SEMESTER 1

COA

NOTES

Owner : Annamalai. A.

Start : 18 Aug 2025

End : 23 Sep 2025

SURFACE LEVEL OF COMPUTER ARCHITECTURE

Fundamental Operations

→ Any mathematical operation can be simplified to only 2 operations:
- ↳ Addition (+)
- ↳ Subtraction (-)

Multiplication (×):
Just repeated addition

Division (÷):
Just repeated sub.

Exponent:
Repeated × → Repeated +

Log:
Repeated ÷ → Repeated −

Factorial: Just multiplication
↳ Hence, multiple repeated +.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \dots$$

fact: +
exp: +
div: +

→ When it comes to computing there are three more operators:
- ↳ and (&)
- ↳ or (|)
- ↳ not (~)

Number Systems

→ There are 4 number systems widely used:
- ↳ Decimal (base 10)
- ↳ Binary (base 2)
- ↳ Octal (base 8)
- ↳ Hexadecimal (base 16)

★ Why are these the bases?

Decimal (10):
We have 10 fingers in total, which we commonly use to count numbers.

Binary (2):
Just true/false, on/off, yes/no etc.

Octal (8):
Taking one palm, the thumb can be a pointer, and the last 4 fingers can be split into two, making it 8.

Hexadecimal (16):
Similar to octal, but with both palms.

★ Why do we use decimal?

Us, humans usually count numbers with fingers, and we have 10 of those.

★ Why do computers use binary?

- Coreputers process using signals.
- ↳ Hence, only two states are needed —
  - 1 → on → pass signal
  - 0 → off → block signal

Working of a Computer

→ There are 4 main parts:
- ↳ SSD / secondary storage
- ↳ RAM
- ↳ Cache Memory
- ↳ Processor

It can be noted that memory plays an important role here.

SSD → stores O.S. and essential apps

RAM → Main memory - Store necessery items in small amounts.

Cache → Responsible for resuming from where we left.

Processor → Obviously, processes information.

Computer vs Kitchen

→ While cooking, the following can be noted:

Ingredients:

Outside (shop) → storage → shelf → stove

And the sizes are similar:

SSD	store room
RAM	shelf
Cache	freq. used ingredients (salt, sugar, pepper)
Processor	stove

(↑ size)

→ The process of transferring O.S. and apps from SSD to RAM is called booting.

Addressing

→ USB: Universal Serial Bus

→ 1 port can be connected to 256 external ports.

USB (Laptop)
↳
↳
↳ } 256 ports

→ Each port has an 8 bit address. Hence, no. of possibilities = $2^8 = 256$.

Because each bit is 0 or 1, and there are a total of 8 bits.

$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$

IP Address

→ IP: Internet Protocol
(Ensures data packets in a network reaches the right destination)

→ IP Address: A unique address defined for each comp. on a network for transferring data packets correctly.

→ IP uses this address to transfe data correctly.

→ IP address is defined by the manufacturer.

→ Each IP address has 32 bits (4 8-bits)

ex: 202 . 174 . 61 . 120
0-255 0-255 0-255 0-255

Hence, total no. of computers that can be connected:

$2^{32} = 4\ G \approx 4 \times 10^9$

or **4 Billion** computers

(* Today's world has more than 4B computers and we ran out out of IPv4 addresses)

Assembly Language (ASM)

MOV destination, location
move a data (location) to a register (dest.)

ADD dest., source 1, source 2
SUB dest., source 1, source 2

ADD c, a, b ⇒ c = a + b
SUB c, a, b ⇒

ADD a, b, c ⇒ a + b = c
SUB a, b, c ⇒ a - b = c
MUL a, b, c ⇒ a × b = c
DIV a, b, c ⇒ a / b = c

ex: $y = mx + c$

ASM

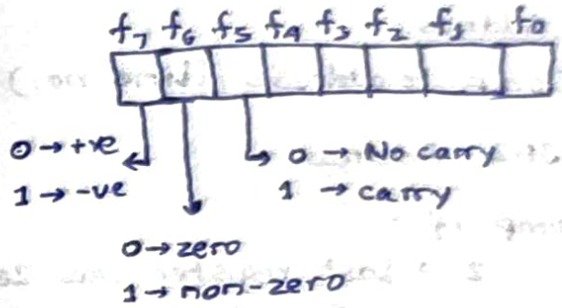
```
MOV R0, m
MOV R1, x
MOV R2, c
MUL R0, R1, R3
ADD R3, R2, R4
MOV y, R4
HLT
```

↳ Halt the program

Register: Temperory storage of 8 bits.

ex. $R_0, R_1, R_2, R_3, R_4, \cdots, R_n$

→ Flag register shows status of GPR using 8 bits.



