COMPUTER ORGANIZATION AND ARCHITECTURE

- Computer Architecture refers to those attributes of a system that have a direct impact on the logical execution of a program. Examples:
    - the instruction set
    - the number of bits used to represent various data types
    - I/O mechanisms
    - memory addressing techniques
- Computer Organization refers to the operational units and their interconnections that realize the architectural specifications. Examples are things that are transparent to the programmer:
    - control signals
    - interfaces between computer and peripherals
    - the memory technology being used
- So, for example, the fact that a multiply instruction is available is a computer architecture issue. How that multiply is implemented is a computer organization issue.

- Architecture is those attributes visible to the programmer
    - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
    - e.g. Is there a multiply instruction?
- Organization is how features are implemented
    - Control signals, interfaces, memory technology.
    - e.g. Is there a hardware multiply unit or is it done by repeated addition?
- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture
- This gives code compatibility
    - At least backwards
- Organization differs between different versions

STRUCTURE AND FUNCTION

- Structure is the way in which components relate to each other
- Function is the operation of individual components as part of the structure
- All computer functions are:

    - **Data processing**: Computer must be able to process data which may take a wide variety of forms and the range of processing.
    - **Data storage**: Computer stores data either temporarily or permanently.

    - **Data movement**: Computer must be able to move data between itself and the outside world.
    - **Control**: There must be a control of the above three functions.

