mem[0]	Mem[8]		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- Fully associative

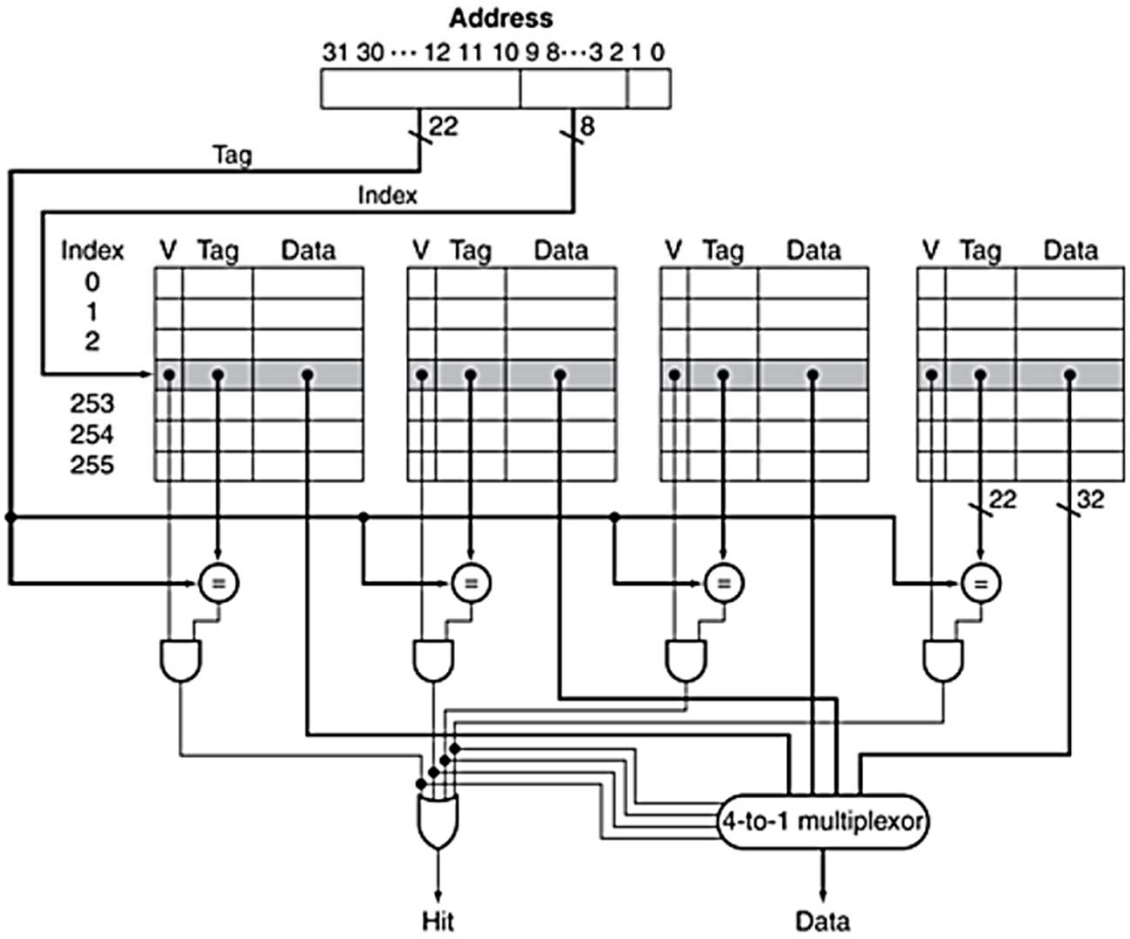
Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

How Much Associativity

- **Increased associativity decreases miss rate**
  - But with diminishing returns

- **Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000**
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Set Associative cash organization



Replacement Policy

- **Direct mapped**: no choice
- **Set associative**
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- **Least-recently used (LRU)**
  - Choose the one unused for the longest time
  - Simple for 2-way, manageable for 4-way, too hard beyond that
- **Random**
  - Gives approximately the same performance as LRU for high associativity

Multilevel Caches

- **Primary cache attached to CPU**
  - Small, but fast
- **Level-2 cache**
  - Services misses from primary cache
  - Larger, slower, but still faster than main memory
- **Main memory**
  - Services L-2 cache misses
- **Some high-end systems include L-3 cache**

Multilevel Cache Example

- **Given**
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns

- **With just primary cache**
  - Miss penalty = 100ns / 0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Example (cont.)

