

UNIT 01/LECTURE 01

Computer Basics and CPU

Von Neumann architecture & Von Neumann model (7 Marks)

The **Von Neumann architecture**, also known as the **Von Neumann model** and **Princeton architecture**, is a computer architecture based on that described in 1945 by the mathematician and physicist John von Neumann and others in the *First Draft of a Report on the EDVAC*.^[1] This describes a design architecture for an electronic digital computer with parts consisting of a processing unit containing an arithmetic logic unit and processor registers, a control unit containing an instruction register and program counter, a memory to store both data and instructions, external mass storage, and input and output mechanisms. The meaning has evolved to be any stored-program computer in which an instruction fetch and a data operation cannot occur at the same time because they share a common bus. This is referred to as the Von Neumann bottleneck and often limits the performance of the system.

The design of Von Neumann architecture is simpler than the more modern Harvard architecture which is also a stored-program system but has one dedicated set of address and data buses for reading data from and writing data to memory, and another set of address and data buses for fetching instructions.

A stored-program digital computer is one that keeps its program instructions, as well as its data, in read-write, random-access memory (RAM). Stored-program computers were advancement over the program-controlled computers of the 1940s, such as the Colossus and the ENIAC, which were programmed by setting switches and inserting patch leads to route data and to control signals between various functional units. In the vast majority of modern computers, the same memory is used for both data and program instructions, and the Von Neumann vs. Harvard distinction applies to the cache architecture, not the main memory.

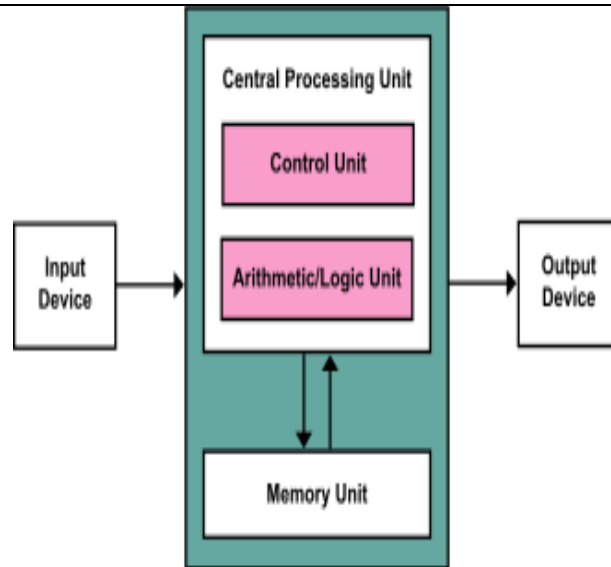


Fig.1.1 FOUR SUBSYSTEMS

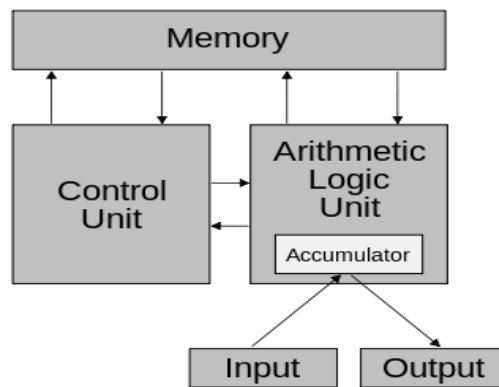


Fig.1.2 The model defines a computer as four subsystems: memory, arithmetic logic unit, control unit, and input/output.

- **Memory**

Memory is the storage area. It is where programs and data are stored during processing.

- **Arithmetic Logic Unit (ALU)**

The ALU is where calculation and logical operations take place.

- **Control Unit**

The control unit controls the operations of the memory, ALU, and the input/output subsystem.

- **Input/Output**

The input subsystem accepts input data and the program from outside the computer; the output subsystem sends the result of processing to the outside.

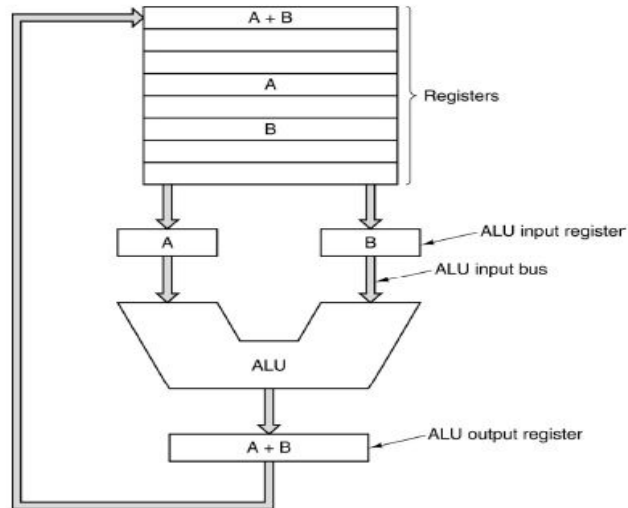


Fig.1.3 The data Path of a typical Von Neumann Machine

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Describe the Von Neumann Model And Explain the functioning of its component?	June 2011,June 2012,Dec 2013,June 2013	7
Q.2	Write the fetch and Execute cycle for the following instructions: AND & LDA	Dec 2013	7
Q.3	Explain different phases of an instruction cycle with flow diagram.	June 2009,Dec 2013	7
Q.4	What the function of the following in computer system (i) Accumulator (ii) Instruction Register (iii) Memory Address Register (iv) Program Counter	June ,Dec 2013,June 2009	7
Q.5	Draw the functional & Structural views of Computer System and explain in details	June 2012	6

Unit-01/Lecture-02

Central Processing Unit

A **central processing unit (CPU)** is the hardware within a computer that carries out the instructions of a computer program by performing the basic arithmetical, logical, control and input/output operations of the system. The term has been in use in the computer industry at least since the early 1960s.^[1] The form, design, and implementation of CPUs have changed over the course of their history, but their fundamental operation remains much the same.