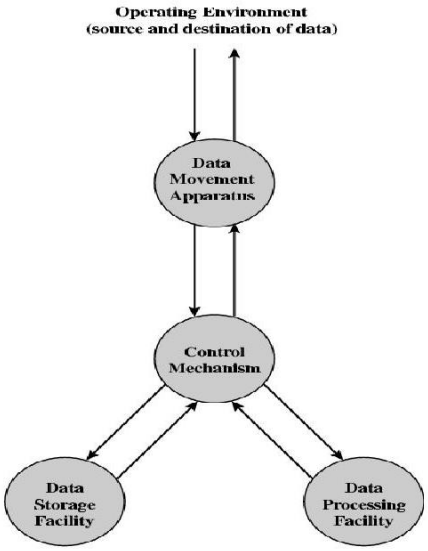


Fig: Functional view of a computer

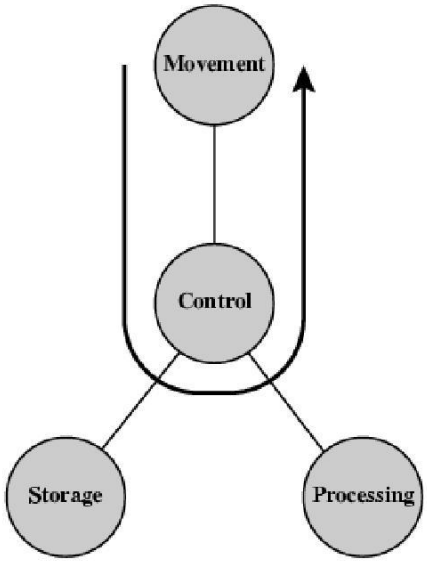


Fig: Data movement operation

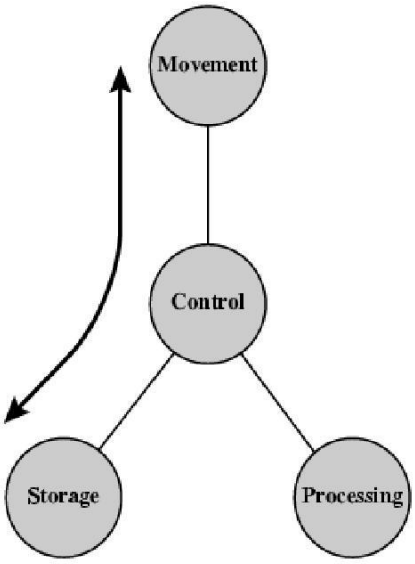


Fig: Storage Operation

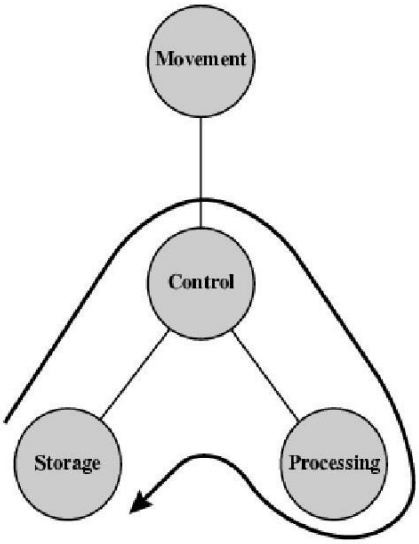


Fig: Processing from / to storage

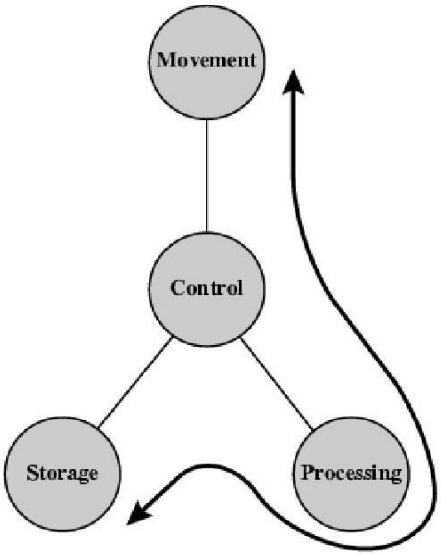


Fig: Processing from storage to i/o

- Four main structural components:
  - Central processing unit (CPU)

- Main memory
- I / O
- System interconnections

- CPU structural components:
  - Control unit
  - Arithmetic and logic unit (ALU)
  - Registers
  - CPU interconnections

