Computer Architecture and Organization

Central Processing Unit (CPU)

- The Central Processing Unit (CPU) is the brain of the computer. Its primary function is to execute programs.
- A Program is a sequence of instruction to perform a specific task. Now the program that is to be executed is stored in the main memory.
- The CPU fetches instruction codes from the memory and decodes them.
- The necessary data for the execution of an instruction may be in Registers of the CPU or in the Memory.
- When the required data for the execution of the instruction is at hand, the CPU executes the instruction and gives the result, which is then stored in the memory or sent to the output device.
- Besides executing programs, the CPU also controls input devices, output devices and other components of the computer. Under its control programs, data are stored in the memory and displayed on the monitor.
- The CPU of a small computer is a microprocessor. For a large computer, number of microprocessors work together in parallel to execute a program.

CPU Organization

The following are the main sections of a CPU.

1. Arithmetic and Logic Unit (ALU)
2. Control Unit
3. Accumulators, General Purpose Registers and Special Purpose Registers.

Arithmetic and Logic Unit (ALU)

- The function of an ALU is to perform arithmetic and Logic operation. Usually ALU performs the following operations.
    1. Addition
    2. Subtraction
    3. Multiplication
    4. Division
    5. Logical AND, OR, Exclusive OR, NOT.
    6. Increment/ Decrement
    7. Left Shift/ Right Shift
    8. Clear (contents of Accumulator or Carry Flags)
- Other mathematical operations such as exponential, logarithmic, trigonometric and floating point operations are performed by special purpose math processor called Floating-Point Unit (FPU)
- Using Math processor, speed up program execution and reduce programming complexity.

Control Unit

- The Control Unit actually acts as the brain of the computer, and controls other devices such as input/output, memory etc.
- It fetches instruction from the memory, decodes the instruction, interprets the instruction to know what tasks are to be performed and sends suitable control signals to other components to perform further necessary steps to execute the instruction.
- It generates timing and control signals, and provides them for all operation.
- It controls the data flow between CPU and peripherals (including memory).
- It provides status, control and timing signals that the memory and I.O devices require.

Registers

- A CPU contains a number of registers to store data temporarily during execution of a program. The number of registers differs from processor to processor.

General Purpose Registers (GPRs) – These registers store data and intermediate results during the execution of a program. They can be accessed by users through instructions in Assembly Language Programming.

Accumulator – It is the most important general purpose register having multiple functions. It is most efficient in data movements, arithmetic and logical operations.

It holds one of the operands of arithmetic and logical operations, and after the execution of the arithmetic and logical instruction, the result is placed in the accumulator.

All data transfer between CPU and device/port are performed through the accumulator.

Special Purpose Registers – A CPU contains a number of special purpose registers for different purposes:

1. Program Counter (PC)
2. Stack Pointer (SP)
3. Status Register
4. Instruction Register (IR)
5. Index Register
6. Memory Address Register (MAR)
7. Memory Buffer Register (MBR) or Data Register (DR)

(1) **Program Counter (PC) –** The program counter keeps track of the address of the Instruction which is to be executed next.
   - It automatically gets incremented after an instruction has been fetched, or its address is modified for jump instruction.
   - Processors that employ Von Neuman Architecture contain a PC but that which use data flow architecture do not contain a PC.

(2) **Stack Pointer (SP) –** A stack is used to save the contents of a register if it is required during the execution of a program. The stack pointer holds the address of the last occupied memory location of the stack.

(3) **Status Register (or Flag Register)** – contains a number of Flags either to indicate certain condition arising after arithmetic and logical operation or to control certain operations.
   - A flag is a flip-flop used by the processor to indicate certain conditions or Set/Reset by the programmer to control certain operations.
   - There are two different types of flags: The control flags are used by the programmer and the condition flags are used by the Processor.
   - Intel 8085 processor contains 4 condition flags (carry flag, zero flag, sign flag, parity flag) and no control flags. Whereas Intel 8086 microprocessor contains 6 condition flags and 3 control flags.
   - The function of the different condition flags are as follows:
     - Carry Flag : indicates whether there is a carry or not after an arithmetic or logical operation.
     - Zero Flag : indicates whether the result of an arithmetic operation is Zero or non Zero.
     - Sign Flag : indicates whether the result is positive or negative.
     - Parity Flag : indicates whether the result contains odd number of 1s or even number of 1s.

