RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Computer System Organisation**

Subject Code: **EC-504**

Semester: **5th**

Dept. of Electronics and Communication Engineering

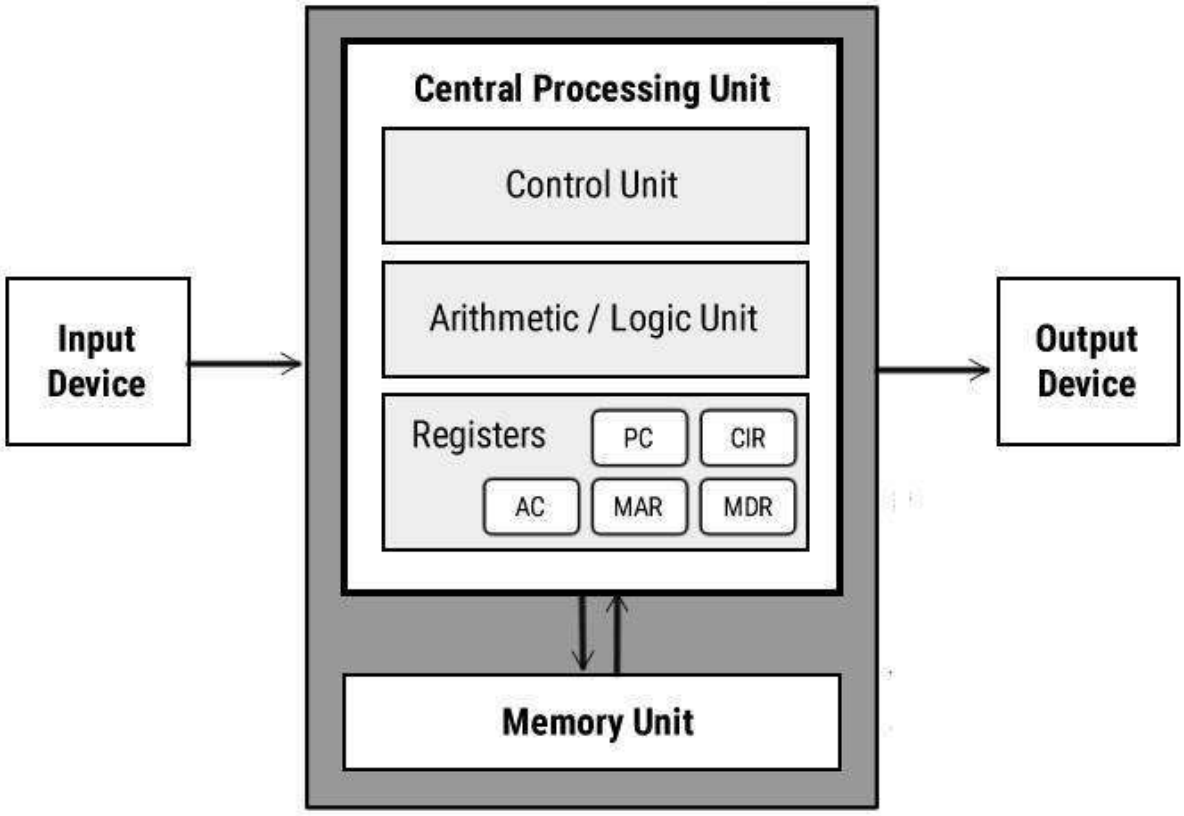
Class Notes EC V Sem CSO (Elective-I)

Unit 1

Computer Basics and CPU

(1).Von Neumann Architecture & its subsystems

Von Neumann architecture was first published by John von Neumann in 1945.His computer architecture design consists of a Control Unit, Arithmetic and Logic Unit (ALU), Memory Unit, Registers and Inputs/Outputs. Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory. This design is still used in most computers manufactured today.



Central Processing Unit (CPU)

The Central Processing Unit (CPU) is the electronic circuit responsible for executing the instructions of a computer program. It is sometimes referred to as the microprocessor or processor. The CPU contains the ALU, CU and a variety of registers.