(i) A = B

ASM

```
MOV R0, A
MOV R1, B
SUB R0, R1, R2
MOV C, R2
```

C → f7 f6 = 0

(ii) A < B

ASM

```
MOV R0, A
MOV R1, B
SUB R0, R1, R2
MOV C, R2
```

C → f7 f6 = 1 1

(iii) A > B

ASM

```
MOV R0, A
MOV R1, B
SUB R0, R1, R2
MOV C, R2
```

C → f7 f6 = 0 1

LSB : Least Significant Bit

MSB : Most Significant Bit

Finding if "n" is odd or even

ASM

```
000   MOV R0, A
001   MOV R1, R0[0]      <- Access LSB
010   JZ 101
011   OUT "odd"
100   JUMP 110
101   OUT "Even"
110   HLT
```

JUMP Address

Jump to address (line no.)

JZ / JNZ Address

Jump if :

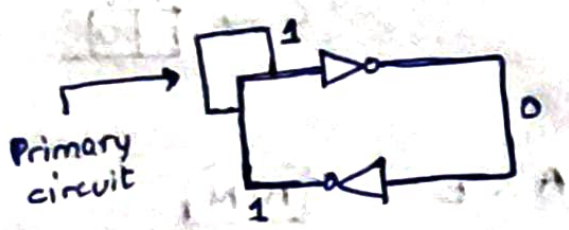
Z → Last register was zero

NZ → " " was non-zero

to address.

JC / JNC → Jump if carry / no-carry

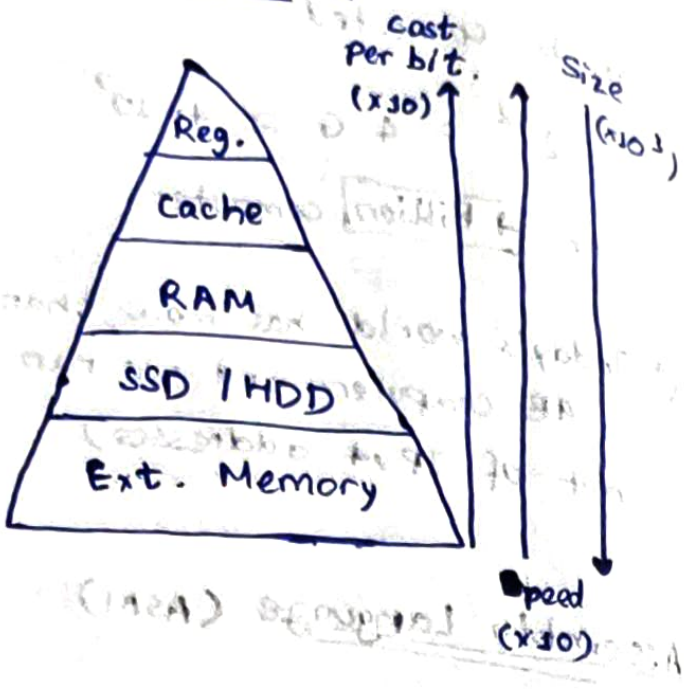
Memory Cell



A FLIP FLOP
(Two cross couple)

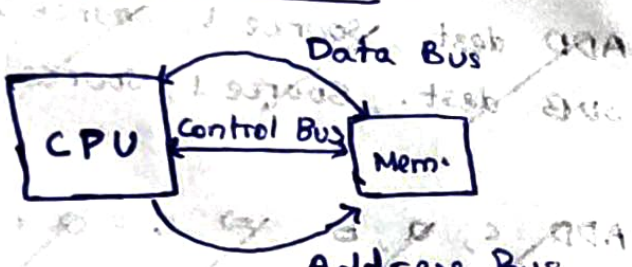
→ This system forms a loop and stores a value.