A computer can have more than one CPU; this is called multiprocessing. All modern CPUs are microprocessors, meaning contained on a single chip. Some integrated circuits (ICs) can contain multiple CPUs on a single chip; those ICs are called multi-core processors. An IC containing a CPU can also contain memory, peripheral devices, and other components of a computer system; this is called a system on a chip (SoC).

Two typical components of a CPU are the arithmetic logic unit (ALU), which performs arithmetic and logical operations, and the control unit (CU), which extracts instructions from memory and decodes and executes them, calling on the ALU when necessary.

Not all computational systems rely on a central processing unit. An array processor or vector processor has multiple parallel computing elements, with no one unit considered the "center". In the distributed computing model, problems are solved by a distributed interconnected set of processors.

Memory

Memory refers to the physical devices used to store programs (sequences of instructions) or data (e.g. program state information) on a temporary or permanent basis for use in a computer or other digital electronic device. The term primary memory is used for the information in physical systems which function at high-speed (i.e. RAM), as a distinction from secondary memory, which are physical devices for program and data storage which are slow to access but offer higher memory capacity. If needed, primary memory can be stored in secondary memory, through a memory management technique called "virtual memory". An archaic synonym for memory is store.

The term "memory", meaning primary memory is often associated with addressable

semiconductor memory, i.e. integrated circuits consisting of silicon-based transistors, used for example as primary memory but also other purposes in computers and other digital electronic devices. There are two main types of semiconductor memory: volatile and non-volatile. Examples of non-volatile memory are flash memory (sometimes used as secondary, sometimes primary computer memory) and ROM/PROM/EPROM/EEPROM memory (used for firmware such as boot programs). Examples of volatile memory are primary memory (typically dynamic RAM, DRAM), and fast CPU cache memory (typically static RAM, SRAM, which is fast but energy-consuming and offer lower memory capacity per area unit than DRAM).

Most semiconductor memory is organized into memory cells or bi stable flip-flops, each storing one bit (0 or 1). Flash memory organization includes both one bit per memory cell and multiple bits per cell (called MLC, Multiple Level Cell). The memory cells are grouped into words of fixed word length, for example 1, 2, 4, 8, 16, 32, 64 or 128 bit. Each word can be accessed by a binary address of N bit, making it possible to store 2^N words in the memory. This implies that processor registers normally are not considered as memory, since they only store one word and do not include an addressing mechanism.

I/O bus: Any path used to transfer data and control information between components of an I/O subsystem. An I/O bus consists of wiring (either cable or backplane), connectors, and all associated electrical drivers, receivers, transducers, and other required electronic components. I/O buses are typically optimized for the transfer of data, and tend to support more restricted configurations than networks. Generally, an I/O bus that connects a host computer's host bus adapter to intelligent storage controllers or devices is called a host I/O bus. An I/O bus that connects storage controllers or host I/O bus adapters to devices is called a device I/O bus. cf. channel, device, device I/O bus, host I/O bus, network.

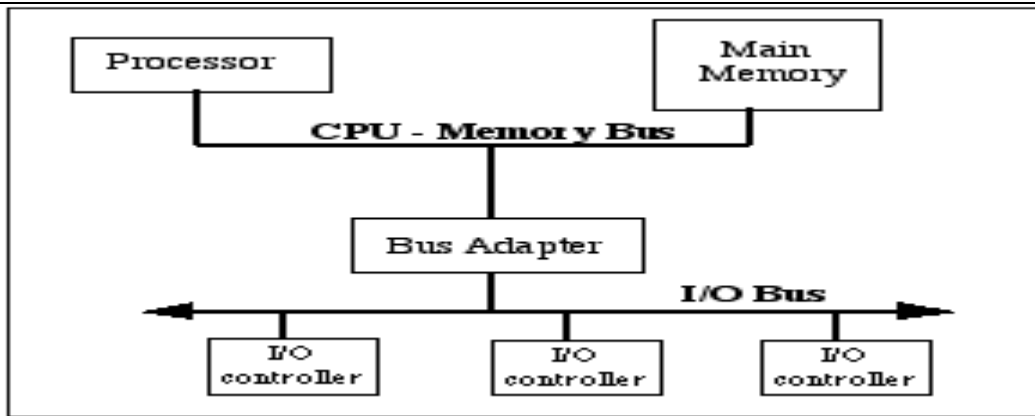


Fig 1.4

System bus

A system bus is a single computer bus that connects the major components of a computer system. The technique was developed to reduce costs and improve modularity. It combines the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation. Although popular in the 1970s and 1980s, modern computers use a variety of separate buses adapted to more specific needs.