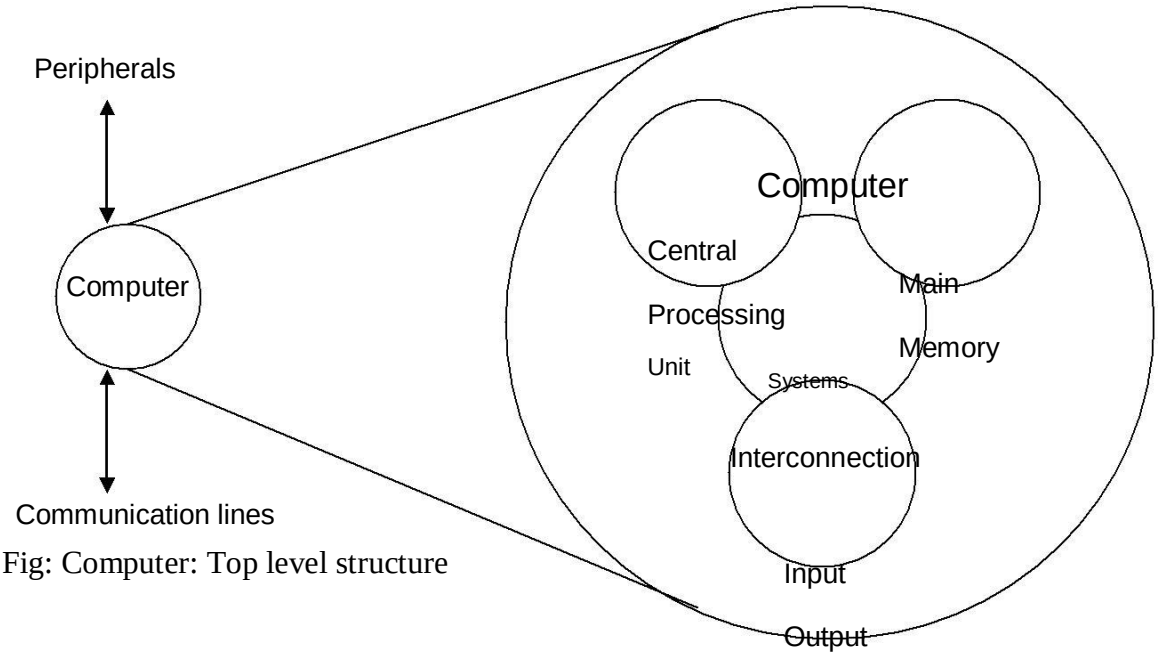


Fig: Computer: Top level structure

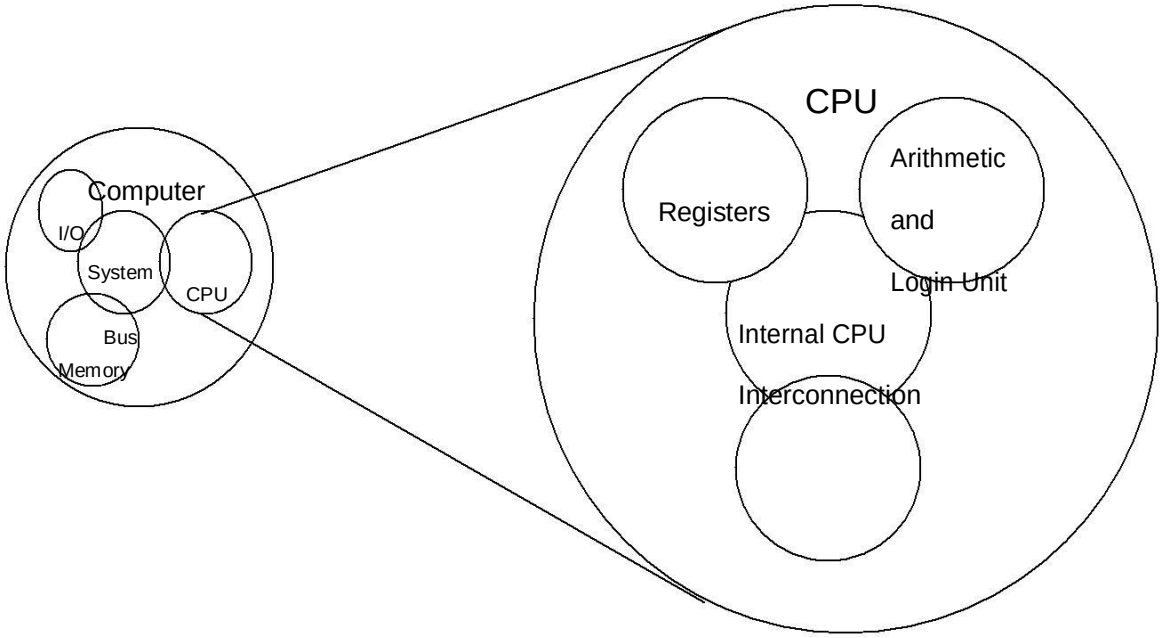


Fig: The central processing unit

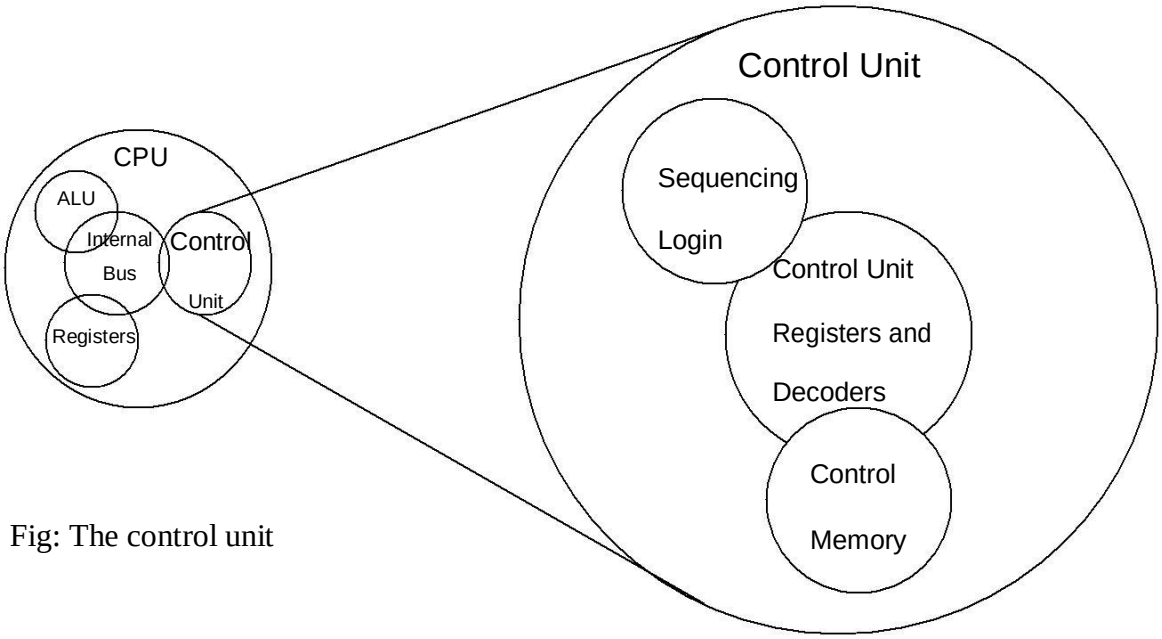
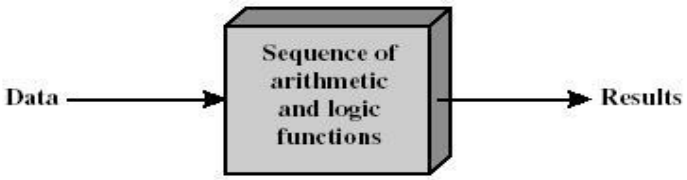