Memory Hierarchy



Instructions in ASM

Reading Instructions



→ CPU : Central Processing unit

Processes data fetched from memory. Also, it can store data in memory.

→ Memory : Data storage

→ Data Bus (D. B)

Bus b/w CPU and memory to transport data

→ Control Bus (C. B.)

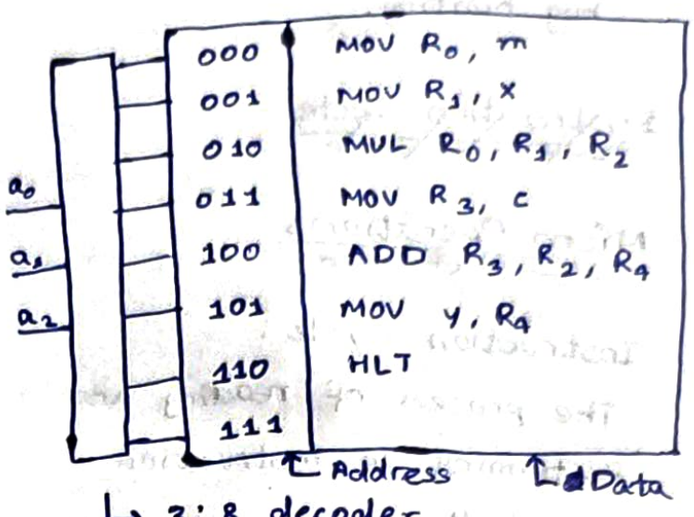
CPU chooses whether to read or write memory.

→ Address Bus (A. B.)

CPU tells which address to store or read from.

On the inside:

consider this code (set of instructions)



↳ 3:8 decoder

→ The address bus passes the address to be read/written using the signals a_0, a_1, a_2 which is sent to a decoder.

→ The decoder, using a circuit decides which address to enable based on the signals.

→ The control bus tells whether to read or write data.

↳ Read: Enabled data is transfered to data bus.

↳ Write: Data that comes through data bus gets stored.

→ Here, we have only 3 inputs and 8 addresses.
In reality, there is 64 bit rep meaning a total of 2^{64} addresses!

Hence, $2^{64} \approx \boxed{1.6 \times 10^{19}}$ data can be stored!

ASM - ROTATE and SHIFT

→ ROTATE: Each bit moves 1 pos. away, the last bit goes first.

ROTATE LEFT	ROTATE RIGHT
0110	0110
4 times: 1100	4 times: 0011
1001	1001
0011	1100
0110	0110

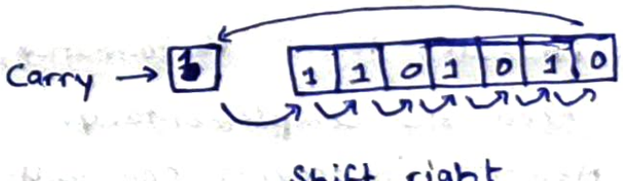
After n rotates (n = no. of bits) the same number will be obtained.

→ SHIFT: Each bit shifts by 1 pos., and a 0 fills the new empty space.

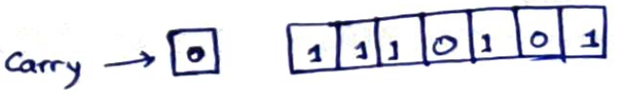
SHIFT LEFT	SHIFT RIGHT
0110 ← 0	0 → 0110 ← 6
6	
1100	0011 ← 3
12	
Multiply by 2	Divide by 2

→ Rotate w/ carry!

Similar to rotate, but MSB is allocated for carry.



Shift right



Processor

→ Cost: Same processor costs differently depend depending on the scenario.

Personal < Work < Server
80 k 1L-2L 4L-5L

→ This is due to testing. More intense the testing, more expensive it is and more stable.

→ Cheaper ones undergo less testing while expensive ones more.

→ But expensive processors crash less often due to intense testing.

→ It is not possible to completely prevent crash because not every case is tested.
Testing every case takes a very long time (1000s of years).

Process of Manufacturing a Chip

→ Takes 6-18 mo.

→ User → Design → Layout
(who wants to make chip)
↓
Foundry
↓
Chip

★ Why is chip manufacturing difficult?