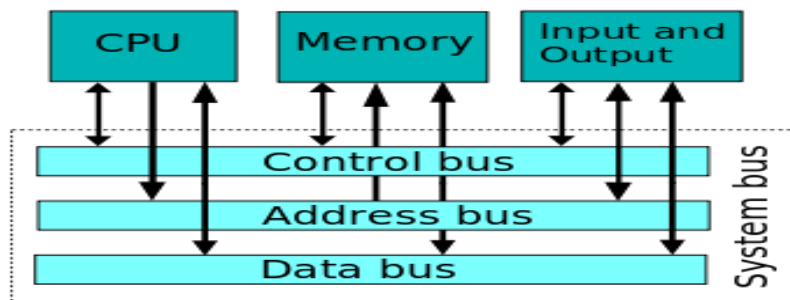


Fig 1.5 Example of Single system computer bus

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Explain the following terms: Instruction Register ,Memory Address Register ,Memory data Register	June 2009 ,D ec2013	7
Q.2	Explain the bus interconnection scheme	June 2009	6

Unit-01/Lecture-03

Memory Register

Registers are memories located within the Central Processing Unit (CPU). They are few in number (there are rarely more than 64 registers) and also small in size, typically a register is less than 64 bits in size.

The contents of a register can be “read” or “written” very quickly however, often an order of magnitude faster than main memory and several orders of magnitude faster than disk memory.

Different kinds of register are found within the CPU. General Purpose Registers are available for general use by the programmer. Unless the context implies otherwise we'll use the term "Register" to refer to a General Purpose Register within the CPU. Most modern CPU's have between 16 and 64 General Purpose Registers. Special Purpose Registers have special uses and are either nonprogrammable and internal to the CPU or accessed with special instructions by the programmer.

Examples of such registers include:

- Program Counter/Instruction Pointer Register (PC/IP)
- Instruction Register (IR)
- ALU Input & Output Registers
- Condition Code (Status/Flags) Register
- Stack Pointer Register (SP)

Although Register Size (the size of the Register bit-group) tends to vary according to register type, the Word Size of an Architecture is often (but not always!) defined by the Size of the General Purpose Registers.

In contrast to Main memory and disk memory, registers are “addressed” directly by specific instructions or by encoding a register number within a computer instruction. At the programming (assembly) language level of the CPU, registers are normally specified with special identifiers (e.g.

R0, R1, R7, SP, PC)

A final point, the contents of a register are lost if power to the CPU is turned off, so registers are unsuitable for holding long-term information or information that is needed for retention after a power-shutdown or failure. Registers are however, the fastest memories, and if exploited can result in programs that execute very quickly.

Main Memory (RAM)

If we were to sum all the bits of all registers within CPU, the total amount of memory probably would not exceed 5,000 bits. Most computational tasks undertaken by a computer require a lot more memory. Main memory is the next fastest memory within a Computer and is much larger in size.

Typical main memory capacities for different kinds of computers are:

Personal Computer	256MB
Fileserver	4GB
Database Mainframe	32GB

Computer Architectures also impose an architectural constraint on the maximum allowable RAM. This constraint is normally equal to 2^{WordSize} memory locations

RAM (Random Access Memory) is the most common form of Main Memory. RAM is normally located on the motherboard and so is typically less than 12 inches from the CPU. ROM (Read Only Memory) is like RAM except that its contents cannot be overwritten. ROM memory is often used to store the “boot” or start-up program that a computer executes when powered on.

Although slower than register memory, the contents of any location in RAM can still be “read” or “written” very quickly. The time to read or write is referred to as the access time and is the same for all RAM locations.

In contrast to register memory, RAM is used to hold both program code (instructions) and data

(numbers, strings etc). Programs that are running are normally “loaded” into RAM from a disk prior to execution by the CPU.

Locations in RAM are identified by an addressing scheme e.g. numbering the bytes in RAM from 0 onwards. The contents of RAM are lost if the power is turned off.

Disk Memory

Disk memory is used to hold programs and data over the longer term. The contents of a disk are NOT lost if the power is turned off. Disk capacities range from 2GB to over 40 GB (40×10^9). Disks are much slower than Register and Main memory, the access-time to data on disk is typically between 5 and 15 milliseconds (5×10^{-3} sec), although disks can typically transfer hundreds or thousands of bytes in one go.

Disks can be housed internally within a Computer “box” or externally. There are also many kinds of Disk unit, e.g: Magnetic Hard Disks, Floppy Disks (V. Slow), Magneto-Optical CD Roms/Disks, DVDs.

Disk locations are identified by special disk addressing schemes (e.g. track and sector numbers).