(4) **Instruction Register (IR)** – It holds the instruction until it is decoded. Some microprocessors have two IRs and so they can fetch and save the next instruction while one is being executed.

(5) **Index Register** – Index registers are used for addressing. The address of an operand is the sum of the contents of the IR and a constant.

(6) **Memory Address Register (MAR)** – It holds the address of the instruction or data to be fetched from the memory. The CPU transfers the address of the next instruction from Program Counter (PC) to MAR. From MAR it is sent to memory through **Address Bus**.

(7) **Memory Buffer Register (MBR) or Data Register (DR)** – It holds the instruction code or data, received or sent to the memory. It is connected to the **Data Bus**.

Instructions

- An instruction is a command given to a computer to perform a specified operation on a given data.
- Each Instruction consists of two parts:
  - An Opcode (Operation Code) and
  - An Operand
- The first part of an instruction, known as opcode specifies the operation to be performed and the second part, called operand in the data on which computer performs the specified operation.

Intel 8085 Instructions

- Intel 8085 is a 8-bit Microprocessor, widely used in Laboratories for training purpose.
- It has a 8-bit Accumulator (A) and six 8-bit GPRs (B, C, D, E, H, L).
- Two 8-bit register pairs (B-C, D-E and H-L) are combined to handle 16-bit data.
- Special purpose registers are:
  - One 16-bit Program Counter (PC)
  - One 16-Bit Stack Pointer (SP)
  - One Instruction Register (IR)
- It does not have a Status Register, but contains a set of F/F to store status Flags.
- The combination of the binary bits, which indicate Status Flags is called Program Status Word (PSW). Here 5 bits shows status flags and 3 bits are undefined.

- PSW and Accumulator are treated as a 16-bit unit for stack operation.
- In addition to these registers, it also contains:
    - A temporary Register
    - Address Buffer Register and
    - Data Buffer Register

Some Instructions of 8085 Microprocessor:

1. **MOV r1, r2** : Contents of register r2 is transferred to register r1. (e.g. MOV A,B)
2. **MOV r, M** : Contents of memory location M whose address is in H-l pair are transferred to register r. (MOV A,M → transfers the contents in H-L pair to Accumulator.)
3. **MVI r, data** : The data specified in the instruction will be transferred to the register r. (MVI B,08→ will transfer 08 to register B.
4. **LXI rp, 16 bit data** : The 16-bit data specified in the instruction will be transferred to the register pair rp. (LXI H, 2500H → transfers 2500 to H-L pair).
5. **LDA addr** (load to A) : Contents of the memory address specified in the instruction are transferred to the Accumulator. (LDA 2200 → transfers the contents of the memory location 2200 to the Accumulator).
6. **STA addr** (store from A) : The contents of the Accumulator are transferred to the memory location addr. (STA 2000→ transfers the contents of the Accumulator to memory location 2000.
7. **ADD r** : The contents of the register r are added to the contents of the Accumulator, and the result is stored back to the Accumulator. (ADD C → will add the content of C to the Accumulator and the store the result back in the Accumulator).
8. **ADD M** : The contents of the memory location, whose address is in H-L pair are added to the Accumulator and the result is store there. (e.g. ADD M).
9. **ADI, data** : The data in the instruction are added to the contents of the Accumulator and the result is stored there. (ADI, 05 → will add 5 to the contents of the Accumulator).
10. **SUB r** : The content of the register r is subtracted from the Accumulator and result is stored back.
11. **RAL** : Contents of the Accumulator is rotated left one bit through carry.
12. **IN port addr** : Transfers data from the input device or input port to the accumulator.
13. **OUT port addr** : Transfers the contents of the accumulator to the output device/port.
14. **HLT** : After execution of this instruction, the Microprocessor halts.