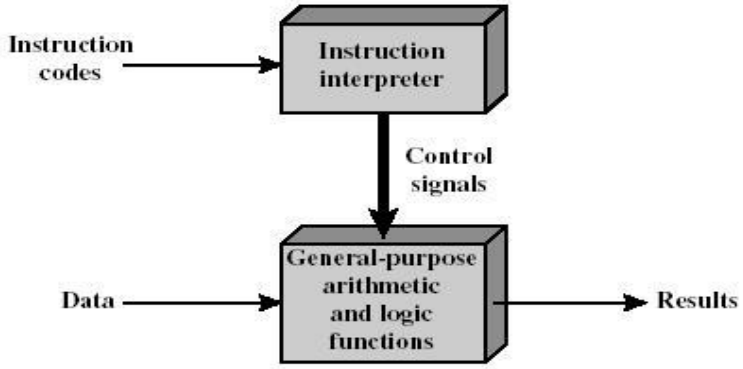
Fig: The control unit

COMPUTER COMPONENTS

- The Control Unit (CU) and the Arithmetic and Logic Unit (ALU) constitute the Central Processing Unit (CPU)
- Data and instructions need to get into the system and results need to get out
  - Input/output (I/O module)
- Temporary storage of code and results is needed
  - Main memory (RAM)
- Program Concept
  - Hardwired systems are inflexible
  - General purpose hardware can do different tasks, given correct control signals
  - Instead of re-wiring, supply a new set of control signals



(a) Programming in hardware



(b) Programming in software

COMPUTER FUNCTION

The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.

- Two steps of Instructions Cycle:
  - Fetch
  - Execute

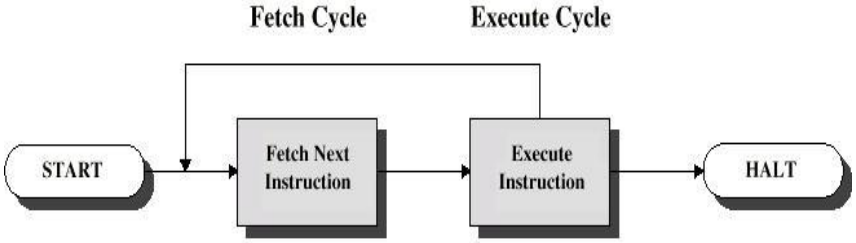


Fig: Basic Instruction Cycle

- Fetch Cycle
  - Program Counter (PC) holds address of next instruction to fetch
  - Processor fetches instruction from memory location pointed to by PC
  - Increment PC
    - Unless told otherwise
  - Instruction loaded into Instruction Register (IR)

- Execute Cycle
  - Processor interprets instruction and performs required actions, such as:
    - Processor - memory
      - data transfer between CPU and main memory
    - Processor - I/O
      - Data transfer between CPU and I/O module
    - Data processing
      - Some arithmetic or logical operation on data
    - Control
      - Alteration of sequence of operations
      - e.g. jump
    - Combination of above