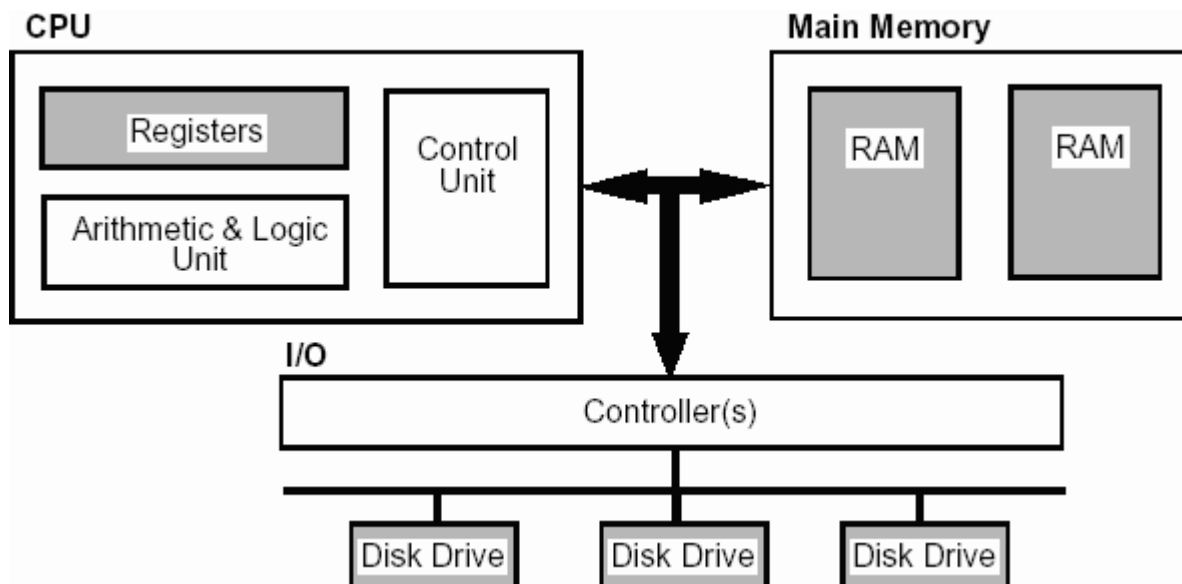


Fig 1.6 Registers

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Explain the following terms: Memory Address Register ,Memory data Register	June 2009 ,D ec2013	7

Unit-01/Lecture-04

Program counter (PC)

The program counter (PC), commonly called the instruction pointer (IP) in Intel x86 and Itanium microprocessors, and sometimes called the instruction address register (IAR), the instruction counter, or just part of the instruction sequencer,[3] is a processor register that indicates where a computer is in its program sequence.

In most processors, the PC is incremented after fetching an instruction, and holds the memory address of (“points to”) the next instruction that would be executed. (In a processor where the incrementation precedes the fetch, the PC points to the current instruction being executed.)

Instructions are usually fetched sequentially from memory, but control transfer instructions change the sequence by placing a new value in the PC. These include branches (sometimes called jumps), subroutine calls, and returns. A transfer that is conditional on the truth of some assertion lets the computer follow a different sequence under different conditions.

A branch provides that the next instruction is fetched from somewhere else in memory. A subroutine call not only branches but saves the preceding contents of the PC somewhere. A return retrieves the saved contents of the PC and places it back in the PC, resuming sequential execution with the instruction following the subroutine call.

Or

A program counter is a register in a computer processor that contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its stored value by 1. After each instruction is fetched, the program counter points to the next instruction in the sequence. When the computer restarts or is reset, the program counter normally reverts to 0.

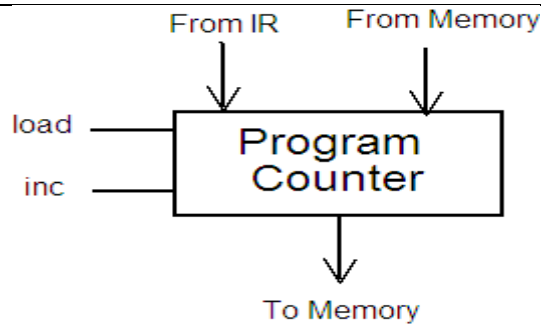


Fig 1.7 The Program Counter

Accumulator

In a computer's central processing unit (CPU), an accumulator is a register in which intermediate arithmetic and logic results are stored.

Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) to main memory, perhaps only to be read right back again for use in the next operation. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register. Early electronic computer systems were often split into two groups, those with accumulators and those without. Modern computer systems often have multiple general purpose registers that operate as accumulators, and the term is no longer as common as it once was. However, a number of special-purpose processors still use a single accumulator for their work, in order to simplify their design.

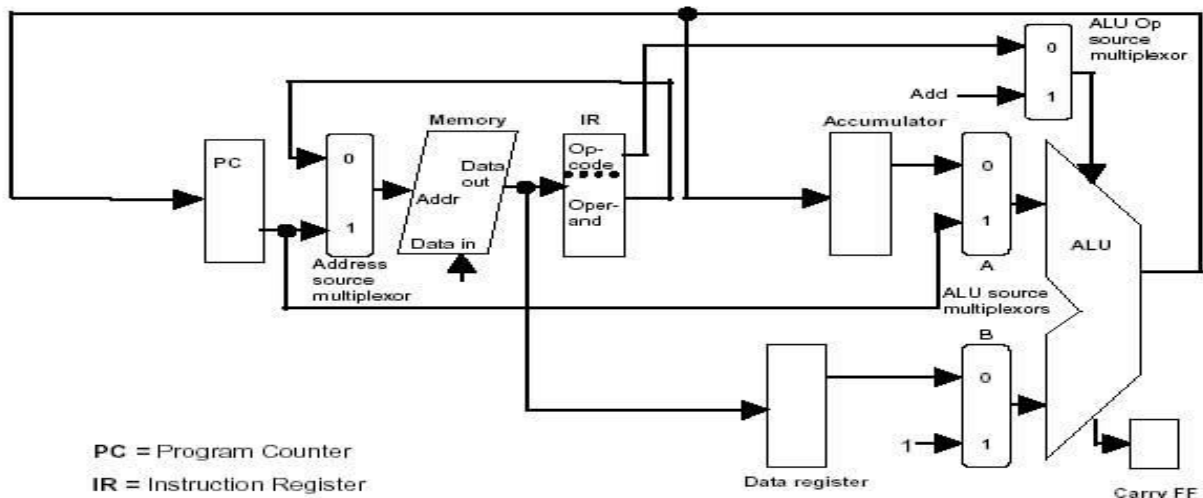


Fig 1.8 The Accumulator

Or

An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit). The term "accumulator" is rarely used in reference to contemporary CPUs, having been replaced around the turn of the millennium by the term "register." In modern computers, any register can function as an accumulator.