Registers

Registers are high speed storage areas in the CPU. All data must be stored in a register before it can be processed.

MAR	MemoryAddress Register	Holds the memory location of data that needs to be accessed
MDR	Memory Data Register	Holds data that is being transferred to or from memory
AC	Accumulator	Where intermediate arithmetic and logic results are stored
PC	Program Counter	Contains the address of the next instruction to be executed
CIR	Current Instruction Register	Contains the current instruction during processing

Arithmetic and Logic Unit (ALU) :The ALU allows arithmetic (add, subtract etc) and logic (AND, OR, NOT etc) operations to be carried out.

Control Unit (CU) :The control unit controls the operation of the computer's ALU, memory and input/output devices, telling them how to respond to the program instructions it has just read and interpreted from the memory unit. The control unit also provides the timing and control signals required by other computer components.

Memory Unit : The memory unit consists of RAM, sometimes referred to as primary or main memory. Unlike a hard drive (secondary memory), this memory is fast and also directly accessible by the CPU.RAM is split into partitions. Each partition consists of an address and its contents (both in binary form).The address will uniquely identify every location in the memory.Loading data from permanent memory (hard drive), into the faster and directly accessible temporary memory (RAM), allows the CPU to operate much quicker.

(2) System Buses

Buses are the means by which data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory.A standard CPU system bus is comprised of a control bus, data bus and address bus.

Address Bus	Carries the addresses of data (but not the data) between the processor and memory
Data Bus	Carries data between the processor, the memory unit and the input/output devices
Control Bus	Carries control signals/commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer

(3) Micro-Operations

The operations executed on data stored in registers are called micro-operations. A micro-operation is an elementary operation performed on the information stored in one or more registers.

Example: Shift, count, clear and load.

Types of Micro-Operations

The micro-operations in digital computers are of 4 types:

1. Register transfer micro-operations transfer binary information from one register to another.
2. Arithmetic micro-operations perform arithmetic operations on numeric data stored in registers.
3. Logic micro-operations perform bit manipulation operation on non-numeric data stored in registers.
4. Shift micro-operations perform shift micro-operations performed on data.

Arithmetic Micro-Operations

Some of the basic micro-operations are addition, subtraction, increment and decrement.

Add Micro-Operation
It is defined by the following statement:

$$R3 \rightarrow R1 + R2$$

The above statement instructs the data or contents of register R1 to be added to data or content of register R2 and the sum should be transferred to register R3.

Subtract Micro-Operation

Let us again take an example:

$$R3 \rightarrow R1 + R2' + 1$$

In subtract micro-operation, instead of using minus operator we take 1's compliment and add 1 to the register which gets subtracted, i.e R1 - R2 is equivalent to R3 → R1 + R2' + 1

Increment/Decrement Micro-Operation

Increment and decrement micro-operations are generally performed by adding and subtracting 1 to and from the register respectively.

$$R1 \rightarrow R1 + 1$$

$$R1 \rightarrow R1 - 1$$

Symbolic Designation	Description
R3 ← R1 + R2	Contents of R1+R2 transferred to R3.
R3 ← R1 - R2	Contents of R1-R2 transferred to R3.
R2 ← (R2)'	Compliment the contents of R2.
R2 ← (R2)' + 1	2's compliment the contents of R2.
R3 ← R1 + (R2)' + 1	R1 + the 2's compliment of R2 (subtraction).
R1 ← R1 + 1	Increment the contents of R1 by 1.
R1 ← R1 - 1	Decrement the contents of R1 by 1.

Logic Micro-Operations

These are binary micro-operations performed on the bits stored in the registers. These operations consider each bit separately and treat them as binary variables.

Let us consider the X-OR micro-operation with the contents of two registers R1 and R2.

P: R1 ← R1 X-OR R2

In the above statement we have also included a Control Function.

Assume that each register has 3 bits. Let the content of R1 be 010 and R2 be 100.