Classification of Instruction Formats

A. Based on Instruction word size

- The binary codes for all instructions are not of the same length.
- These instructions and data are fed to the computer in binary format (0 and 1), known as machine language, but they are written in hexadecimal form, for the convenience of the user.
- Instructions are classified into the following three types according to their word length (i.e. length of the binary code).

(i) Single Byte instruction (Binary code of instruction one byte)

MOV A,B → (78 H in Hex form)
ADD B → (80 H)
RAL → (17 H)

(ii) Two Byte instruction (Machine code is of two bytes)

MVI A, 05 → (3E, 05) Note that 3E is the code for MVI A
IN 01 → (DB, 01)

(iii) Three Byte instruction

Here the first byte is the OPCODE and the 2nd and 3rd byte are either address or data.

LXI H 2500H → (21, 00, 25)
Loads H-L pair with 16-bit data 2500H
21 : opcode instruction for LXI H
00 : 8 LSBs of the data
25 : 8 MSBs of the data

LDA 2400H → (3A, 00, 24)
Loads accumulator with the contents of the memory location 2400H
3A : opcode for instruction LDA
00 : 8 LSBs of the data
24 : 8 MSBs of the data

B. Classification of Instruction based on number of operand address they contain:

(i) **0 – Address Instruction**
Do not contain any operand address

(ii) **1 – Address Instruction**
Only one operand address is specified in the instruction. The other operand address is implied, which is the Accumulator.

(iii) **2 - Address Instruction**
Here both the operand address is specified. The result is place in one of the specified address.

(iv) **3 – Address Instruction**
Two addresses are specified for two operand and one address for result.

Addressing Modes

- Each instruction needs data on which it has to perform specified operation.
- This operand (data) may be in the
    - Accumulator
    - General Purpose Registers
    - In some specified location in the memory.
- The techniques of specifying the address of the data are known as addressing modes.
- The important Addressing Modes are:

(i) Direct or Absolute Addressing:

- The address of the data is specified in the instruction itself.
- e.g. STA 2500H → (Stores the contents of the Accumulator in 2500H)
  LDA 2500H → (Loads A with the contents of 2500H)

(ii) Register Addressing:

- The operands are located in the registers (GPRs), i.e. the content of the register is the operand.

- e.g. MOV A, B (Transfers the contents of B to register A)

(The opcode for this instruction is 78H. In addition to the operation to be performed, the OPCODE also specified the address of the register mentioned in the Instruction.

78H, in Binary is 0111 1000, where 01→ denotes MOV, 111→ is the binary for register A, 000→ Binary for register B).

- ADD B (Adds the contents of register B to the contents of the Accumulator).

(The opcode for this instruction is 80H. The binary code is 1000 0000. The first 5 bits 10000→ specifies the operation to be performed. Last three bits 000 → is the binary code for register B).

(iii) Register Indirect Addressing:

- Here the address of the operand is given indirectly. The contents of the register or the register pair are the address of the operand.
- LXI H, 2400H (Load HL pair with 2400H. Move the content of the memory location whose address is in HL pair (2400H) to the accumulator)
- MOV A, M (In this case, the address of the memory location is not directly given in the instruction. The address of the memory location is stored in HL Pair which has been specified by earlier instruction "LXI H, 2400H").
- LXI H, 2200H (Loads HL Pair with 2200H)
- ADD M (Adds the contents whose address is in HL pair to the Accumulator).

(iv) Immediate Addressing

The operand is given in the instruction itself.

- MVI A, 06 (Move 06 to Accumulator)
- ADI 05 (Add 05 to the contents of A)
- LXI H, 2500H (Load HL Pair with 2500H)

(v) Implicit (or Implied) Addressing

These instructions operate only on one operand, which is the Accumulator.