Instruction register (IR)

In computing, an instruction register (IR) is the part of a CPU's control unit that stores the instruction currently being executed or decoded.[1] In simple processors each instruction to be executed is loaded into the instruction register which holds it while it is decoded, prepared and ultimately executed, which can take several steps.

Some of the complicated processors use a pipeline of instruction registers where each stage of the pipeline does part of the decoding, preparation or execution and then passes it to the next stage for its step. Modern processors can even do some of the steps out of order as decoding on several instructions is done in parallel.

Decoding the op-code in the instruction register includes determining the instruction, determining where its operands are in memory, retrieving the operands from memory, allocating processor resources to execute the command (in super scalar processors), etc.

The output of IR is available to control circuits which generate the timing signals that control the various processing elements involved in executing the instruction.

In the Instruction cycle, the instruction is loaded into the Instruction register after the processor fetches it from the memory location pointed by the Program counter.

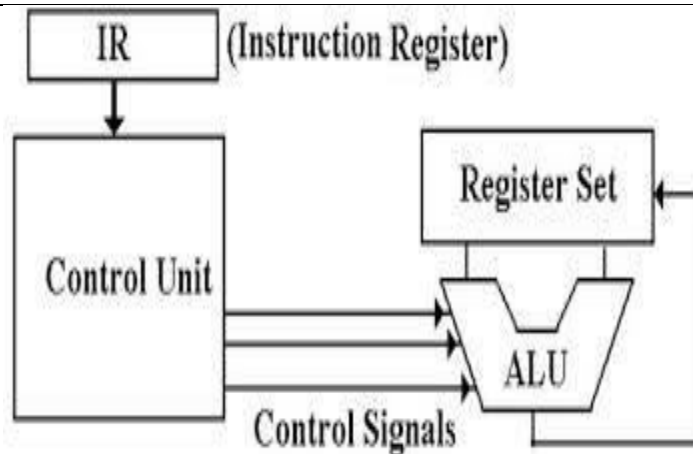


Fig 1.9 The Instruction Register

Micro Operations

Micro-operation generally involves a transfer between registers, transfer between registers and external bus, or a simple ALU operation. Each clock pulse defines a time unit, which are of equal duration. Micro-operations are performed within this time unit. If multiple micro-operations do not interfere with one another then grouping of micro-operations can be performed within one time unit. Grouping can be performed as long as, Proper sequence of events are followed:

PC → MAR must be done first in order for MEMORY → MDR

Conflicts are avoided

MEMORY → MDR can not be in the same time unit as MDR → IR

Four micro-operations:

Each micro-operation involves the movement of data into or out of a register.

The Fetch Cycle

$$\begin{aligned}
 t_1 &: PC \rightarrow MAR \\
 t_2 &: MEMORY \rightarrow MDR \\
 t_3 &: PC + 1 \rightarrow PC \\
 & \quad MDR \rightarrow IR
 \end{aligned}$$

Indirect Cycle

- Occurs if the instruction specifies an indirect address.
- Consists of three time unit and three micro-operations.
- Data is transferred to the MAR from the IR, which is used to fetch the address of the operand, the IR is then updated from MDR so it contains a direct address rather than indirect.

$t_1 : IR \rightarrow MAR$

$t_2 : MEMORY \rightarrow MDR$

$t_3 : MDR \rightarrow IR$

Interrupt Cycle

Occurs if any enabled interrupts have occurred at the completion of the execute cycle.

- The contents of the PC are transferred to the MDR, so that they can be saved for return from the interrupt.
- MAR is loaded with the address at which the contents of the PC are to be saved
- PC is loaded with the address at the start of the interrupt routine.
- Final step is to store the MDR into MEMORY.

$t_1 : PC \rightarrow MDR$

$t_2 : \text{Saved Address} \rightarrow MAR$

$\text{Routine Address} \rightarrow PC$

$t_3 : MDR \rightarrow MEMORY$

The Execute Cycle

- Execute cycle is not as predictable as other cycles (fetch, indirect, or interrupt).
- Number of time units and micro-operations varies for every execution cycle.

$t_1 : IR \rightarrow MAR$

$t_2 : MEMORY \rightarrow MDR$

$t_3 : MDR + R1 \rightarrow R1$

Instruction Cycle

- Each phase decomposed into sequence of elementary micro-operations (fetch, indirect, and interrupt cycles)
- Execute cycle
- One sequence of micro-operations for each opcode
- Need to tie sequences of micro-operations together

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Write the function of the following in computer system (i)Accumulator (ii) Instruction Register (iii) Program Counter (iii) Memory Address Register	June 2011,June 2012,Dec 2013,June 2013	7

UNIT 01/LECTURE 05

Register Transfer Language

- Digital systems are composed of modules that are constructed from digital components, such as registers, decoders, arithmetic elements, and control logic
- The modules are interconnected with common data and control paths to form a digital computer system
- 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
- Examples are shift, count, clear, and load
- Some of the digital components from before are registers that implement micro operations .

