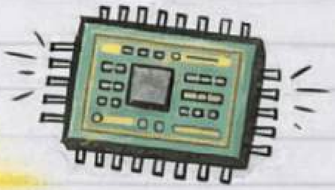
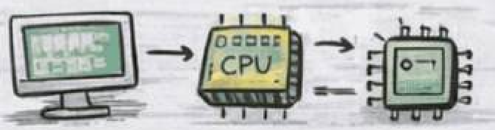


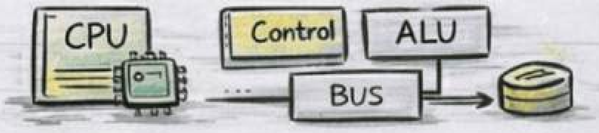
COA
Module-1 Notes

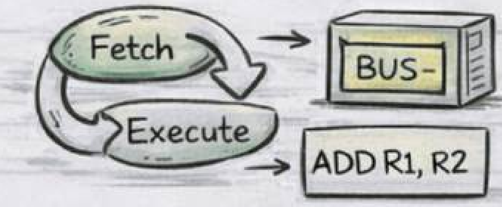
by pyqfort.com

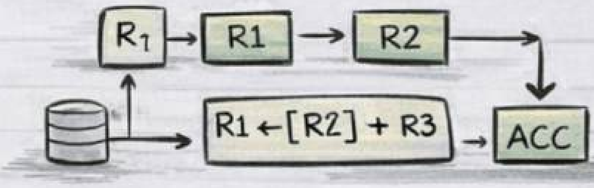
Contents Covered:

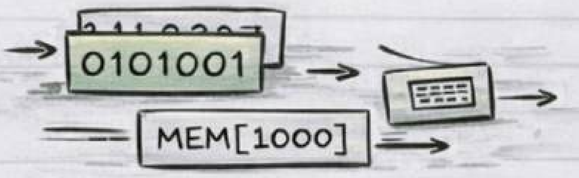
- Intro to COA
- CPU Components
- Registers
- Instruction Execution Cycle
- Register Transfer Language
- Data Transfer Instructions
- Microoperations
- Addressing Modes
- Instruction Set

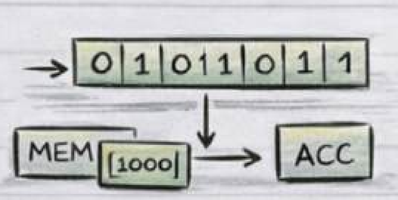




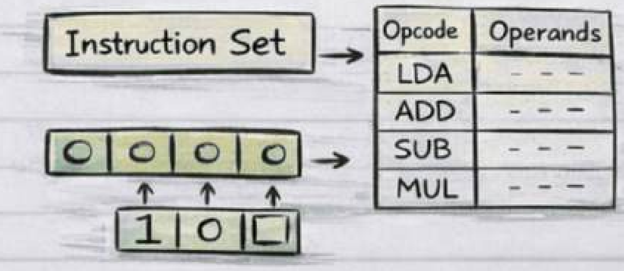








- Addressing Modes
- Instruction Set



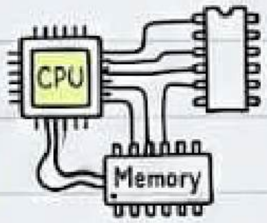
Opcode	Operands
LDA	- - -
ADD	- - -
SUB	- - -
MUL	- - -

Module 1: Computer Organization & Architecture Intro

1. Computer Architecture

- Refers to attributes visible to the programmer
- Examples: Instruction set, number of bits for data types, I/O mechanisms, addressing techniques

