IT 110 Lecture 1

Introduction and Components

What is a system?

A system is a collection of components linked together and organized in such a way as to be recognizable as a single unit.

What is an architecture?

The fundamental properties, and the patterns of relationships, connections, constrains, and linkages among the components and between the system and its environment ate known collectively as the architecture of the system.

Elements of an information system architecture

- Hardware
- Software
- Data
- People

Components

Each of these can itself be a system with its own architecture.

- Networks ← Links

Abstraction of hardware as a programming language

- Input/output
- Arithmetic, logic, and assignment
- Selection, conditional branching (if-then-else, if-goto)
- Looping, unconditional branching (while, for, repeat-until, goto)

Counting Systems

- Base 10 counting :
- Ten one digit numbers (0–9)
- To expand beyond 1-digit, add a position on the left, representing the next power of ten.

Each position represents a power of ten (a *positional number system*).

For examole : 315,826

3	1	5	8	2	6
3×10^5	1×10^4	5×10^3	8×10^2	2×10^1	6×10^0

- **Base 2 counting**

  Two one digit numbers (0–1)

  To expand beyond 1-digit, add a position on the left, representing the next power of two.

0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7

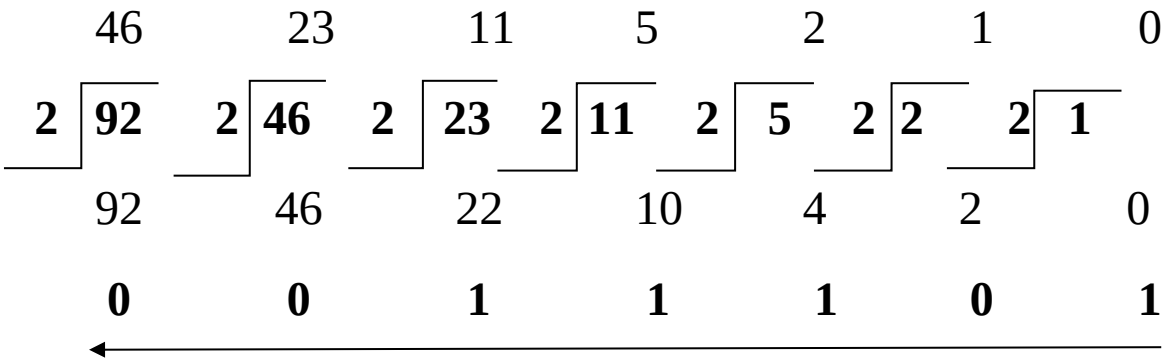
1	**0**	**1**
$\mathbf{1 \times 2^2}$	$\mathbf{0 \times 2^1}$	$\mathbf{1 \times 2^0}$

- **Leading zeros**
    - Are insignificant, but often written to indicate the number of *bits* in a quantity.

  For example : 0110 = 110

- **Converting to and from binary**
- Base 10 to base 2 conversion**: repeated division with remainders**

Example: Convert $(92)_{10}$ to binary.



The answer from left to right is $(1011100)_2$

- Base 2 to base 10 conversion: **repeated multiplication and addition**

Example: Convert $(1011100)_2$ to decimal

1 0 1 1 1 0 0

$2^6\times1$ $2^5\times0$ $2^4\times1$ $2^3\times1$ $2^2\times1$ $2^1\times0$ $2^0\times0$ } add

$= 0 + 0 + 4 + 8 + 16 + 0 + 64 =$ **$(92)_{10}$**

- **Binary is cumbersome** ثقيل، بطئ

Base 10	Base 2	Base 8	Base 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Long strings of 1's and 0's are hard to read. Group into sets or 3 (octal) or 4 (hexadecimal).

Example: Rewrite 110111100101_2 as octal and hexadecimal.

- Group by 3: 110 111 100 101 → 6745_8
- Group by 4: 1101 1110 0101 → $DE5_{16}$

IT 110 Lecture 3
Signed Integer Representations

Binary subtraction:

Important notes:

- Always use zero fill to keep your work consistent to help you doing calculations accurately.
- To group binary and octal use 3-bianry digit numbers
- To group hexadecimal use 4-bianry digit numbers
- In the binary operations always work with 8-bits

*** البنود التالية: إلا ما يجي سؤال عليها، حسب ما ذكره الدكتور علي ***

1. **Signed magnitude**:
   a. Write the two bits as positives.
   b. To get the nagation of the negative bits just replace the most segnificant by "1", which means negative sign.
   c. Add the two bits together, but you will always get a wrong answer with the signed numbers subtraction.
2. **1's complement**: notice that the most significant bit (the left most bit) represents a negative number if it is "1", and it represnts a positive number if it is "0".
   2.1.1. Flip the bits of the negative number: "swap one's and zero's".
   2.1.2. Write the positive bit underneath the negated bit.
   2.1.3. Add the two bits together, if you got an overflow, add it to the result, and that will be the 1's complement bits.
3. **2's complement**: Find the "2's" complement:
   3.1.1. Change the bits of the negative number; going from right to left, invert every digit after the first "1".
   3.1.2. Write the positive bit underneath the negated bits.
   3.1.3. Add the two bits together, if you got an overflow, ignore (truncate) it.

3.1.4. If the most significant number (the 8th bit, not the overflow) in the result is "1" it means that result still in the negative format, so we revers it by changing the bits again, going from right to left, and inverting every digit after the first "1".

IT 110 Lecture 4
Little Man Computer and Instruction Cycle

*** من المهم معرفة الأكواد ورموزها ***

Mnemonic	Code	Description
LDA	5XX	Load calculator with data from box XX
STO	3XX	Store calculator value in box XX
ADD	1XX	Add data in box XX to calculator
SUB	2XX	Subtract data in box XX from calculator
IN	901	Get input from inbox, put in calculator
OUT	902	Write calculator total to outbox
HLT	000	Stop executing
BRZ	7XX	Zero? Next instruction is in box XX
BRP	8XX	Positive? Next instruction is in box XX
BR	6XX	Next instruction is in box XX
DAT		Data storage reserved

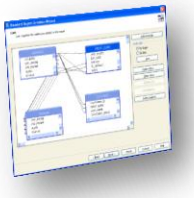
Generations of programming languages

- First generation: programmed directly in binary using wires or switches.

- Second generation: assembly language. Human readable, converted directly to machine code.

- Third generation: high-level languages, while loops, if-then-else, structured. Most programming today, including object-oriented.

- Fourth generation: 1990s natural languages, non-procedural, report generation. Use programs to generate other programs. Limited use today.