Semnan University

- **Now add L-2 cache**
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- **Primary miss with L-2 hit**
  - Penalty = 5ns / 0.25ns = 20 cycles
- **Primary miss with L-2 miss**
  - Extra penalty = 400 cycles
- **CPI =** $1+0.02\times20+0.005\times400=3.4$
- **Performance ratio =** $9/3.4=2.6$

Multilevel Cache Considerations

- **Primary Cache**
  - Focus on minimal hit time
- **L-2 Cache**
  - Focus on low miss rate to avoid main memory access
  - Hit time has less overall impact
- **Results**
  - L-1 cache usually smaller than a single cache
  - L-1 block size smaller than L-2 block size

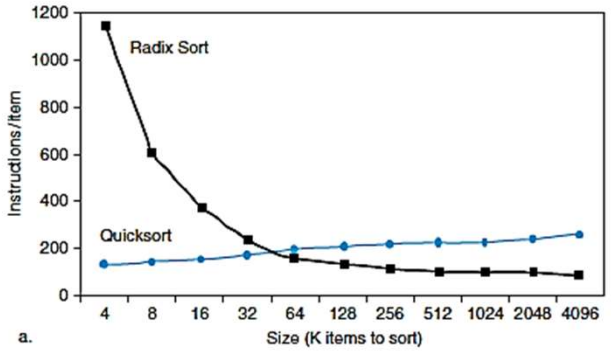
Interactions with Advanced CPUs

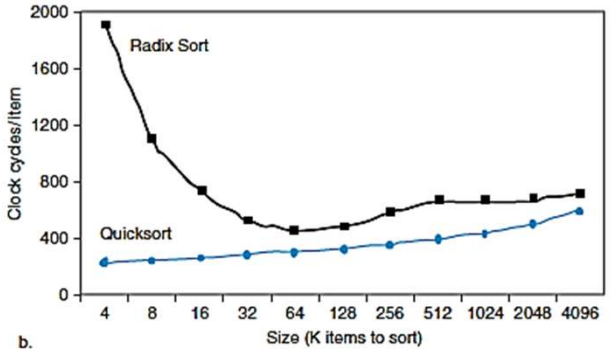
- **Out-of-order CPUs can execute instructions during cache miss**
  - Pending store stays in load/store unit
  - Dependent instructions wait in reservation stations
    - Independent instructions continue

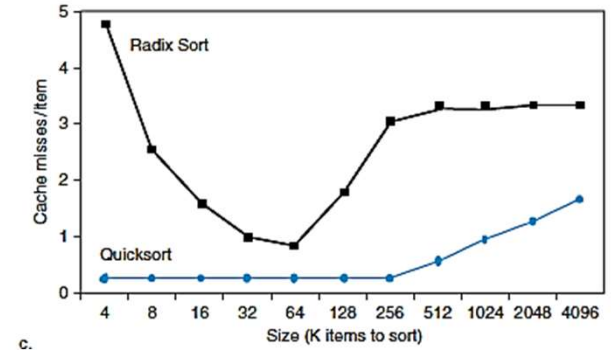
- **Effect of miss depends on program data flow**
  - Much harder to analyze
  - Use system simulation

Interactions with Software

- Misses depend on memory access patterns
  - Algorithm behavior
  - Compiler optimization for memory access