→ PPA wall: Power, Performance, Area are complementary. Meaning, improving one will hurt the others.

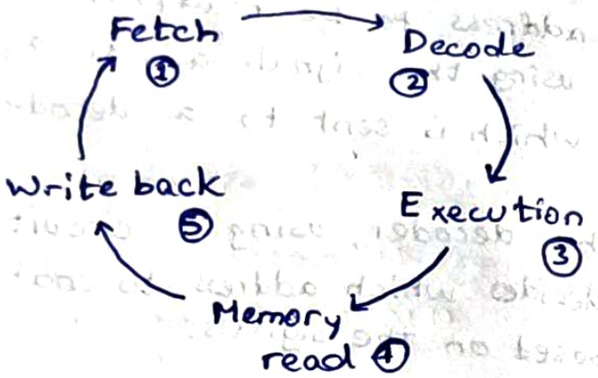
→ 60% of chip design is spent on verification and bug hunting.

Instruction Cycle

Micro Operations

Instruction Cycle:
The process of reading and performing an instruction repeatedly.

How it goes?



Steps 1-5 → 1 Instruction Cycle

Machine Cycle:
The steps executed in each part of the inst. cycle.

Inside Fetch

1. CU (Control Unit)
Pass address to MAR (Memory Address Register)
2. MAR → Computer Signal to memory
3. Pass data to MDR (Memory Data Register)
4. MDR → IR (Instruction Register)

Important Terms

→ MAR : Memory Address Register
(Holds address to be read in the memory)

→ MDR : Memory Data Register
(Holds data that has been read from memory)

→ CU : Control Unit
(Responsible for controlling registers, data entry/exit)

→ ALU : Arithmetic Logic Unit
(Does all arithmetic and logical operations)

→ IR : Instruction Register
(Holds the current executing instruction)

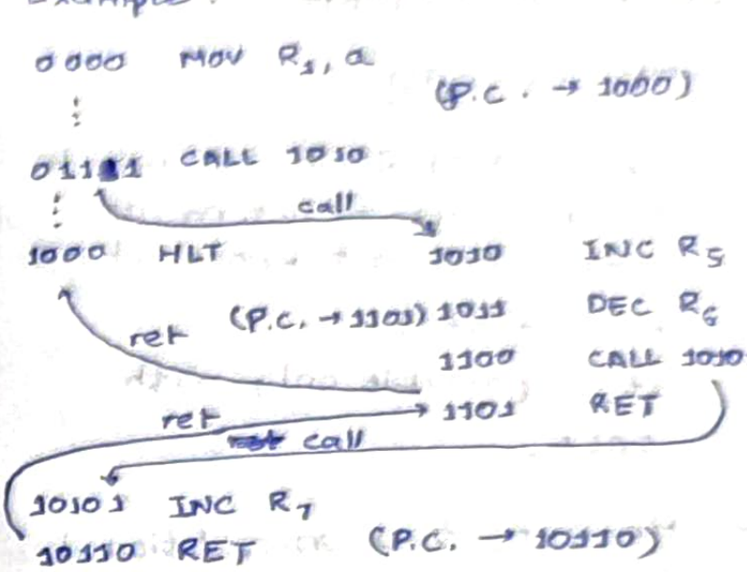
→ PC : Program Counter
(Holds the address of the next instruction)
(Part of the CPU, inc. by 1 when fetching current instruction)

It flushes and changes the address it holds when JUMP statements.

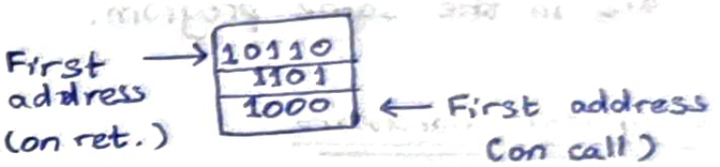
→ SP : Stack Pointer
(Used in call & return, especially to return to where call was executed)

Works with stack memory, which holds the P.C. value upon reading call. The S.P. is responsible for the machine to come back to where it left upon calling return.

Example :



Stack memory -



Upon call, new address (from P.C.) is pushed into the stack memory.

Upon return, the topmost address in the stack memory is popped.