Generations of programming languages

- Key idea: Regardless of the language of writing, computers only process machine code.
- All non-machine code goes through a translation phase into machine code.
  - Code generators
  - Compilers
  - Assemblers

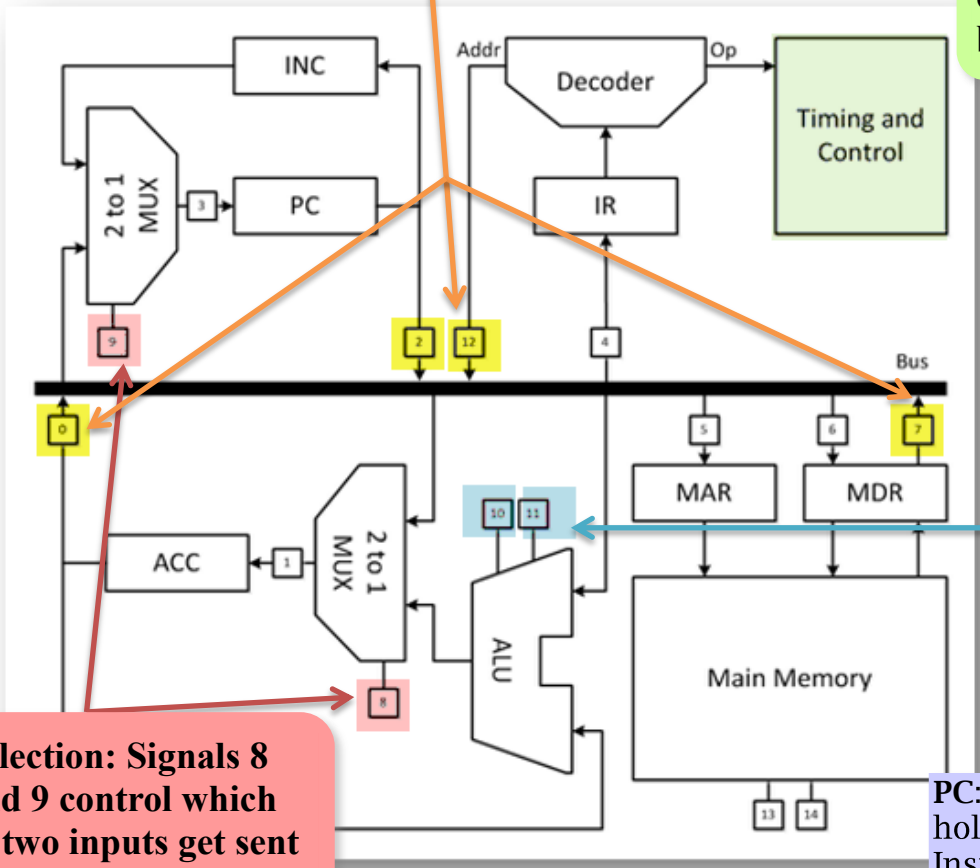
IT 110 Lecture 6

Fetch/Execute Cycle

Bus Access: Signals 0, 2, 7, and 12 control which data gets written to the bus.

***** مهم: التعاريف والجدول وأرقام البوابات *****

Control signals: determine order of operations, access to bus, loading of registers, etc.



ALU Operations: Signals 10 and 11 choose among addition, subtraction, multiplication and division performed by the ALU.

Selection: Signals 8 and 9 control which of two inputs get sent to output.

Number	Operation	Number	Operation
0	ACC→bus	8	ALU→ACC
1	Load ACC	9	INC→PC
2	PC→bus	10	ALU operation
3	Load PC	11	ALU operation
4	Load IR	12	Addr→bus
5	Load MAR	13	CS
6	Bus→MDR	14	R/W
7	Load MDR		

PC: Program Counter register, holds the address of the current Instruction being executed.

IR: Instruction Register, holds the actual instruction being executed currently by the computer.

MAR: Memory Address Register, holds the address of a memory location.

MDR: Memory Data Register, holds the address of a memory location.

Decoder: assembles the complete instruction with its operands, ready for execution

Fetch/Execute Cycle

Summary

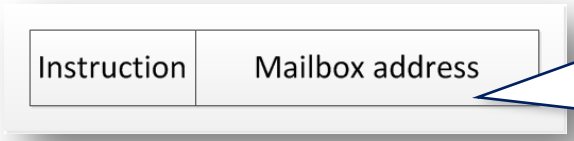
- The fetch/execute cycle consists of many steps and is implemented in the control unit as microcode.
- Control signals select operations, control access to the bus, and allow data to flow from component to component.
- Adding new instructions means modifying the microprogram in the control unit.

Instruction Set Architectures

ISA determines instruction formats

- The LMC is a one-address architecture (an accumulator-based machine).

e.g., the instruction ADD X



ADD takes two operands. One is implicit (the accumulator). The other is an address (location).

- There are other instruction set architectures, all based on the number of explicit operands.
  - 0-address (stack)
  - 1-address (accumulator)
  - 2-address
  - 3-address

*** إلا ما يجي عليها سؤال Address machines ***

0-Address Machines

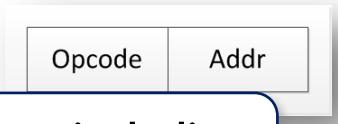
- All operands for binary operations are implicit on the *stack*. Only push(input)/pop(output) reference memory.
  e.g., calculating a = a * b + c – d * e

Code	# Memory Refs
PUSH A	1
PUSH B	1
MUL	0
PUSH C	1
PUSH D	1
PUSH E	1
MUL	0
SUB	0
ADD	0
POP A	1

In a stack-based machine, the stack is typically a set of very fast registers, minimizing trips to memory; 6 memory accesses, not including instruction fetch.

1-Address Machines

Accumulator is a source and destination. Second source is explicit.
e.g., calculating a = a * b + c – d * e

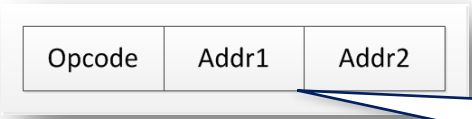


Code	# Memory Refs
LOAD A	1
MUL B	1
ADD C	1
STORE T1	1
LOAD D	1
MUL E	1
STORE T2	1
LOAD T1	1
SUB T2	1
STORE A	1

10 memory references, not including instruction fetch.

2-Address Machines

Two source addresses for operands. One source is also the destination.



Addr1 is both source and destination.

e.g., calculating a = a * b + c – d * e

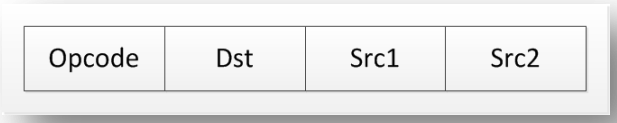
Code	# Memory
MOVE T1, A	2
MUL T1, B	3
ADD T1, C	3
MOVE T2, D	2
MUL T2, E	3
SUB T1, T2	3
MOVE A, T1	2

Using memory-to-memory operations, 18 memory accesses (not including instruction fetch). What if T1 and T2 were registers?