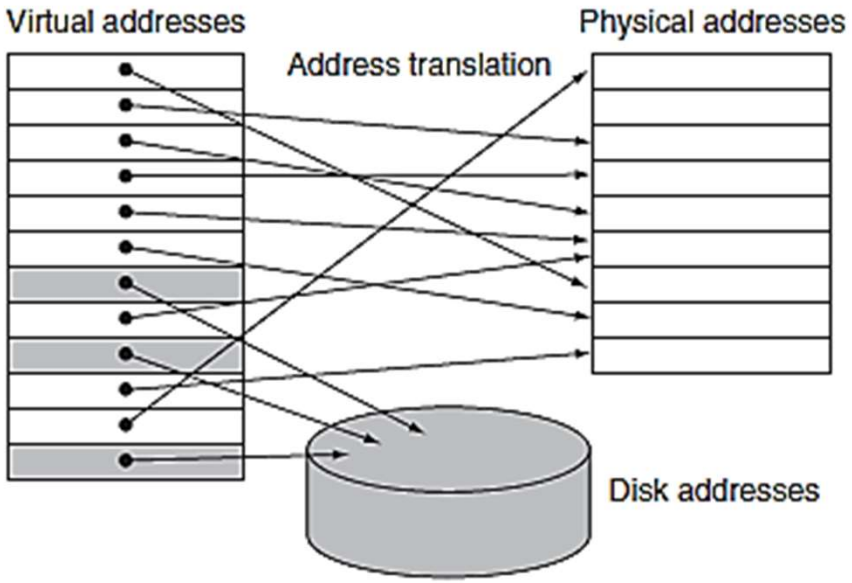
Virtual Memory

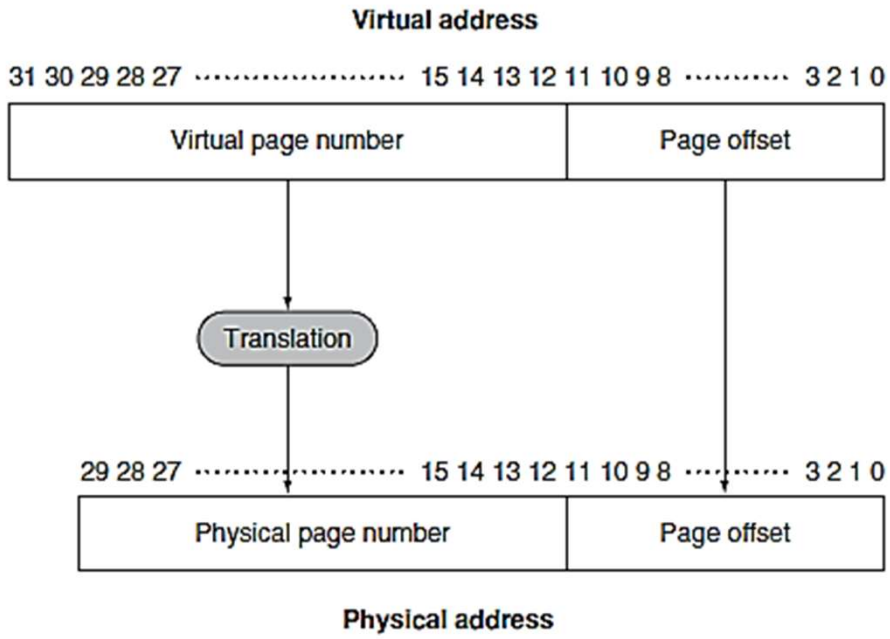
- Use main memory as a “cache” for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM “block” is called a page
  - VM translation “miss” is called a page fault

Address Translation

Semnan University

- Fixed-size pages (e.g., 4K)





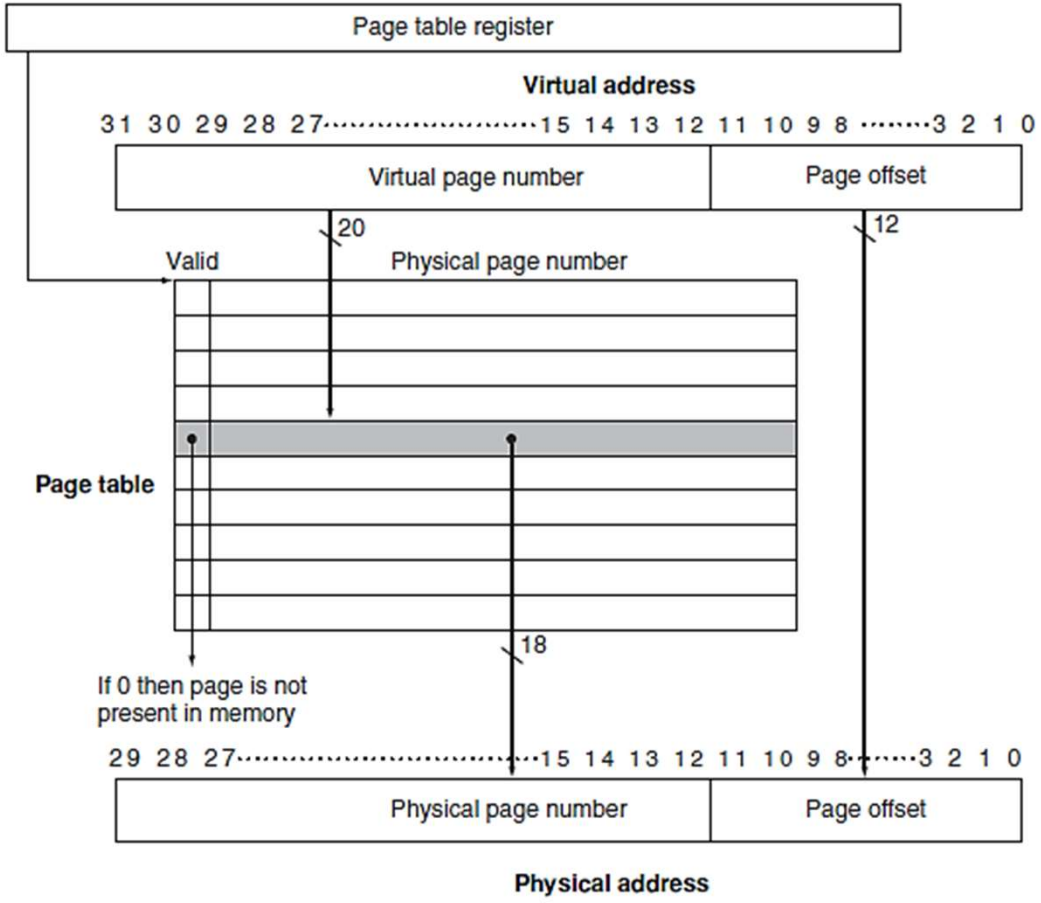
Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code
- Try to minimize page fault rate
  - Fully associative placement
  - Smart replacement algorithms