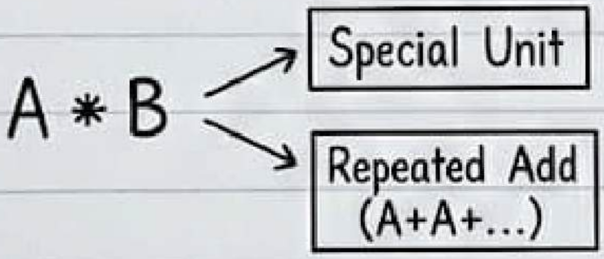
2. Computer Organization



- Refers to operational units and their interconnections that realize the architectural specifications
- Examples: Control signals, interfaces, memory technology

Key Difference & Example

- Organization is about how features are implemented (hardware details), while architecture is about the programmer's view
- Example: Architecture defines a 'multiply' instruction. Organization decides how it's done (e.g., special multiply unit or repeated addition)

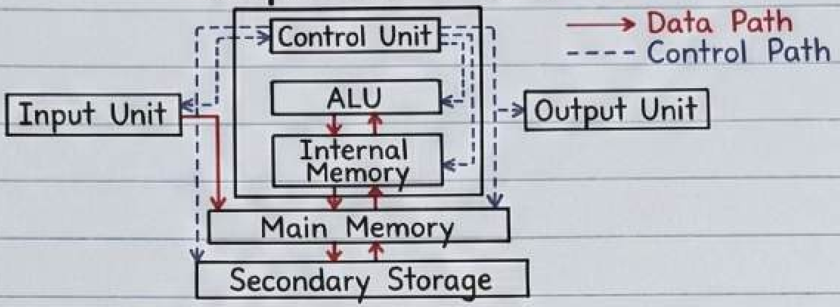


Functional Blocks & CPU Components

Central Processing Unit (CPU)

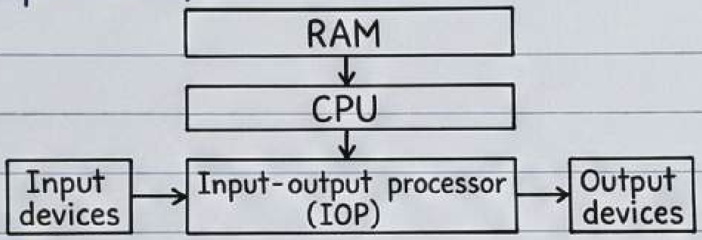
Often called the "brain", it performs thinking, calculating, and decision-making. Executes instructions for all tasks.

Components of CPU



Control Unit (CU): Controls operations, follows instructions, manages data flow.

ALU (Arithmetic Logic Unit): Performs math (add, subtract) & logic (comparisons).



Input Unit: Gets data (keyboard, mouse) for processing.

Output Unit: Sends processed data (monitor, printer) to user.

Memory Types

Main Memory (RAM): Holds current data/instructions.

Secondary Storage: Stores data/programs not in use (HDD, SSD).

Internal Memory: Temporary storage (registers, cache).

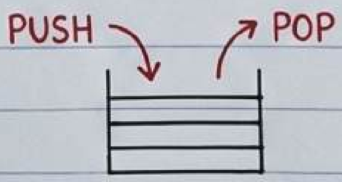
Stack Organization (LIFO)

Last-in, first-out list.

Insertion = PUSH (pushes new item on top).

Deletion = POP (removes top item).

Simulated by incrementing/decrementing the Stack Pointer (SP) register, which holds the address of the top item.



Registers

- The 'Instruction Register (IR)' is in the CPU & holds the 'machine instruction' currently being decoded/executed.

CPU
IR
DECODE

- Once fetched from memory, instruction is loaded into IR for the 'control unit'.

- The 'Memory Address Register (MAR)' holds the 'memory address' for read/write operations. It's like the CPU writing down a specific 'house number' to visit in memory.

- The 'Program Counter (PC)' keeps track of instructions & holds the 'address of the next instruction' to be executed.

- PC is 'incremented' after each fetch.