T1 here is destination and A is source

3-Address Machines

- One destination operand, two source operands, all explicit.



e.g., calculating a = a * b + c – d * e

Code	# Memory Refs
MPY T1, A, B	3
ADD T1. T1. C	3
MPY T2, D, E	3
SUB A, T1, T2	3

12 memory accesses, not including instruction fetch. What if T1, T2 were registers?

Code	# Memory Refs
MPY R1, A, B	2
ADD R1. R1. C	1
MPY R2, D, E	2
SUB A, R1, R2	1

6 memory accesses; general purpose registers make a substantial difference.

Comparison

*** المقارنات ما راح يجى عليها أى سؤال ***

Assume 8 registers (3 bits), 32 op-codes (5 bits), 15-bit addresses, 16-bit integers. Which ISA accesses memory the least?

	Instructions	Data refs	Total
0-address	10 x 20 bits = 200 bits	6 x 16 bits = 96 bits	296 bits
1-address	10 x 20 bits = 200 bits	10 x 16 bits = 160 bits	360 bits
1½-address	7 x 23 bits = 161 bits	6 x 16 bits = 96 bits	257 bits
2 address	7 x 35 bits = 245 bits	18 x 16 bits = 288 bits	519 bits
3-address	4 x 50 bits = 200 bits	12 x 16 bits =192 bits	392 bits
3-address (regs)	4 x 38 bits = 152 bits	6 x 16 bits = 96 bits	248 bits

Two clear winners: 1½-address (RISC) and 3-address with registers (CISC).

Summary

- The instruction set architecture determines the format of instructions (and therefore the assembly language).
- Four basic types with variations:
  - 0-address (stack)
  - 1-address (accumulator)
  - 2-address (register variant is 1½-address)
  - 3-address (with register variant)

ISA dramatically affects the number of times memory is accessed.

RISC vs. CISC

الرسك والسيسك مهمة جداً، لابد يجي عليها سؤال

Definitions

- CISC: Complex Instruction Set Computers.
- RISC: Reduced Instruction Set Computers.

- **What is CISC?**

- A type of microprocessor design. CISC processors require more CPU transistors in an effort to maximize code density in memory.
- Most common microprocessor designs such as the Intel 80x86 and Motorola 68K series followed the CISC philosophy.
- CISC was developed to make compiler development simpler.

- **CISC Attributes**

A 2-operand format: where instructions have a source and a destination. Register to register, register to memory, and memory to register commands.

Variable length instructions: where the length often varies according to the addressing mode.

Multi-clock cycle instructions

- **CISC motivation:**

1- High number of operations (300+).
2- Compilers have less work to do to translate HLL into machine code.
3- Large number of instruction formats
4- Multi-clock cycle instructions
5- Fewer registers; more memory access.
6- Large number of transistors, CPU complexity, therefore higher CPU prices.

- **CISC Disadvantages:**

1- instruction hardware become more complex.

individual instructions could be any length.
more time to execute.
slowing down the performance.

2- Many specialized instructions aren't used frequently enough .
20% of the available instructions are used.

3- Take More time to examine the condition code bits.

- **What is RISC?**
- a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions.
- The first RISC projects came from IBM, Stanford, and UC-Berkeley.

- **RISC Attributes**:
- One cycle execution time : RISC processors have a CPI (clock per instruction) of one cycle. This is due to a technique called PIPELINING .
- Pipelining: technique that allows for simultaneous execution of parts or instructions.
- large number of registers.

- **RISC motivation** :

1- Lower number of operations (150+)

2- Compilers have more work to do.
3- Small number of instruction formats
4- All instructions take one cycle.
5- Load/store architecture
6- Smaller number of transistors, lower CPU complexity, therefore lower CPU prices.

- **RISC Disadvantages:**

By making the hardware simpler, RISC architectures put a greater burden on the software.

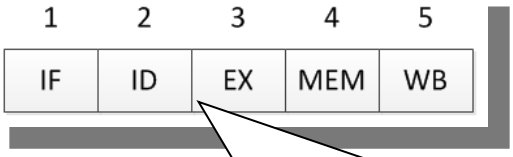
المقارنة بين المعالجين مهمة جداً جداً

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

CPU Performance Enhancements

- **General Enhancements :**
  - Use RISC-based techniques :
    - Fewer instruction formats, fixed-length → faster decoding.
    - More general purpose registers → fewer memory accesses .
  - Clock cycle and instruction cycle :
  - Most instructions take several clock cycles to execute:
    - Fetch the new instruction [IF].
    - Decode the instruction [ID].
    - Execute the instruction [EX].
    - Access memory (if needed) [MEM].
    - Write back to the registers [WB].
  - Each stage takes a clock cycle, so complete execution takes 5 cycles.

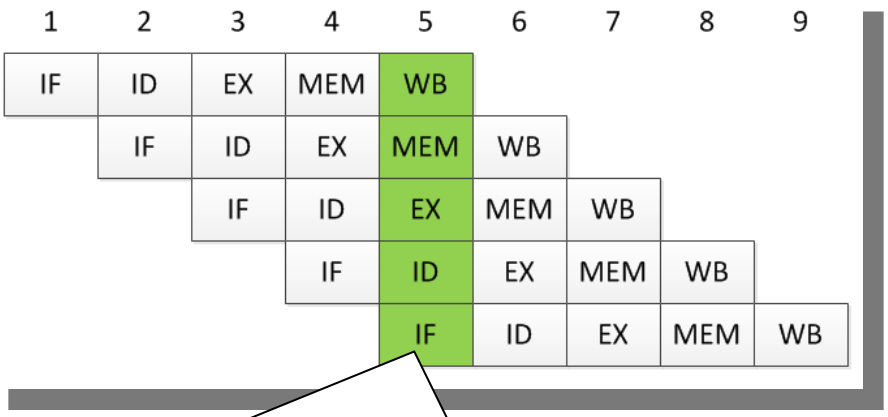
الخطوات الخمس مهمة
Instruction cycle



Notice that the ALU used in stage 3 is idle in stages 1, 2, 4, and 5. The same can be said for other components if they are all discrete. Underutilized hardware!

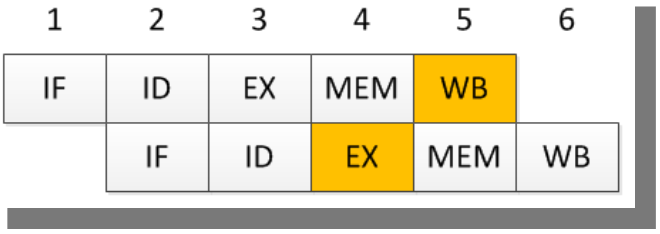
- **Solution: offset and overlap in a *pipeline*.**

pipeline
مهم جداً



By cycle 5, the CPU is executing 5 instructions at once. After this, one instruction completes every cycle. An *n*-stage pipelined CPU is *n* times faster than a non-pipelined CPU.