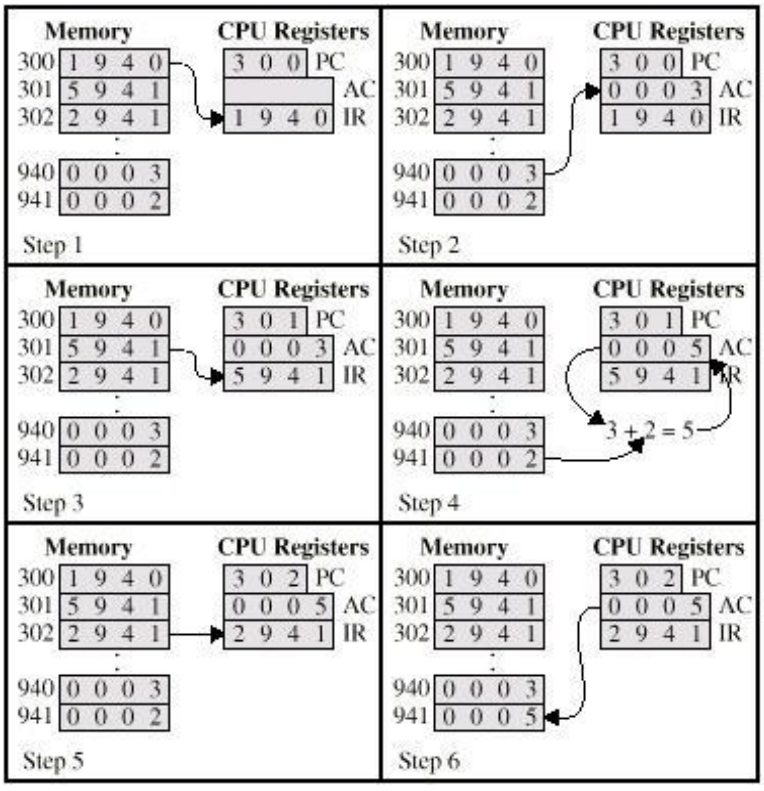


Fig: Example of program execution (consists of memory and registers in hexadecimal)

- The PC contains 300, the address of the first instruction. The instruction (the value 1940 in hex) is loaded into IR and PC is incremented. This process involves the use of MAR and MBR.
- The first hexadecimal digit in IR indicates that the AC is to be loaded. The remaining three hexadecimal digits specify the address (940) from which data are to be loaded.
- The next instruction (5941) is fetched from location 301 and PC is incremented.
- The old contents of AC and the contents of location 941 are added and the result is stored in the AC.
- The next instruction (2941) is fetched from location 302 and the PC is incremented.
- The contents of the AC are stored in location 941.

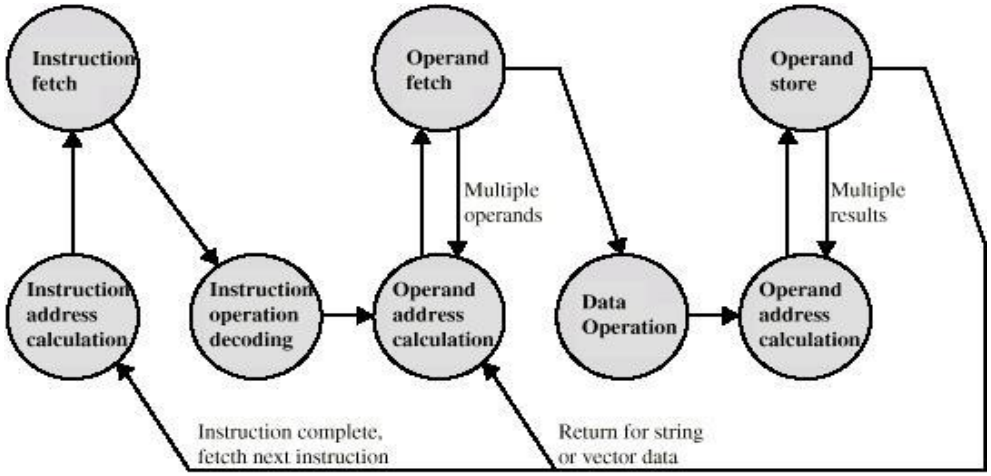
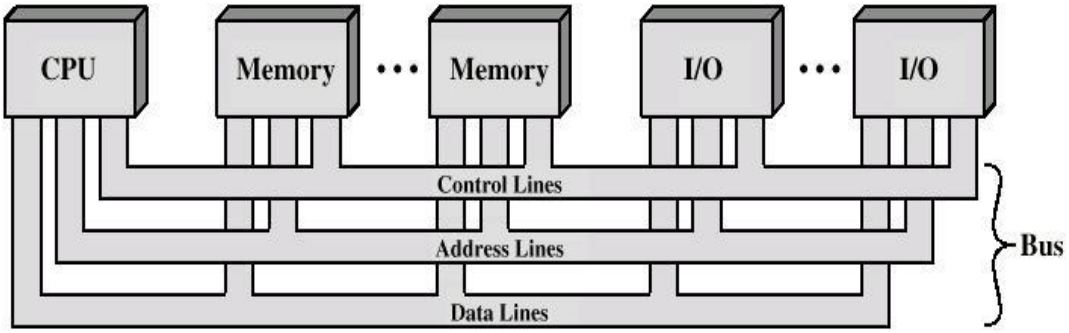