The X-OR micro-operation will be:

010 → R1

100 → R2

110 → R1 after P=1

Shift Micro-Operations

These are used for serial transfer of data. That means we can shift the contents of the register to the left or right. In the shift left operation the serial input transfers a bit to the right most position and in shift right operation the serial input transfers a bit to the left most position.

There are three types of shifts as follows:

a) Logical Shift

It transfers 0 through the serial input. The symbol "shl" is used for logical shift left and "shr" is used for logical shift right.

R1 ← she R1

R1 ← she R1

The register symbol must be same on both sides of arrows.

b) Circular Shift

This circulates or rotates the bits of register around the two ends without any loss of data or contents. In this, the serial output of the shift register is connected to its serial input. "cil" and "cir" is used for circular shift left and right respectively.

c) Arithmetic Shift

This shifts a signed binary number to left or right. An arithmetic shift left multiplies a signed binary number by 2 and shift left divides the number by 2. Arithmetic shift micro-operation leaves the sign bit unchanged because the signed number remains same when it is multiplied or divided by 2.

(4) Register Transfer Language(RTL)

The symbolic notation used to describe the micro-operation transfers amongst registers is called Register transfer language.

The term register transfer means the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register.

The word language is borrowed from programmers who apply this term to programming languages. This programming language is a procedure for writing symbols to specify a given computational process.

Following are some commonly used registers:

1. Accumulator: This is the most common register, used to store data taken out from the memory.
2. General Purpose Registers: This is used to store data intermediate results during program execution. It can be accessed via assembly programming.
3. Special Purpose Registers: Users do not access these registers. These registers are for Computer system,
   - MAR: Memory Address Register are those registers that holds the address for memory unit.
   - MBR: Memory Buffer Register stores instruction and data received from the memory and sent from the memory.
   - PC: Program Counter points to the next instruction to be executed.
   - IR: Instruction Register holds the instruction to be executed.

Register Transfer

Information transferred from one register to another is designated in symbolic form by means of replacement operator.

R2 ← R1 It denotes the transfer of the data from register R1 into R2.

Normally we want the transfer to occur only in predetermined control condition. This can be shown by following if-then statement: if (P=1) then (R2 ← R1). Here P is a control signal generated in the control section.

Control Function

A control function is a Boolean variable that is equal to 1 or 0. The control function is shown as:

P: R2 ← R1

The control condition is terminated with a colon. It shows that transfer operation can be executed only if P=1.

(5).Instruction Fetch, Decode and Execution Cycle

Each computer's CPU can have different cycles based on different instruction sets, but will be similar to the following cycle:

(a)Fetch the instruction: The next instruction is fetched from the memory address that is currently stored in the program counter and stored into the instruction register. At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

(b)Decode the instruction: During this cycle the encoded instruction present in the instruction register is interpreted by the decoder.

(c)Read the effective address: In the case of a memory instruction (direct or indirect) the execution phase will be during the next clock pulse. If the instruction has an indirect address, the effective address is read from main memory, and any required data is fetched from main memory to be processed and then placed into data registers (clock pulse: T_3). If the instruction is direct, nothing is done during this clock pulse. If this is an I/O instruction or a register instruction, the operation is performed during the clock pulse.

(d) Execute the instruction: The control unit of the CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register. If the ALU is involved, it sends a condition signal back to the CU. The result generated by the operation is stored in the main memory or sent to an output device. Based on the feedback from the ALU, the PC may be updated to a different address from which the next instruction will be fetched.

(6) Instruction Format of basic Computer

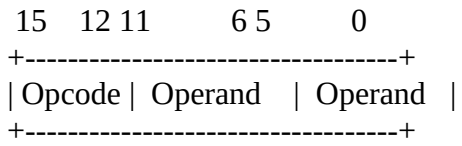
Computer instructions are the basic components of a machine language program. They are also known as macro operations, since each one is comprised of a sequences of micro operations.

Each instruction initiates a sequence of micro operations that fetch operands from registers or memory, possibly perform arithmetic, logic, or shift operations, and store results in registers or memory.