- RAL (Rotate the contents of Accumulator left through carry).
- RLC (Rotate the contents of Accumulator left)
- CMA (take compliment of the content of A).

Some other addressing modes are:

vi. Indexed Addressing
vii. Based Addressing

viii. Based Index Addressing
ix. Relative Addressing
x. Relative Indexed Addressing
xi. Page Addressing
xii. Stack Addressing

Intel 8085 uses addressing modes only from (i) to (v).

Definitions of certain terms related to Addressing Modes:

- A large memory is divided into segments.
- The memory address of an operand (data) consists of two components:
    - The starting address of the segment and
    - An offset within the segment
- The starting address of the segment is supplied by the processor.
- The offset is determined by adding any combination of three offset address elements:
    - Displacement
    - Base
    - Index
- The combination depends on the addressing mode of the instruction to be executed.
- The offset is also called effective address.
- The memory address of an operand = Starting address of the segment + offset
- Displacement : A 8-bit or 16-bit immediate value given in the instruction
  Base : Contents of the Base register
  Index : Content of the index register

(vi) Indexed Addressing

The operand's offset is determined by adding an 8-bit or 16-bit displacement (given in the instruction) to the contents of the Index register.

(vii) Based Addressing

In this case, the operand's offset is the sum of the contents of the base register + 8-bit/16-bit displacement given in the instruction.

(viii) Based-Index Addressing

Here the contents of the Base register and the contents of the index register are added together to form the effective address. The base register contains a base address and the index register contains an index.

(ix) Relative Addressing

- Here a signed displacement is added to the current value of the Program Code (PC) to form the effective address.
- This mode of addressing is commonly used in branch (or jump) instructions.
- This is also known as PC relative addressing.
- The effective address specified memory location in relation to the current value of the PC.

(x) Relative Indexed Addressing

In this mode of addressing, the contents of the PC and the contents of the index register are added together to form the effective address.

(xi) Page Addressing

- In Paged mode of addressing, the memory is divided into a number of equal length pages.
- The page size is 256 bytes for 8 bit µP and 4 KB for 16-bit µP.
- The µP contains a page register to hold the page number.
- The instruction contains an offset. This offset indicates the address within the page w.r.t. the starting address of the page.
- The advantage of this mode of addressing is that a fewer bits in the instruction are required to indicate the memory address. The result is shorter instruction and faster execution.

(xii) Stack Addressing

In this case, the address of the operand is specified by the Stack Pointer (SP). The contents of the SP are automatically incremented/ decremented after a PUSH/POP.

Interrupts and Exceptions

- When data are ready, an I/O device can interrupt CPU. After completing the current instruction at hand, the CPU attends the I/O device.
- The CPU enters into a subroutine known as Interrupts Service Sub-routine (ISS) to transfer data from the device.
- Each CPU has interrupt lines through which I/O devices can be connected to the CPU.
- When data transfer is over, the CPU returns to the program it was executing.
- An interrupt caused by an external signal applied to an interrupt input line of a CPU is known as hardware interrupt.
- The normal program execution of a µP can also be interrupted by a special instruction in the program. This is known as software interrupt.
- The internal events, which cause the processor to go out of its normal processing sequence, are called Exception.
- The external events caused by external I/O devices, which prevent the further processing are called Interrupts. It handles external asynchronous events.
- Exceptions handle internal abnormal or unusual conditions, which prevent further processing. The processor treats S/W interrupts as exception.