The internal hardware organization of a digital computer is best defined by specifying

- The set of registers it contains and their functions
- The sequence of micro operations performed on the binary information stored
- The control that initiates the sequence of micro operations

Use symbols, rather than words, to specify the sequence of micro operations

- The symbolic notation used is called a register transfer language
- A programming language is a procedure for writing symbols to specify a given computational process .
- Define symbols for various types of micro operations and describe associated hardware that can implement the micro operations.

Instruction cycle

An instruction cycle (sometimes called fetch-and-execute cycle, fetch-decode-execute cycle, or FDX) is the basic operation cycle of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction requires, and carries out those actions. This cycle is repeated continuously by the central processing unit (CPU), from boot up to when the computer is shut down.

In simpler CPUs, the instruction cycle is executed sequentially: each instruction is completely processed before the next one is started. In most modern CPUs, the instruction cycle is instead executed concurrently in parallel, as an instruction pipeline: the next instruction starts being processed before the previous instruction is finished, which is possible because the cycle is broken up into separate steps.

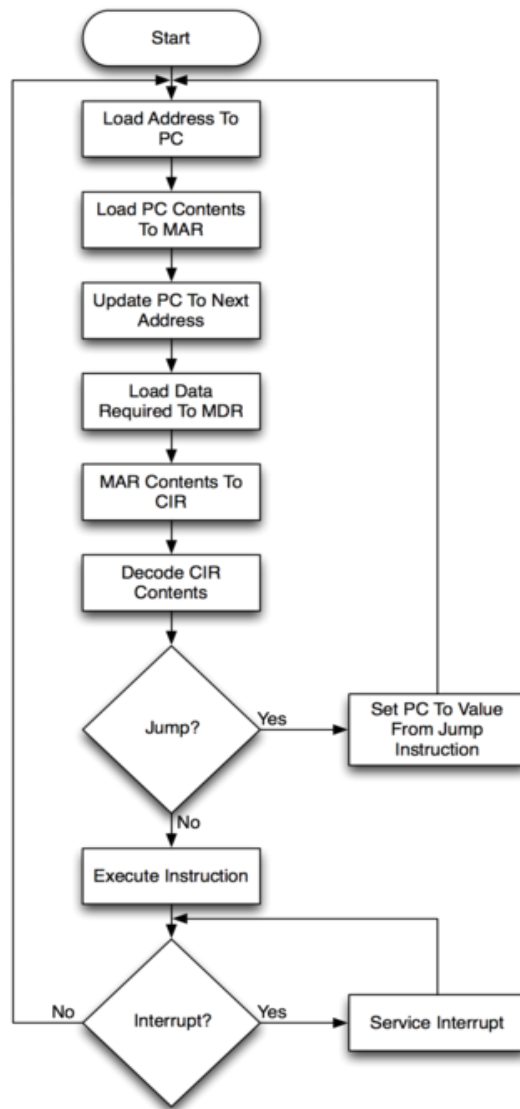


Fig 1.10 A diagram of the Fetch Execute Cycle

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Explain different phases of an instruction cycle with flow diagram	June 2009	7
Q.2	What is Instruction cycle? Explain different phases of instruction cycle and show flow chart for instruction cycle	June 2012,Dec 2013	7

UNIT 01/LECTURE 06

- Program counter (PC) - an incrementing counter that keeps track of the memory address of the instruction that is to be executed next or in other words, holds the address of the next instruction to be executed next.
- Memory address register (MAR) - holds the address of a memory block to be read from or written to.
- Memory data register (MDR) - a two-way register that holds data fetched from memory (and ready for the CPU to process) or data waiting to be stored in memory.
- Instruction register (IR) - a temporary holding ground for the instruction that has just been fetched from memory.
- Control unit (CU) - decodes the program instruction in the IR, selecting machine resources such as a data source register and a particular arithmetic operation, and coordinates activation of those resources.
- Arithmetic logic unit (ALU) - performs mathematical and logical operations.

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

1. Fetching the instruction

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

2. Decode the instruction

The decoder interprets the instruction. During this cycle the instruction inside the IR (instruction register) gets decoded.

3. In case of a memory instruction (direct or indirect) the execution phase will be in 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: T3). If the instruction is direct, nothing is done at this clock pulse. If this is an I/O instruction or a Register instruction, the operation is performed (executed) at clock Pulse.

4. 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 condition of any feedback from the ALU, Program Counter may be updated to a different address from which the next instruction will be fetched.

The cycle is then repeated.

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Write the function of the following in computer system (i)Accumulator (ii) Instruction Register (iii) Program Counter (iii) Memory Address Register	June 2011,June 2012,Dec 2013,June 2013	7

UNIT 01/LECTURE 07

Data Movement Instructions

These instructions provide convenient methods for moving bytes, words, or doublewords of data between memory and the registers of the base architecture. They fall into the following classes:

1. General-purpose data movement instructions.
2. Stack manipulation instructions.
3. Type-conversion instructions.

General-Purpose Data Movement Instructions

MOV (Move) transfers a byte, word, or double word from the source operand to the destination operand. The MOV instruction is useful for transferring data along any of these paths There are also variants of MOV that operate on segment registers. These are covered in a later section of this chapter:

- To a register from memory
- To memory from a register
- Between general registers
- Immediate data to a register
- Immediate data to a memory

The MOV instruction cannot move from memory to memory or from segment register to segment register are not allowed. Memory-to-memory moves can be performed, however, by the string move instruction MOVS.