Fig: Instruction cycle state diagram

BUS INTERCONNECTION

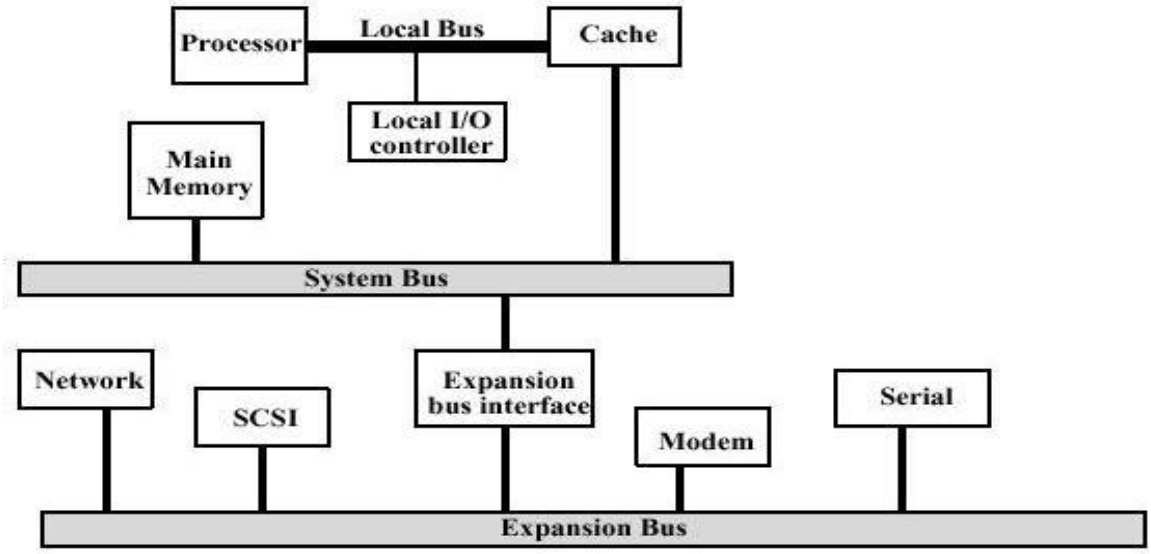
- A bus is a communication pathway connecting two or more devices
- Usually broadcast (all components see signal)
- Often grouped
  - A number of channels in one bus
  - e.g. 32 bit data bus is 32 separate single bit channels
- Power lines may not be shown
- There are a number of possible interconnection systems
- Single and multiple BUS structures are most common
- e.g. Control/Address/Data bus (PC)
- e.g. Unibus (DEC-PDP)
- Lots of devices on one bus leads to:
  - Propagation delays
  - Long data paths mean that co-ordination of bus use can adversely affect performance
  - If aggregate data transfer approaches bus capacity
- Most systems use multiple buses to overcome these problems



- **Data Bus**
  - Carries data
    - Remember that there is no difference between "data" and "instruction" at this level
  - Width is a key determinant of performance
    - 8, 16, 32, 64 bit
- **Address Bus**
  - Identify the source or destination of data
  - e.g. CPU needs to read an instruction (data) from a given location in memory
  - Bus width determines maximum memory capacity of system
    - e.g. 8080 has 16 bit address bus giving 64k address space
- **Control Bus**
  - Control and timing information
    - Memory read
    - Memory write
    - I/O read
    - I/O write
    - Transfer ACK
    - Bus request
    - Bus grant
    - Interrupt request
    - Interrupt ACK
    - Clock
    - Reset