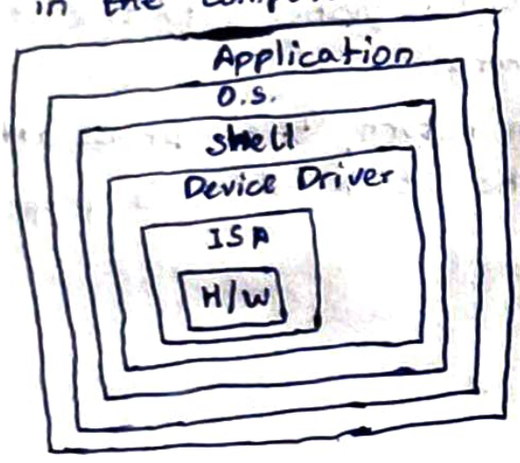
JUMP vs CALL

→ Upon using JUMP, it won't be possible to return as return does not know where to relocate.
It is possible only with CALL.

→ However, two unconditional JUMP statements can act as CALL-RET, considering both are in the same program.

Computer Hardware

→ First level (innermost) in the computer architecture.



~~Hard Disk~~

Hard Drive

As soon as the computer is turned off, all data is lost.

→ This is where hard drive is handy. It stores data permenantly, in a spinning disk covered w/ tiny magnets and a metal arm over it.

→ The arms moves to specific points on the disk where data gets stored.

→ However, speed of storing data in hard drive is no where near to that RAM. (RAM is much faster).

→ Hence, all data gets transf to RAM upon turning on th CPU.

Device Driver

→ A specialised S/W program that allows the O.S. to talk to the hardware devices and control them.

Multi-Core Processor

In a processor, there is 1 ~~CPU~~ CU and 1 ALU.

As we know, CU is responsible for controlling data entry/exit and ALU for operations.

Multiple CU:

Two CUs commanding 1 ALU is really inefficient because ALU can only follow one command at a time.

Multiple ALU:

Allows parallel execution, because the CU decodes the instruction and splits it into multiple micro-operations for ALU to process.

However, for sequential instructions (output of first instruction is the input for the second instruction)

Multiple CU, Multiple ALU:

Allows complete parallelism because there are separate CU/ALU pairs.

★ Can you perform multiple operations at once?

→ Depends. If there is only one bus, only one can be performed at a time. But, the architect becomes more complex in terms of design.

Addressing in ASM

→ [] refers to address.
[#] fetches data from the address #.

ex. ADD R_4, R_2, $[R_3]$
↑ fetches data from the address with value R_3

ADD $[R_1]$, R_2, R_3
↑ The sum of R_2 and R_3 is stored in the location pointed by $[R_1]$.

★ How many cores in a processor is most efficient?

→ Consider 128 coins. Any operation takes 1 sec. i.e. count, add etc.

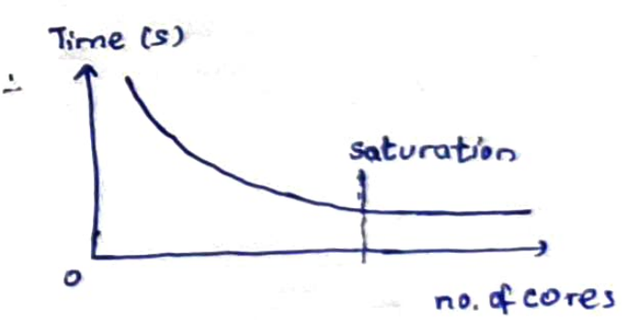
1 Person ⟶ 128 sec

2 Person ⟶ 64 64 → 64 + 1 = 65 sec

4 Person ⟶ 32 32 32 32 → 32 + 1 + 1 = 34 sec

8 Person ⟶ 16 16 16 16 16 16 16 16 → 16 + 1 + 1 + 1 = 19 sec

16 → 8 + 1 + 1 + 1 + 1 = 12 sec
32 → 4 + 5 × 1 = 9 sec
64 → 2 + 6 × 1 = 8 sec
128 → 1 + 7 × 1 = 8 sec
256 → Same as 128 (128 free, 128 count)



Hence, if there are less no. of processors, more time.
If there are more no. of processors, more expensive, but time saturates.

No. of processors for quickest and cost efficient processing:

$$F = \log_2(n)$$

where, n = no. of operations
F = no. of processors