Instructions are encoded as binary instruction codes. Each instruction code contains of a operation code, or opcode, which designates the overall purpose of the instruction (e.g. add,

subtract, move, input, etc.). The number of bits allocated for the opcode determined how many different instructions the architecture supports.

In addition to the opcode, many instructions also contain one or more operands, which indicate where in registers or memory the data required for the operation is located. For example, and add instruction requires two operands, and a not instruction requires one.



The opcode and operands are most often encoded as unsigned binary numbers in order to minimize the number of bits used to store them. For example, a 4-bit opcode encoded as a binary number could represent up to 16 different operations.

(7) Addressing Modes of basic computer

The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

1. To give the programming versatility to the user.
2. To reduce the number of bits in addressing field of instruction.

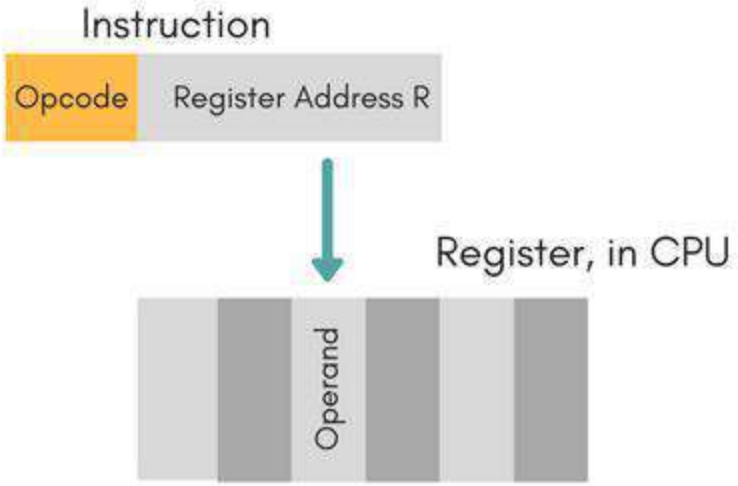
Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

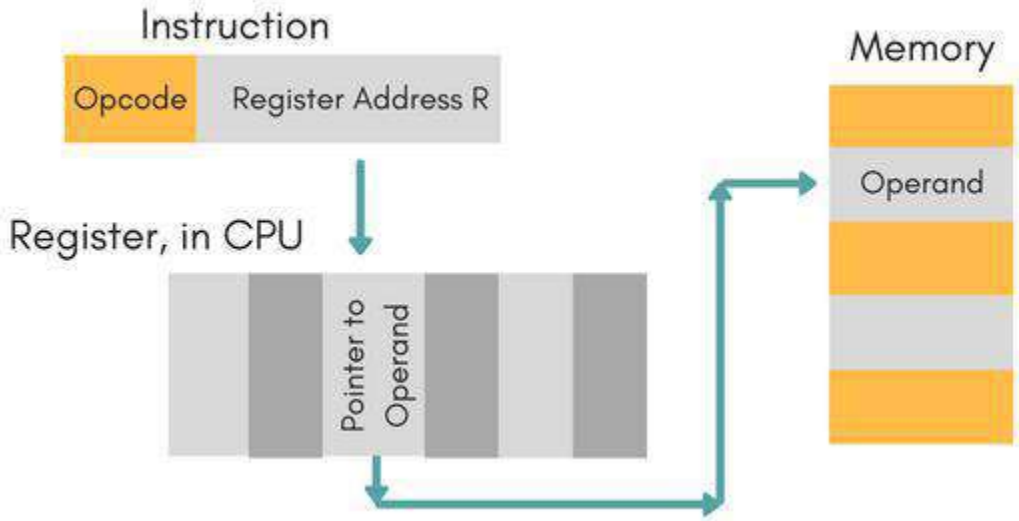
Register Mode

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.



Register Indirect Mode

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.