List of Registers for Basic Computer

Register symbol	Bits	Register name	Function
DR	16	Data register	Holds 'memory operand'
AR	12	Address register	Holds 'address for memory'
AC	16	Accumulator	'Processor register'
IR	16	Instruction register	Holds 'instruction code'
PC	12	Program counter	Holds 'address of next instruction'
TR	16	Temporary register	Holds 'temporary data'
INPR	8	Input register	Holds 'input character'
OUTR	8	Output register	Holds 'output character'

Instruction Execution Cycle

Phases

- Control unit goes through a cycle for each instruction.

- Divided into three major phases:

1. 'Fetch the instruction' from memory.

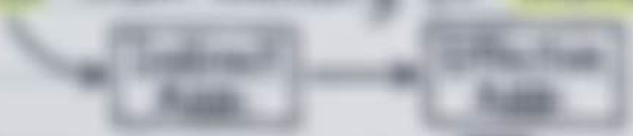
2. 'Decode the instruction'.

3. 'Execute the instruction'.

- Decoding determines 'operation', 'addressing mode', & 'location of operands'.

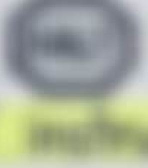
Detailed Cycle

- 1. 'Fetch' instruction from memory.
- 2. 'Decode' the instruction.
- 3. 'Read the effective address' from memory (if 'indirect address').

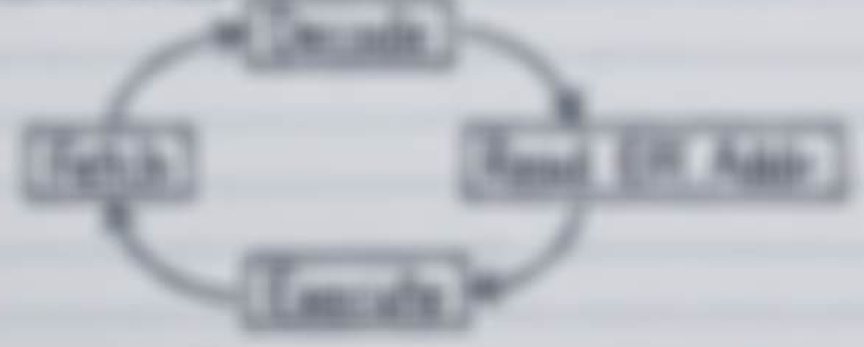


4. 'Execute' the instruction.

- Control goes back to step 1 after step 4.



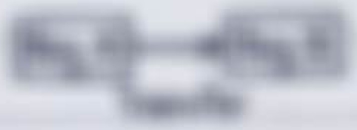
- Process continues indefinitely until a 'HALT instruction' is encountered.



RTL Basic & its Representation

- 'RTL' stands for 'Register Transfer Language'.

- A 'symbolic way' to describe operations on data in 'processor registers' & data 'transfers between registers', memory, or I/O.

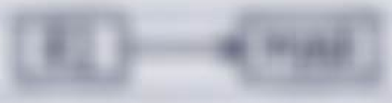


- A 'high-level notation' in 'computer architecture' to specify 'micro-operations' of a CPU.

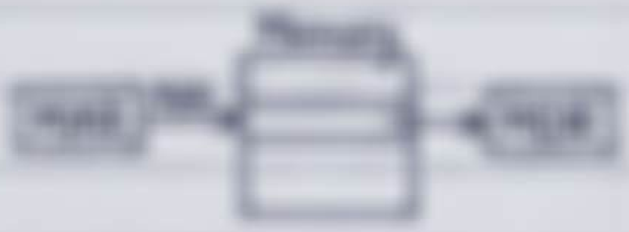
RTL Representation Example (Loading R1 from Memory)

Task: Load register 'R1' with data from a memory location whose address is in register 'R2'.