- **Problems with pipelining :**
  - Dependencies (register interlock)—if an instruction needs a result from the immediately preceding instruction, that result won't be written back until WB, but the result is needed in EX.



Dependencies
الاعتمادية: بعض العمليات معتمدة على البعض فلا يمكن تنفيذ بعض الخطوات إلا بعد الانتهاء من الخطوة السابقة

  - Branching—when the instruction being executed is a branch, we can't know if the branch will be taken until after stage 3. But by that time, other instructions are "in flight."

Branching
التفرع: عندما يبدأ تنفيذ التعليمات، يتفرع التنفيذ، وعندها لا نعرف إذا كان هناك حاجة للتفريع إلا بعد المرحلة الثالية

Summary

- RISC-based CPUs offer general performance enhancements due to simplified formats and single-clock cycle execution.
- Pipelining allows multiple instructions to be in various stages of execution at once.
- Superscalar processing duplicates pipelines in a single core to have multiple instructions executing simultaneously.

  Superscalar يجمع ما بين pipeline وما بين parallel

- Data dependencies and branches are hazards to both pipelining and superscalar architectures.

Memory Performance Enhancements

Within the instruction fetch-execute cycle, the slowest steps are those that require memory access. Therefore, any improvement in memory access can have a major impact on program processing speed. The memory in modern computers is usually made up of dynamic random access memory circuit chips . DRAM is inexpensive. Each DRAM chip is capable of storing millions of bits of data.

Static RAM, or SRAM, is an alternative type of random access memory that is two to three times as fast as DRAM.
SRAM design requires a lot of chip real estate compared to DRAM. 1 or 2 MB of SRAM requires more space than 64MB of DRAM, and will cost more.

■ **Three different approaches are commonly used to enhance the performance of memory:**

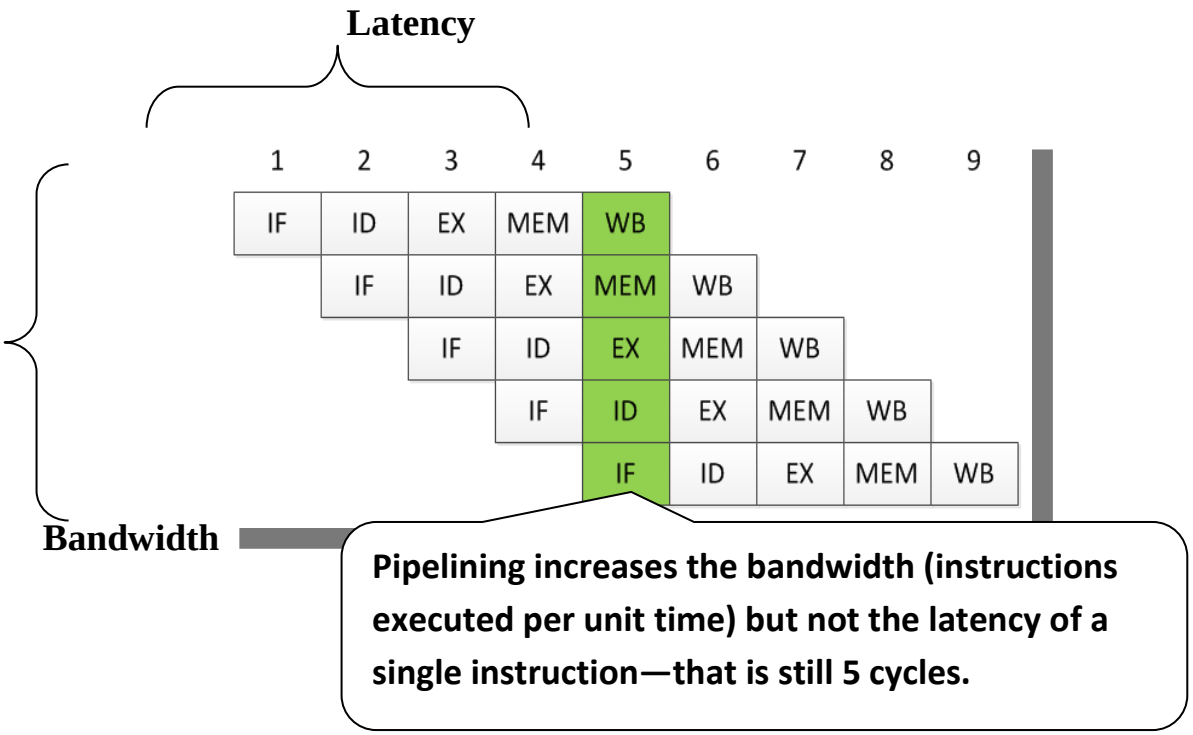
- Wide path memory access.
- Memory interleaving.
- Cache memory.

الخطوات الثلاث لتحسين أداء الذاكرة مهمة مع تعريف Latency, Bandwidth

All three are used simultaneously in the system design.

■**Wide path memory access :**
the simplest means to increase memory access is to widen the data path so as to read or write several bytes or words between the CPU and memory with each access; this technique is known as wide path memory access.
Accessing memory has high *latency* but also high *bandwidth*.

Latency : is the amount of time it takes for a round trip, i.e., the time from when the CS, R/W signals are asserted until the data is in the MDR.
Bandwidth : is the amount of data that can be returned per unit time.



- In the same manner as pipelining, memory bandwidth can be increased.
- Requests for memory aren't satisfied 1 byte at a time, but rather 4, 8, or even 16 bytes at a time.
- Requires a wider bus between CPU and memory.

Memory interleaving
تقسيم الميموري إلى أجزاء

■Memory interleaving :

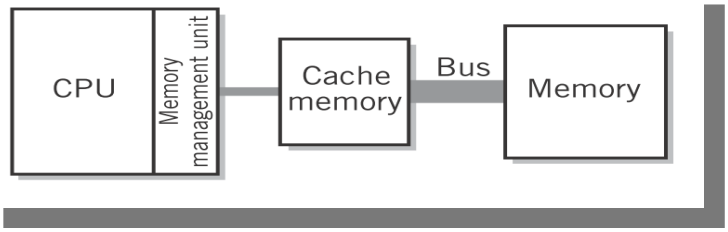
Another method for increasing the effective rate of memory access is to divide memory into parts, called memory interleaving, so that it is possible to access more than one location at a time. Then, each part would have its own address register and data register, and each part is independently accessible.
Memory can then accept one read/write request from each part simultaneously.
Although it might seem to you that the obvious way to divide up memory would be in blocks.