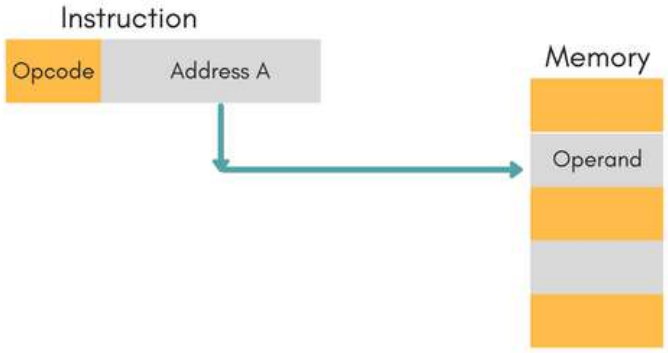


Auto Increment/Decrement Mode

In this the register is incremented or decremented after or before its value is used.

Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself.

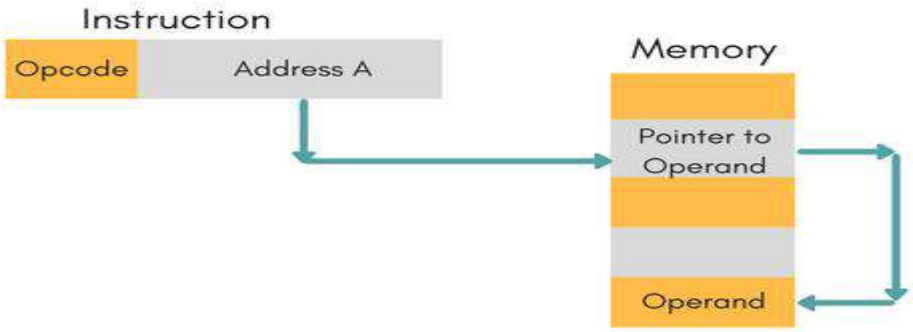


For Example: ADD R1,4000 - In this the 4000 is effective address of operand.

Effective Address is the location where operand is present.

Indirect Addressing Mode

In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the

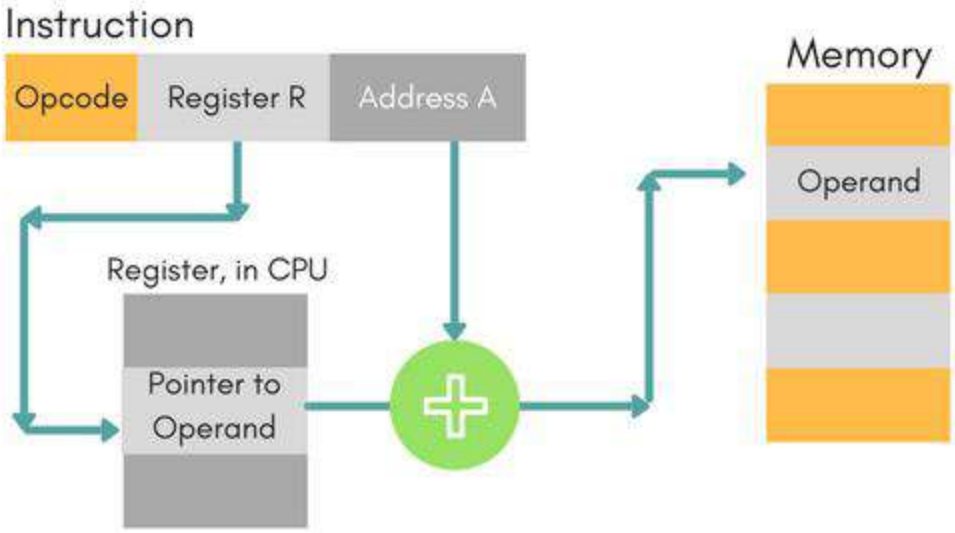


operand.

Displacement Addressing Mode

In this the contents of the indexed register is added to the Address part of the instruction, to obtain the effective address of operand.

EA=A+(R) In this the address field holds two values, A(which is the base value) and R(that holds the displacement), or vice versa.



RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in