Multiple Bus Hierarchies

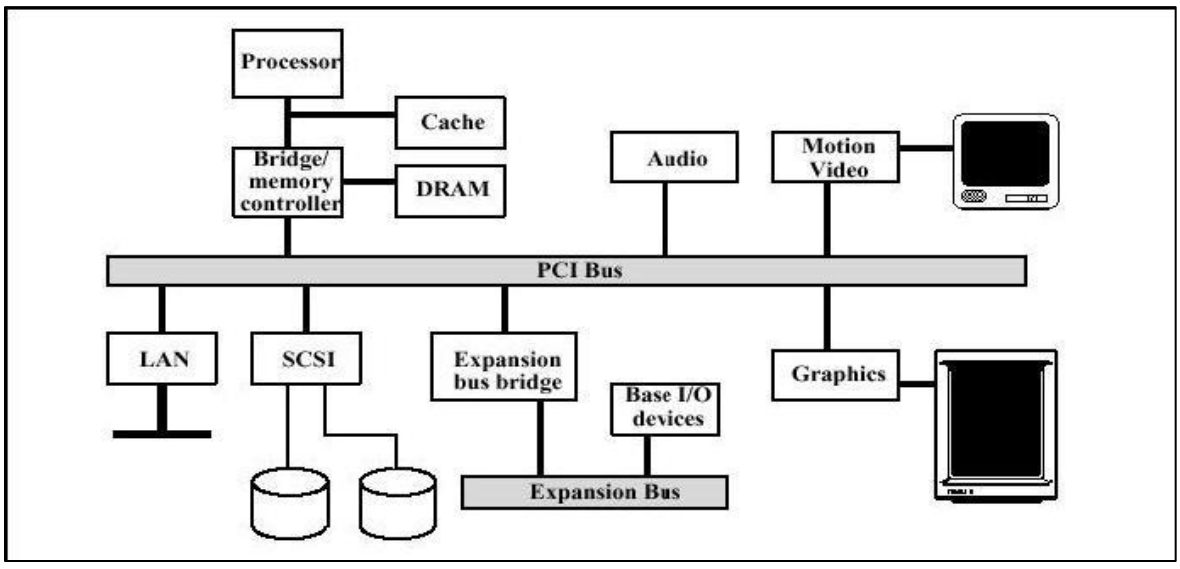
- A great number of devices on a bus will cause performance to suffer
  - Propagation delay - the time it takes for devices to coordinate the use of the bus
  - The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus (in available transfer cycles/second)
- Traditional Hierarchical Bus Architecture
  - Use of a cache structure insulates CPU from frequent accesses to main memory
  - Main memory can be moved off local bus to a system bus
  - Expansion bus interface
    - buffers data transfers between system bus and I/O controllers on expansion bus
    - insulates memory-to-processor traffic from I/O traffic



Traditional Hierarchical Bus Architecture Example

PCI

- PCI is a popular high bandwidth, processor independent bus that can function as mezzanine or peripheral bus.
- PCI delivers better system performance for high speed I/O subsystems (graphic display adapters, network interface controllers, disk controllers etc.)
- PCI is designed to support a variety of microprocessor based configurations including both single and multiple processor system.
- It makes use of synchronous timing and centralised arbitration scheme.
- PCI may be configured as a 32 or 64-bit bus.
- Current Standard
    - up to 64 data lines at 33Mhz
    - requires few chips to implement
    - supports other buses attached to PCI bus
    - public domain, initially developed by Intel to support Pentium-based systems
    - supports a variety of microprocessor-based configurations, including multiple processors
    - uses synchronous timing and centralized arbitration



Note: Bridge acts as a data buffer so that the speed of the PCI bus may differ from that of the processor's I/O capability.

PCI Bus Lines

- Systems lines
  - Including clock and reset
- Address & Data
  - 32 time mux lines for address/data
  - Interrupt & validate lines
- Interface Control
- Arbitration
  - Not shared
  - Direct connection to PCI bus arbiter
- Interrupt lines
  - Not shared
- Cache support
- 64-bit Bus Extension
  - Additional 32 lines
  - Time multiplexed
  - 2 lines to enable devices to agree to use 64-bit transfer
- JTAG/Boundary Scan
  - For testing procedures

CPU REGISTERS

In *computer architecture*, a processor **register** is a very fast computer memory used to speed the execution of computer programs by providing quick access to commonly used values-typically, the values being in the midst of a calculation at a given point in time.

These **registers** are the top of the memory hierarchy, and are the fastest way for the system to manipulate data. In a very simple *microprocessor*, it consists of a single memory location, usually called an *accumulator*. **Registers** are built from fast multi-ported memory cell. They must be able to drive its data onto an internal bus in a single clock cycle. The result of ALU operation is stored here and could be re-used in a subsequent operation or saved into memory.

Registers are normally measured by the number of bits they can hold, for example, an "8-bit register" or a "32-bit register". Registers are now usually implemented as a register file, but they have also been implemented using individual flip-flops, high speed core memory, thin film memory, and other ways in various machines.

The term is often used to refer only to the group of registers that can be directly indexed for input or output of an instruction, as defined by the instruction set. More properly, these are called the "*architected registers*". For instance, the x86 instruction set defines a set of eight 32-bit registers, but a CPU that implements the X86 instruction set will contain many more hardware registers than just these eight.