■ Cache memory :

A different strategy is to position a small amount of high-speed memory, for example, SRAM, between the CPU and main storage.

This high-speed memory is invisible to the programmer and cannot be directly addressed in the usual way by the CPU. Because it represents a ‘‘secret’’ storage area, it is called cache memory.

- Cache memory is the only technique that tries to minimize latency.
- DRAM has high latency but is inexpensive.
- SRAM has low latency but is expensive.
- Use a small amount of expensive SRAM as a buffer against the large amount of DRAM.



- Hit : Requested data exists in cache—very fast.
- Miss : Data not in cache, fetch from memory, copy into cache, and then treat as a hit.

- **Cache entries consist of:**

مهم: مكونات الكاش ميموري

  - Tag : address.
  - Data : copy of memory.
  - Dirty bit : indicates if data in cache is newer than contents of memory.

- **Cache replacement algorithm**
  - Once cache fills, a miss will cause an existing line to be replaced. Which one?
    - Least recently used (LRU).
    - First in first out (FIFO).
    - Least frequently used.
    - Random.
    - Etc.

مهم: خوارزمية التخلص من الداتا الموجودة فى الكاش ميمورى

- **What should happen on a memory write?** مهم
  - Write through—write to cache and then immediately write to memory. Safe, simple, slow.
  - Write back—write only to cache. Use dirty bit to write back to memory when line is replaced. Complicated, fast.

Cache *coherency* gets particularly tricky with multiple cores and multiple levels of cach.

Summary :

- Latency is the round trip time to deliver a single request.
- Bandwidth is the number of requests that can be fulfilled in a unit time.
- Three ways of improving memory performance:
  - Wide path memory access—increase bandwidth to memory by fetching multiple bytes at a time.
  - Memory interleaving—increase bandwidth to memory by fetching in parallel across blocks.
  - Cache memory—decrease latency to memory by having fast copies closer to the CPU. Must keep memory synchronized with cache.

IT 110 Lecture 11

Programmed I/O, Interrupts, and DMA

I : Input .

O: Output .

Input/Output Characteristics :

- Many orders of magnitude slower than memory.
- Character vs. block based.
- Burst vs. steady transfers.

Three approaches to I/O

مهم

- Programmed.
- Interrupt-driven.
- Direct memory access.

Programmed I/O :

- CPU is responsible for reading/writing to devices:
- Special “input” instruction on CPU.
- I/O data register and I/O address register.
- Each device is assigned a unique address.

- **Memory mapped I/O alternative :**
  - Treat the I/O device as a memory address for reads and writes. Simplifies programmer interface; slightly more complicated control circuitry.
- **Problems with all programmed I/O :**
  - Must check status bits to see if I/O is "ready."
  - Use a polling loop (busy-wait) to send and receive data to devices**.**

Interrupts :

- Busy-waits (polling) wastes resources but has simpler hardware.
- Alternative: After an I/O request from the CPU, let the I/O device notify the CPU when data is ready to be read (called an interrupt).

IRQ: stands for Interrupt Request

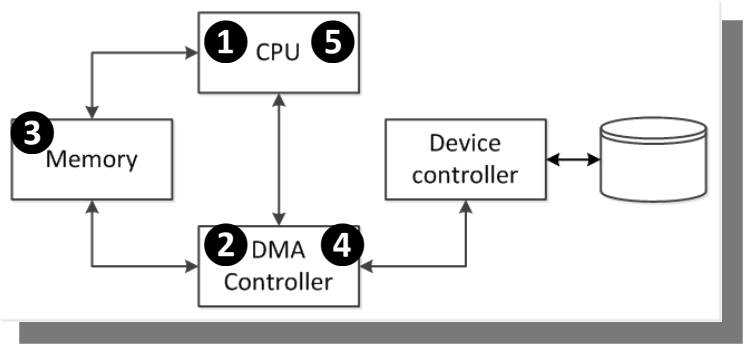
  - Each device is assigned an IRQ line (signal).
  - I/O controller sets IRQ line status high.
  - CPU detects IRQ at beginning of fetch/execute.
  - CPU saves state of running program and switches to an IRQ handler routine.
  - Routine services the request.
  - Control is returned to the previously running code
- **Problems with interrupt driven I/O :**
  - CPU still involved with each interrupt.
  - Only transfers a single byte/word.

Disk or network transfers may be hundreds or thousands of bytes. IRQ handler code may be hundreds of instructions. Still too much overhead.

DMA :

*** الرسمة والشروحات أدناه: مهمة وكلاهما مطلوب ***

- Direct Memory Access (DMA)
  - Add a specialized kind of CPU that can directly transfer data from device to memory.



  - Requires memory arbitration or dual-ported memory.

How the programmed I/O, DMA, and interrupt methodologies work together :

1. CPU uses PIO to specify memory address, operation (read/write), byte count, and block location on disk.
2. DMA controller initiates I/O with the device controller.
3. DMA controller receives data and transfers it to memory.
4. DMA controller interrupts CPU to notify data transfer is complete.
5. CPU handles interrupt. All bytes are in memory for processing.

Summary

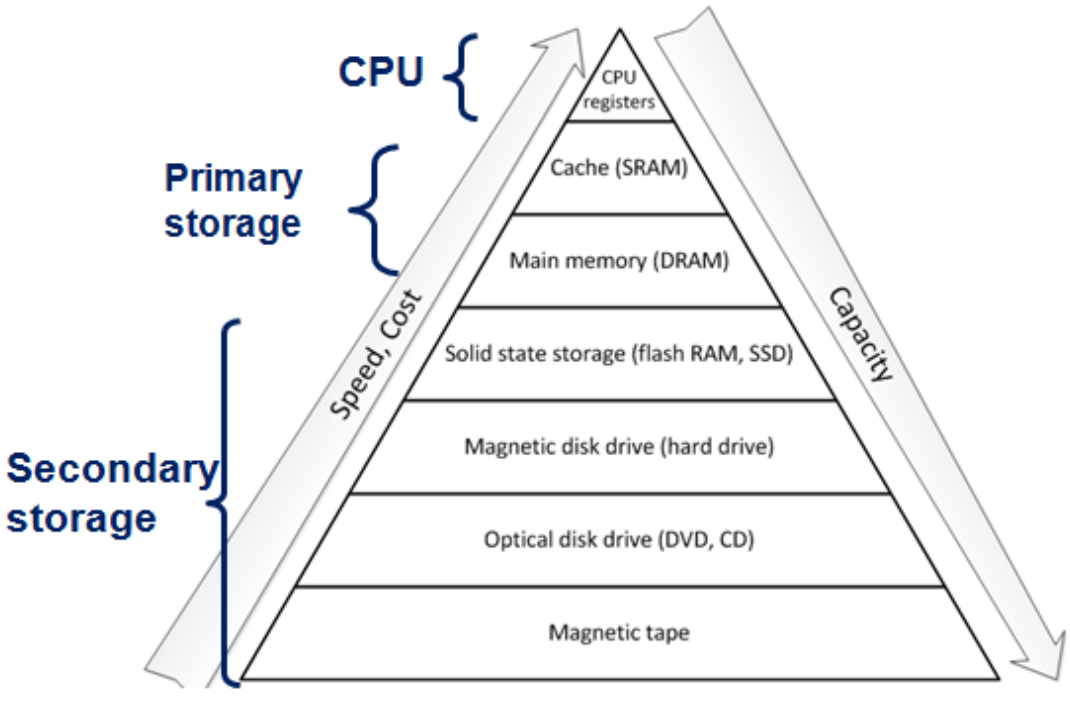
- Purely programmed I/O requires special I/O instructions, I/O data and address registers, and polling loops that waste CPU resources.
- Interrupt-driven I/O avoids busy-waiting but is unsuitable for large block transfers due to interrupt handler execution overhead.
- DMA combines PIO and IRQ handlers with a special controller to transfer large amounts of block data efficiently directly to memory.