1. MAR ← R2



2. MDR ← M(MAR)

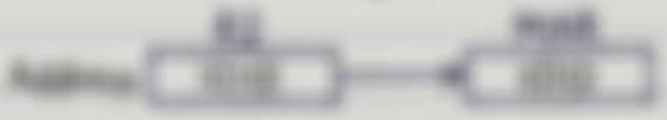


3. R1 ← MDR

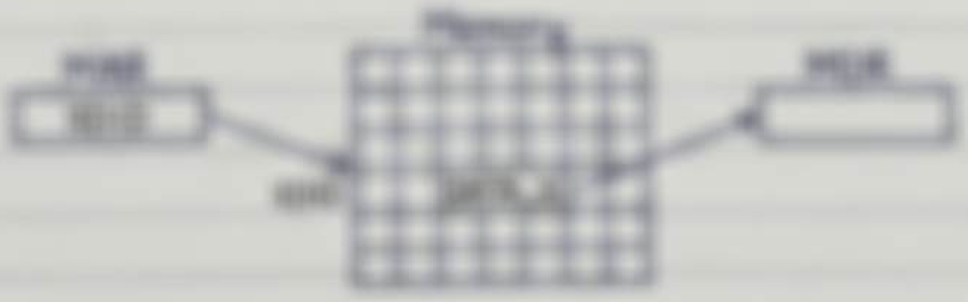
RTL Interpretation: Loading R1 from Memory

Task: Load register 'R1' with data from a memory location whose address is in register 'R2'.

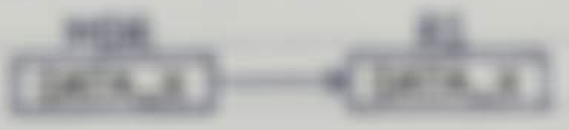
1. MAR ← R2
   - Interpretation: The content of register 'R2' (holds a 'memory address') is transferred to the 'Memory Address Register (MAR)'.
   - Meaning: The CPU is preparing to 'access memory' at the address specified by R2.



2. MDR ← M[MAR]
   - Interpretation: The 'data' from the memory location whose address is currently in 'MAR' is read from 'memory (M)' and transferred to the 'Memory Data Register (MDR)'.
   - Meaning: The 'data' from the desired memory location is 'fetched' and temporarily stored in MDR.



3. R1 ← MDR
   - Interpretation: The content of the 'Memory Data Register (MDR)' is transferred to register 'R1'.
   - Meaning: The 'fetched data' is now 'loaded' into the target register R1.



"Instructions"

- Data Transfer Instructions: Move data between registers, memory, and I/O devices.
  - Examples:
    - MOV (Move data from source to destination)
    - LOAD (Load data from memory into a register)
    - STORE (Store data from a register into memory)
    - PUSH (Push data onto a stack)
    - POP (Pop data from a stack)

- Arithmetic Instructions: Perform arithmetic operations like addition, subtraction, multiplication, division.
  - Examples:
    - ADD (Add) A + B = C
    - SUB (Subtract)
    - MUL (Multiply)
    - DIV (Divide)
    - INC (Increment) +1
    - DEC (Decrement) -1
- Logical Instructions: Perform bitwise logical operations like AND, OR, XOR, NOT.
  - Examples:
    - AND (Logical AND)
    - OR (Logical OR)

    - XOR (Logical Exclusive OR)
    - NOT (Logical NOT/Complement)
    - SHL (Shift Logical Left)
    - SHR (Shift Logical Right)

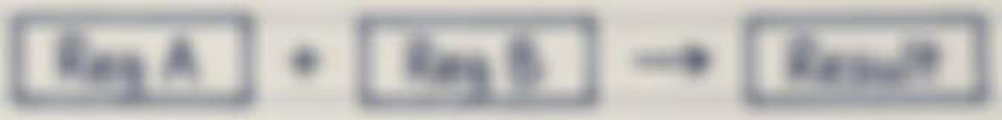
- Control Transfer (Branch/Jump) Instructions: Change the sequence of instruction execution.
  - Examples:
    - JMP (Unconditional Jump to a new address)
    - JZ (Jump if Zero flag is set)
    - JNZ (Jump if Not Zero flag is set)