Instruction Cycle

- A program is a sequence of well-defined instructions. Both inputs (i.e. data and the operation to be performed on the data), to the program are stored in the memory.
- The main job of the computer system is to execute these instructions.
- Few instructions in Assembly Language are:

- ADD – Performs addition
- SUB – Performs subtraction
- MUL – Performs multiplication
- MOV – Moves the contents from one location to another
- DIV – Performs division
- LDA – Loads the contents of variable
- JMP – Jumps to an instruction
- ABS – Calculate absolute value

- The instruction cycle is the sequence of events that takes place as the instruction is read from the memory and executed.
- A simple instruction cycle consists of the following steps:
  (i) **Fetch Cycle** – Fetching the instruction from the memory.
  (ii) **Decode Cycle** – Decoding the instruction
  (iii) **Execute Cycle** – Executing the instruction
  (iv) **Store Cycle** – Storing the result back to the memory

1. The Fetch Cycle:

During this cycle, the instruction which is to be executed next, is fetched from the memory to the processor. The steps performed during the fetch cycle are as follows:

i. The Program Counter (PC) keeps track of the memory location of the next instruction.
ii. This address is transferred from PC to MAR.
iii. The instruction is read from the memory.
iv. The instruction thus obtained is stored in the MBR → IR and the PC gets incremented by 1.
v. In the IR, the unique bit patterns that make up the machine language are extracted and sent to the decoder.

2. The Decode Cycle

The decode cycle is responsible for recognizing the operation that the bit pattern represents and activating the correct circuitry to perform the operation.

The steps are:

i. The OPCODE of the instruction is first read and then interpreted in the machine language.
ii. The operand is then transferred to the DR.

3. The Execute Cycle

Once the instruction has been decoded, the operation specified by the OPCODE is performed on user-provided data in ALU.

The following steps are involved:

i. The data is fetched into ALU from the memory location pointed by MBR.
ii. The operation specified by the decoded op-code is performed on the data in ALU.

4. Store Cycle

After the fetch, decode and execute cycles been executed, the results are ready to be stored. The steps involved are:

i. The results from the execution cycle are stored in the MBR.
ii. Then the results from MBR are stored back to the memory.

Instruction Set

- The processors are built with the ability to execute a limited set of basic operation.
- The collection of these operations is known as the processors Instruction Set.
- The instruction set is hardwired (embedded) in the processor, which determines the machine language for the processor.
- The more complicated the Instruction Set, the slower the processor works.
- Since each processor has its unique Instruction Set, machine language programs written for one processor will normally not run on a different processor.
- Therefore, all O/S and S/W programs are constructed within the boundaries of the processor's Instruction Set.
- Thus the design of the Instruction Set for a processor is very important.
- Based upon the Instruction Sets, the two common type of architectures are
  (i) Complex Instruction Set Computer (CISC)
  (ii) Reduced Instruction Set Computer (RISC)

CISC Architecture

- Earlier programming was done in low level languages such as machine language and assembly languages. Later on programmers started using high level languages. Compilers were used to convert these programs to equivalent low-level languages.
- To make compiler development easier, Complex Instruction Set Computer (CISC) was developed.
- These CISC-based processors shifted most of the burden of generating machine instructions to the processor.
- In fact the first PC microprocessors were CISC processors because all the instructions that the processor could execute were built into the processors.

RISC Architecture

- Reduced Instruction Set Computer (RISC) is a processor architecture that utilizes a small highly optimized set of instruction.
- RISC architecture simplifies the instruction set of the processor, which helps in reducing the execution time.
- Optimization of each instruction in the processor is done through a technique known as **pipelining**.

- Pipelining allows the processor to work on different steps of the instruction at the same time.
- Thus RISC processors are smaller, consume less power and run cooler than CISC processors.

Comparisons between RISC and CISC Architectures

CISC Architecture	RISC Architecture
• Complex instructions	• Simple instructions
• Smaller program code size	• Lengthier Program code size
• Increased Processor size	• Reduced processor size
• Less memory intensive	• More memory intensive
• Consumes more power	• Consumes less power
• Generates more heat	• Generates less heat

Processor Speed

- The overall speed of a computer system is determined by several factors such as the clock speed of the processor and the speed and size of the data bus.
- The clock speed is the rate at which the processor process information and this is measured in millions of cycles per second (mega hertz)