Storage Hierarchy and Disk Technology

Storage Hierarchy :

Computer storage is often conceptualized hierarchically, based upon the speed with which data can be accessed.

- *Performance is driven by latency and bandwidth.*
- *The more layers away from the CPU . . .*
  - *. . . the higher the latency.*
  - *. . . the larger the capacity.*



*** الرسمة والشروحات مهمة وكلاهما مطلوب ***

At the top of the hierarchy are :
CPU registers used to hold data for the short term while processing is taking place.

cache memory(SRAM) : is a small fast memory that is used to hold current data and instructions.

The CPU will always attempt to access current instructions and data in cache memory before it looks at conventional memory.

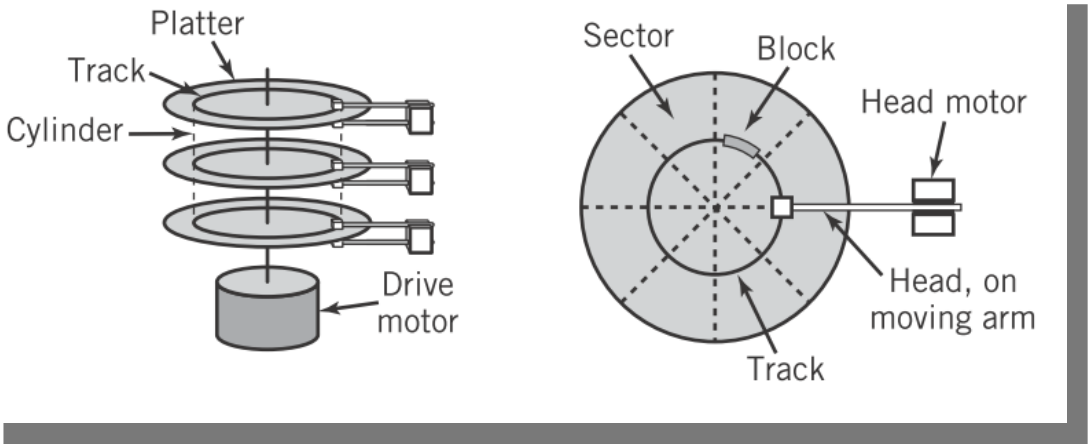
conventional memory (Main memory)(DRAM): The CPU accesses the data or instruction in conventional memory if cache memory is not present.

- Both conventional and cache memory are referred to as primary memory.
- Except for flash memory, access to secondary storage is significantly slower than primary storage.
- Flash memory uses a special type of transistor that can hold data indefinitely without power.
- The magnetic media used for disk and tape and the optical media used for DVD and CD disks also retain data indefinitely.
- Secondary storage has the additional advantage that it may be used to store massive amounts of data.
- Even though RAM is relatively inexpensive, disk and tape storage is much cheaper yet.
- Additional advantage that secondary storage may be used for offline archiving, for moving data easily from machine to machine, and for offline backup storage.

Magnetic Disk Technology : *** مهم جداً جداً ***

Terminology:

- Platter: a spinning disc within a drive, made of glass or aluminum, and coated with magnetic media.
- Head: floats above the media, reading or writing the magnetically encoded data.
- Track: a ring on a single platter.
- Cylinder: a track across all platters.
- Sector: a wedge shaped slice of a platter.
- Block: the intersection of a track and a sector.

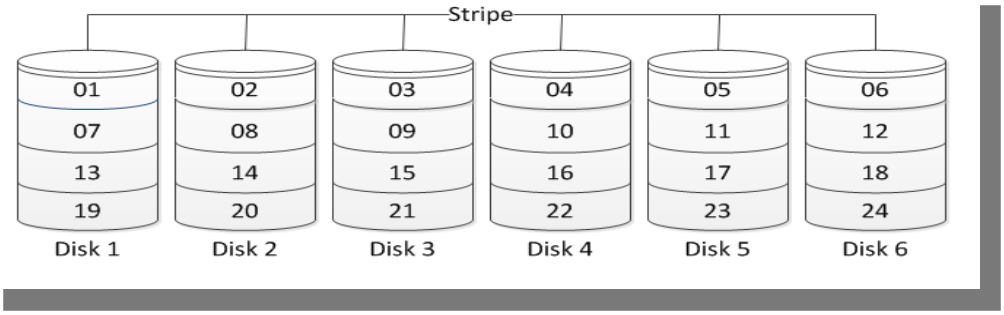


- Seek time: time to move the head to the desired track
- Latency time: time to rotate the desired sector to be under the head
- Transfer time: time to read a block after seek and latency are accounted for
- CAV (constant angular velocity): used by HDD; disk always spins at the same speed. Problem: wastes space on the outer rings
- CLV (constant linear velocity): The number of bits passing under the head is constant. Faster angular velocity at the inner tracks; slower on the outer.