- Input/Output (I/O) Instructions: Manage communication with I/O devices.
  - Examples:
    - IN (Input data from an I/O port)
    - OUT (Output data to an I/O port)

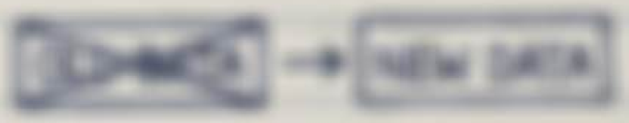
Microoperations

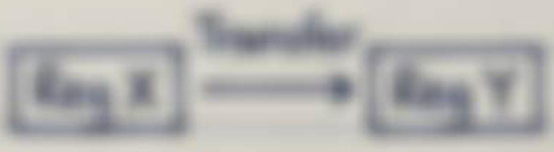
- Operations executed on data stored in 'registers' are called 'microoperations'.

- An 'elementary operation' performed on the information stored in one or more registers.



- The result may 'replace' previous info in a register or be 'transferred' to another.





- Examples:
  - 'shift' [← 0101 ←]
  - 'count' [5 (+1)]
  - 'clear'
  - 'load' → [DATA]

'Addressing Modes'

- 'Addressing modes' are different ways to tell the computer 'where to find the data' (operands) for an instruction.

- Why use them?
  - 'Flexibility': Access data in various ways.
  - 'Efficiency': Faster or use fewer bits.
  - 'Powerful Programming': Complex data structures & control flows.
  - 'Support for Data Structures': Pointers, arrays, etc.
  - 'Program Relocation': Code can run from different memory locations.
- Specific Modes:
  - 'Implied Mode': Operand location is 'inherent' in the instruction (e.g., a specific register). No address field needed.

  - 'Immediate Addressing': The actual 'operand (data)' is 'part of the instruction' itself.
    - Example: MOV AX, '1234H'

  - 'Register Addressing': Instruction specifies a 'CPU register' containing the operand.
    - Example: ADD BX, 'CX'

Reg CX
Instr.
Reg BX

  - 'Register Indirect Mode': Register contains the 'memory address' of the operand (a 'pointer').
    - Example: MOV AX, '[BX]'

Reg BX
Instr.
Memory

  - 'Direct Address Mode': Instruction contains the 'full memory address'.
    - Example: MOV AL, '[2000H]'

Instr.
Memory

  - 'Indirect Address Mode': Instruction has a memory address that holds 'another memory address' to the data (pointer to a pointer).
    - Example: MOV AX, '[[3000H]]'

Memory
Memory
Instr.

  - 'Relative Address Mode': Effective address = 'PC + offset' (displacement)'.
    - Example: JMP '+10'

PC
+
Offset
Effective Addr

  - 'Indexed Addressing Mode': Effective address = 'Index Register + Base Address'.
    - Example: MOV AX, '[SI + 2000H]'

Base Addr
+
Index Reg
Effective Addr

  - 'Base Register Addressing Mode': Effective address = 'Base Register + displacement'.
    - Example: MOV AX, '[BX + 10H]'

Instruction Set

An instruction set determines how machine language programs are constructed.

Opcode	Mode	Address

- "Opcode (Operation Code)": Specifies the operation to be performed (e.g., add, subtract).
- "Mode (Addressing Mode)": Used to locate the operands needed for the operation.
- Early computers had small & simple sets. Later, they became complex with a large number of instructions (>100-200) and addressing modes.

CISC (Complex Instruction Set Computer)

- "Large number of instructions" (typically 100-250).
- "Large variety of addressing modes" (typically 5-20).
- "Variable-length" instruction formats.
- Instructions manipulate operands in memory.

RISC (Reduced Instruction Set Computer)

- Aims to reduce execution time by simplifying the instruction set.
- "Relatively few instructions & addressing modes".
- "Memory access limited" to load and store instructions.
- All operations done within CPU registers.
- "Fixed-length", easily decoded format. "Single-cycle" execution.