UNIT 01/LECTURE 08

Addressing modes of basic computer

Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand (or operands) of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere.

In computer programming, addressing modes are primarily of interest to compiler writers and to those who write code directly in assembly language.

- Immediate Addressing

An immediate instruction for loading 4 into register. For example

MOV	R1	4
-----	----	---

- Direct Addressing

A method for specifying an operand in memory is just to give its full address. This is called direct addressing. The instruction will always access exactly the same memory location. Thus direct addressing can only be used to access global variables whose address is known at compile time.

Many programs have global variables so this method is widely used.

- Register Addressing

Register addressing is conceptually the same as direct addressing but specifies a register rather

than a memory location. Because registers are so important this addressing mode is the most common one on most computers. Many compilers determine the most frequently used variables and place them in registers. This addressing mode is known as register mode.

- Register Indirect Addressing

In this mode, the operand being specified comes from memory or goes to memory, but its address is not hardwired into the instruction, as in direct addressing. Instead the address is contained in a register. When an address is used in this manner it is called a pointer. Register indirect addressing can reference memory without having a full memory address in the instruction.

- Indexed Addressing

It is frequently useful to be able to reference memory words at a known offset from a register. (Remember in IJVM we referenced local variables by giving their offset from LV).Addressing memory by giving a register (explicit or implicit) plus a constant offset is called indexed addressing .For example

MOV	R4	R2	124300
-----	----	----	--------

- Based Indexed Addressing

Some machines have an addressing mode in which the memory address is computed by adding up two registers plus an (optional) offset. Sometimes this mode is called based-indexed addressing.

- Stack Addressing

The ultimate limit in reducing address lengths is having no addresses at all. As we have seen, zero-address instructions, such as IADD are possible in conjunction with a stack. It is traditional in mathematics to put the operator between the operands ($x + y$), rather than after the operands ($x y +$).Between the operands is called infix notation.

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Explain the various types of addressing modes with an example?	June 2013	7
Q.2	Explain various addressing modes with the help of example.	June 2012	7
Q.3	Explain with an example , how effective address is calculated in different types of addressing modes	Dec 2010	6

UNIT 01/LECTURE 09

Organization of 8085 processor

The Intel 8085 is an 8-bit microprocessor introduced by Intel in 1977. It was binary-compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built.

The "5" in the model number came from the fact that the 8085 required only a +5-volt (V) power supply rather than the +5V, -5V and +12V supplies the 8080 needed. Both processors were sometimes used in computers running the CP/M operating system, and the 8085 later saw use as a microcontroller, by virtue of its low component count. Both designs were eclipsed for desktop computers by the compatible Zilog Z80, which took over most of the CP/M computer market as well as taking a share of the booming home computer market in the early-to-mid-1980s.

The 8085 had a long life as a controller. Once designed into such products as the DECtape controller and the VT100 video terminal in the late 1970s, it continued to serve for new production throughout the life span of those products (generally longer than the product life of desktop computers).

The 8085 Architecture follows the "von Neumann architecture", with a 16-bit address bus, and a 8-bit data bus. The 8085 used a multiplexed Data Bus i.e .the address was split between the 8-bit address bus and 8-bit data bus. (For saving Number of Pins).

Registers:

The 8085 can access 2¹⁶ (= 65,536) individual 8-bit memory locations, or in other words, its address space is 64 KB. Unlike some other microprocessors of its era, it has a separate address space for up to 2⁸ (=256) I/O ports. It also has a built in register array which are usually labeled A (Accumulator), B, C, D, E, H, and L. Further special-purpose registers are the 16-bit Program Counter (PC), Stack Pointer (SP), and 8-bit flag register F. The microprocessor has three maskable interrupts (RST 7.5, RST 6.5 and RST 5.5), one Non-Maskable interrupt (TRAP), and one externally serviced interrupt (INTR). The RST n.5 interrupts refer to actual pins on the processor-a feature which permitted simple systems to avoid the cost of a separate interrupt controller chip.

Buses:

Address bus - 16 line bus accessing 216 memory locations (64 KB) of memory.

Data bus - 8 line bus accessing one (8-bit) byte of data in one operation. Data bus width is the traditional measure of processor bit designations, as opposed to address bus width, resulting in the 8-bit microprocessor designation.

Control buses - Carries the essential signals for various operations.

MICROPROCESSOR:

A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to those instructions and provides result as output.

POWER SUPPLY & CLOCK FREQUENCY OF 8085:

The power supply of 8085 is +5V and clock frequency in 3MHz.

Applications of microprocessor-based system:

It is used:

- i. For measurements, display and control of current, voltage, temperature, pressure, etc.
- ii. For traffic control and industrial tool control.
- iii. For speed control of machines.