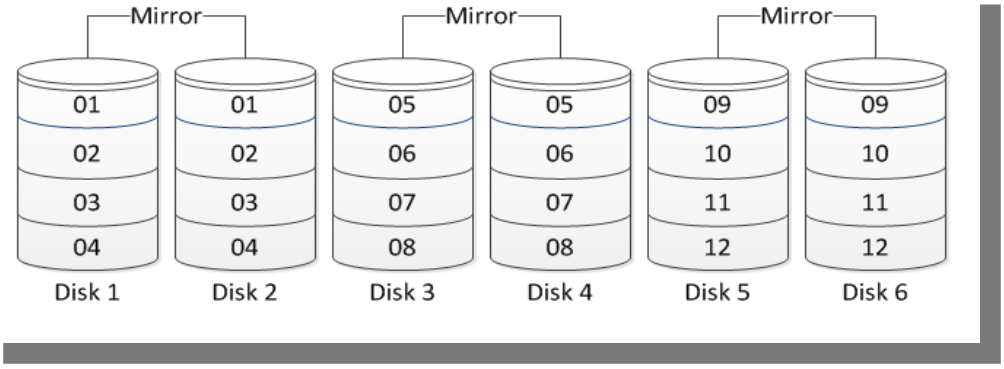
RAID : * مهم جداً مع معرفة أنواعه والتركيز على الرسومات *****

- Disks often fail because they are at least partly mechanical. RAID (redundant array of independent disks) attempts to improve redundancy and bandwidth.
- Combine three primary functions:
- Mirroring
- Striping
- Parity checks

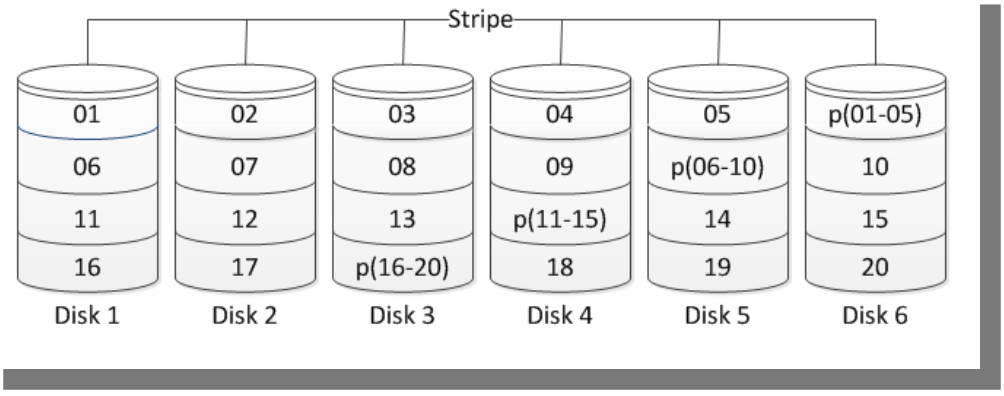
- ***RAID 0: Striping***



- ***RAID 1: Mirroring***



- ***RAID 5: Striping with distributed parity***



- ***RAID 10: Stripe across mirrors***

Summary :

- Memory hierarchy shows the inverse relationship between speed and capacity in computing systems.
- Magnetic disks have several kinds of latency: seek time, rotational delay, and transfer time.
- RAID attempts to compensate for latency and failures by employing striping, mirroring, and parity checks.

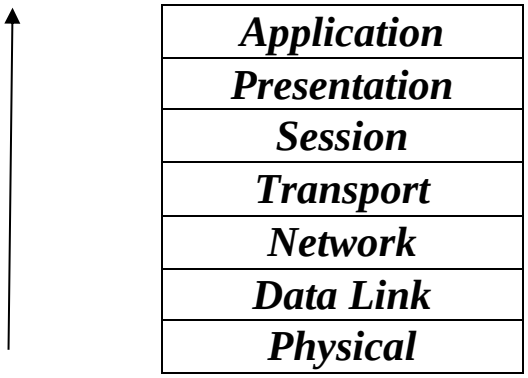
Data Communication Concepts

Elements of a network *** مهم جداً مع التعاريف ***

- ***Protocols*** : Rules about how messages are sent, received, directed, and interpreted. Like grammar in a human language.
- ***Messages*** : Data that is sent and received as part of a communication. Two parts: protocol header and data payload. Protocol header is the envelope in which data is carried.
- ***Media*** : Material through which the messages move. Wired—copper (electrical) or fiber (optical). Wireless—any non-conducting material (radio waves).
- ***Devices*** : Equipment that sends, receives, or directs messages through media. Endpoints or intermediate devices.

OSI and TCP/IP models: The Open Systems Interconnection Reference Model (OSI) is a theoretical model, developed overmanyyears as a standard by the International StandardsOrganization (ISO). TCP/IP is an older and more practical model, independently developed to meet the needs of the original Internet design, and regularly modified and updated to meet current needs.

- OSI Model: does not specify concrete protocols, but rather specifies the functions that concrete protocols will need to implement at each layer.



Physical : Transmits raw bits in either code words or symbols. No knowledge of the data it transmits. Examples: high and low voltages over copper twisted pair wire, or colors of light in fiber.

Data Link : Groups of bits called *frames* sent and received on a single network type. Handles synchronization and collision detection or avoidance. Protocol examples include Ethernet, Token Ring, FDDI, and 802.11 (wireless).

Network : Makes it possible to send units of information (*packets*) across different kinds of networks (*routing*). Uniform addressing schema, network congestion control. Protocol examples include IP (internet protocol), IPX (internetwork packet exchange), and ICMP (internet control message protocol).

Transport: Ensures reliable delivery of packets, error recovery, flow control, congestion control, and multiplexing of the network by several applications at once. Example protocols include TCP (transmission control protocol) and UDP (user datagram protocol).

Session : Provides enhanced end-to-end session services such as authentication and authorization. Example protocols include PAP (password authentication protocol), NetBIOS, and PPTP (point-to-point tunneling protocol)

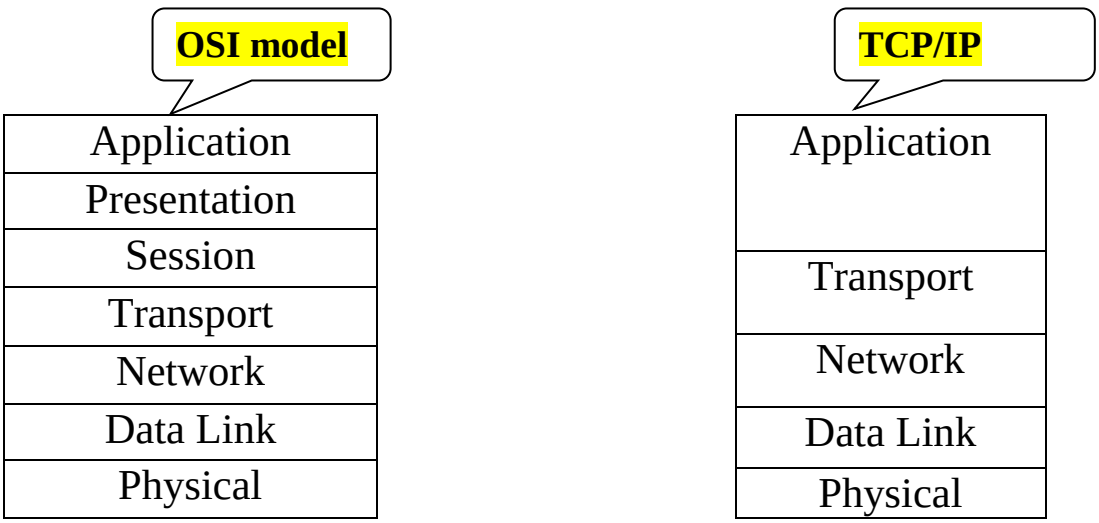
Presentation : Manages the way data is represented and formatted via encryption, compression, serialization, and encodings. Example protocols include ASCII and XML.