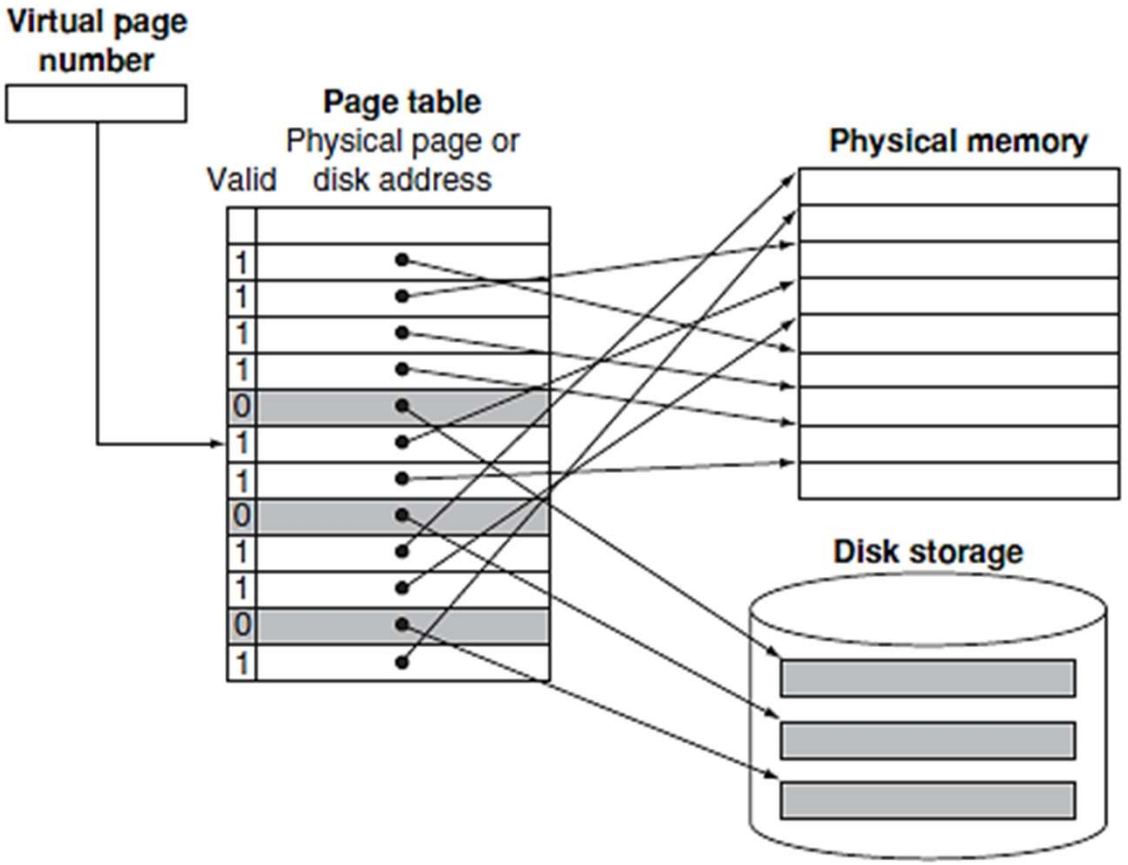
Page Tables

- **Stores placement information**
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- **If page is present in memory**
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- **If page is not present**
  - PTE can refer to location in swap space on disk

Translation Using a Page Table



Mapping Pages to Storage



Replacement and Writes

- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write-through is impractical
  - Use write-back
  - Dirty bit in PTE set when page is written