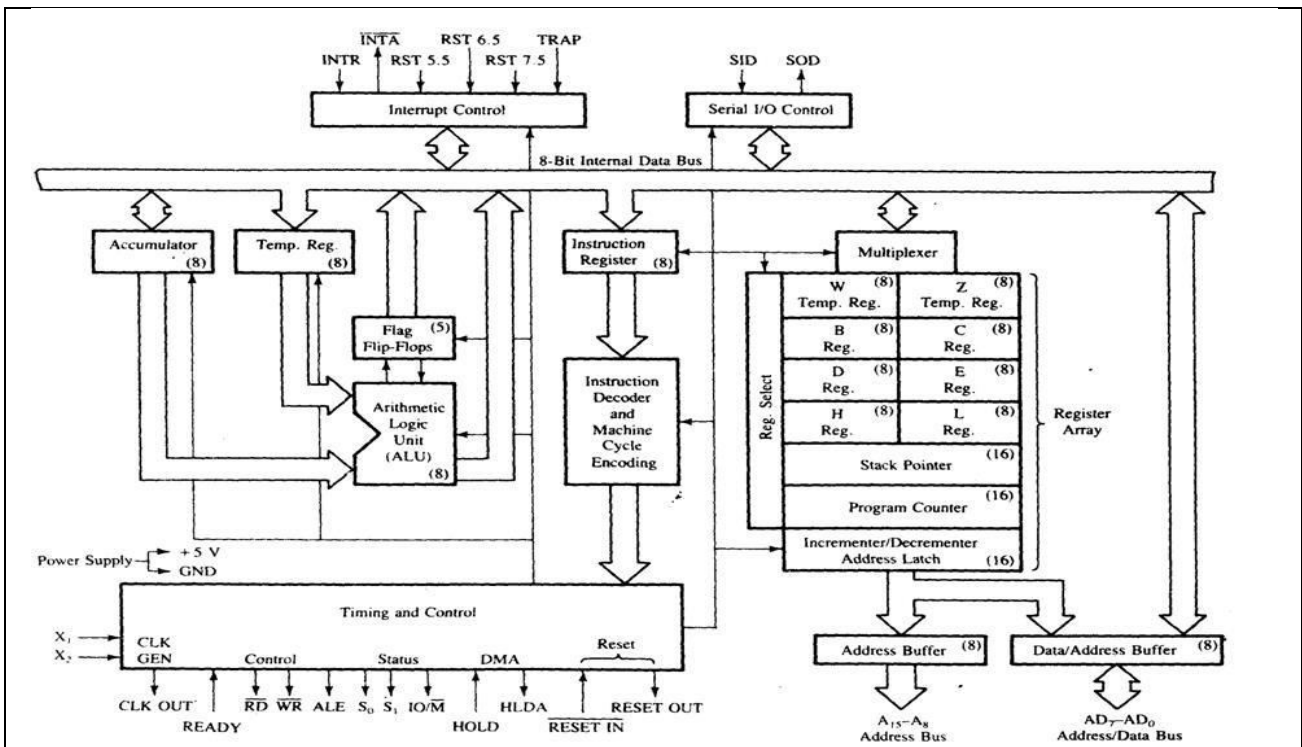


Fig 9.1 8085 MICROPROCESSOR ARCHITECTURE

FUNCTIONS OF AN ACCUMULATOR:

The accumulator is the register associated with the ALU operations and sometimes I/O operations. It is an integral part of ALU. It holds one of data to be processed by ALU. It also temporarily stores the result of the operation performed by the ALU.

16 – BIT REGISTERS OF 8085 MICROPROCESSOR:

Stack pointer (SP) and Program counter (PC).

REGISTER ORGANIZATION OF 8085:

W(8)	Z(8)
Temp. Reg	Temp. Reg
B(8)	C(8)
Register	Register
D(8)	E(8)
Register	Register
H(8)	L(8)
Register	Register
Stack Pointer(16)	
Program Counter(16)	

REGISTER PAIRS OF 8085:

B-C register pair

D-E register pair

H-L register pair

STACK AND STACK RELATED INSTRUCTIONS:

The stack is a group of memory locations in the R/W memory that is used for the temporary storage of binary information during the execution of the program.

The stack related instructions are PUSH & POP

CALL	PUSH
When CALL is executed the microprocessor automatically stores the 16-bit address of the instruction next to CALL on the stack	The programmer uses the instruction PUSH to save the contents of the register pair on the stack
When CALL is executed the stack pointer is decremented by two	When PUSH is executed the stack pointer register is decremented by two

COMPARISON RET AND POP:

RET	POP
RET transfers the contents of the top two locations of the stack to the PC	POP transfers the contents of the top two locations of the stack to the specified register pair
When RET is executed the SP is incremented by two	When POP is executed the SP is incremented by two
Has 8 conditional RETURN instructions	No conditional POP instructions

PURPOSE OF SID AND SOD LINES:

SID (Serial input data line): It is an input line through which the microprocessor accepts serial data.

SOD (Serial output data line): It is an output line through which the microprocessor sends output serial data.

OPCODE:

The part of the instruction that specifies the operation to be performed is called the operation code or Opcode.

FUNCTION OF IO/M SIGNAL IN THE 8085:

It is a status signal. It is used to differentiate between memory locations and I/O operations.

*When this signal is low (IO/M = 0) it denotes the memory related operations.

*When this signal is high (IO/M = 1) it denotes an I/O operation.

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Describe the major hardware functional unit of 8085 microprocessor with neat complete functional diagram	Dec 2010	7
Q.2	Draw and explain functional block diagram of 8085 microprocessor. Also draw its flag structure	June 2010, June 2011, June 2012	7