Application : Provides protocols for specific applications. Examples include FTP (file transfer protocol), SMTP (simple mail transfer protocol), SNMP (simple network management protocol), LDAP (lightweight directory access protocol), and HTTP (hypertext transfer protocol). Most are defined in RFCs.

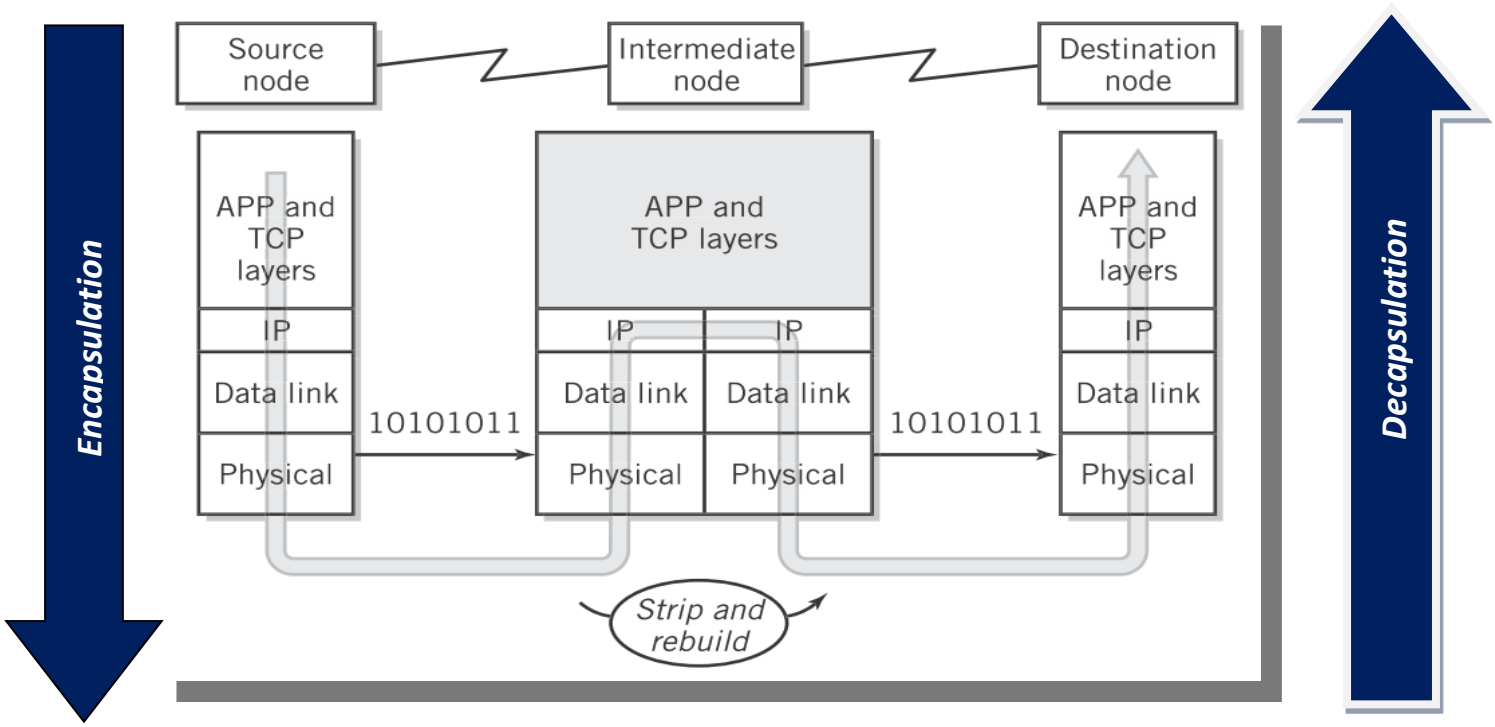
OSI and TCP/IP models:

- TCP/IP model: a real-world protocol stack used for most network communication today.
- Layer separate concerns and build interoperability between different manufacturers.
- Intermediate devices examine headers and reformat protocol data units for the next hop.

***** الفروقات بين البروتوكلين مهمة جداً *****

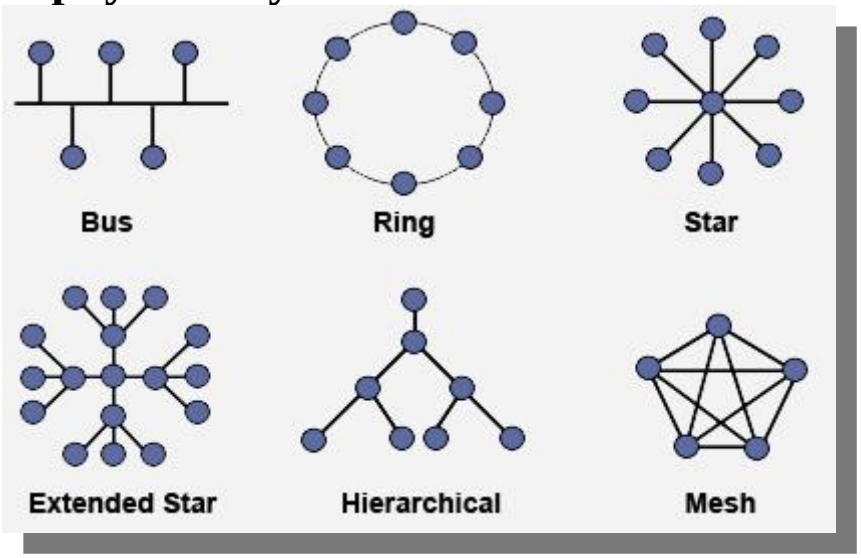


***** هذا الدایاجرام مع الشرح مهم جداً جداً *****
قد یطلب الرسم وقد یطلب الشرح أو کلاهما



Network Topologies : *** مهم جداً جداً ***

- **Logical vs. physical layouts :**



Summary

- Networks consist of protocols, messages, media, and devices.
- The OSI model provides seven layers of functionality that are concretely provided in the 5 layers of TCP/IP.
- As data moves down layers, it is encapsulated in the lower protocol data unit, and as it moves up, it is de-capsulated.
- Networks can be arranged logically and physically as busses, stars, rings, meshes or hybrids of each.