There are several other classes of registers:

(a) **Accumulator**: It is most frequently used register used to store data taken from memory. Its number varies from microprocessor to microprocessor.

(b) **General Purpose registers**: General purpose registers are used to store data and intermediate results during program execution. Its contents can be accessed through assembly programming.

(c) **Special purpose Registers**: Users do not access these registers. These are used by computer system at the time of program execution. Some types of special purpose registers are given below:

- **Memory Address Register (MAR**): It stores address of data or instructions to be fetched from memory.
- **Memory Buffer Register (MBR)**: It stores instruction and data received from the memory and sent from the memory.
- **Instruction Register (IR):** Instructions are stored in instruction register. When one instruction is completed, next instruction is fetched in memory for processing.
- **Program Counter (PC):** It counts instructions.

The instruction cycle is completed into two phases:

(a) **Fetch Cycle** and

(b) **Execute Cycle**.

There are two parts in instruction- opcode and operand. In fetch cycle **opcode** of instruction is fetched into CPU. The opcode, at first, is reached to Data Register (DR), then to Instruction Register (IR). Decoder accesses the opcode and it decodes opcode and type of operation is declared to CPU and execution cycle is started.

INSTRUCTION FORMATS

The computer can be used to perform a specific task, only by specifying the necessary steps to complete the task. The collection of such ordered steps forms a 'program' of a computer. These ordered steps are the instructions. Computer instructions are stored in central memory locations and are executed sequentially one at a time. The control reads an instruction from a specific address in memory and executes it. It then continues by reading the next instruction in sequence and executes it until the completion of the program. A computer usually has a variety of Instruction Code Formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction. An n bit instruction that k bits in the address field and m bits in the operation code field come addressed 2k location directly and specify 2m different operation

- The bits of the instruction are divided into groups called fields.
- The most common fields in instruction formats are:
  - o An Operation code field that specifies the operation to be performed.
  - o An Address field that designates a memory address or a processor register.
  - o A Mode field that specifies the way the operand or the effective address is determined.

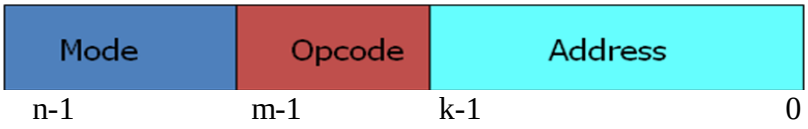


Fig: Instruction format with mode field

The operation code field (Opcode) of an instruction is a group of bits that define various processor operations such as add, subtract, complement, shift etcetera. The bits that define the mode field of an instruction code specify a variety of alternatives for choosing the operands from the given address. Operation specified by an instruction is executed on some data stored in the processor register or in the memory location. Operands residing in memory are specified by their memory address. Operands residing in processor register are specified with a register address.

Types of Instruction

- Computers may have instructions of several different lengths containing varying number of addresses.
- The number of address fields in the instruction format of a computer depends on the internal organization of its registers.

- Most computers fall into one of 3 types of CPU organizations:

Single accumulator organization:- All the operations are performed with an accumulator register. The instruction format in this type of computer uses one address field. For example: ADD X, where X is the address of the operands .

General register organization:- The instruction format in this type of computer needs three register address fields. For example: ADD R1,R2,R3

Stack organization:- The instruction in a stack computer consists of an operation code with no address field. This operation has the effect of popping the 2 top numbers from the stack, operating the numbers and pushing the sum into the stack. For example: ADD

Computers may have instructions of several different lengths containing varying number of addresses. Following are the types of instructions.

1. **Three address Instruction**
   With this type of instruction, each instruction specifies two operand location and a result location. A temporary location T is used to store some intermediate result so as not to alter any of the operand location. The three address instruction format requires a very complex design to hold the three address references.
   Format: Op X, Y, Z; X ← Y Op Z

   Example: ADD X, Y, Z; X ← Y + Z
   - ADVANTAGE: It results in short programs when evaluating arithmetic expressions.
   - DISADVANTAGE: The instructions requires too many bits to specify 3 addresses.

2. **Two address instruction**
   Two-address instructions are the most common in commercial computers. Here again each address field can specify either a processor register, or a memory word. One address must do double duty as both operand and result. The two address instruction format reduces the space requirement. To avoid altering the value of an operand, a MOV instruction is used to move one of the values to a result or temporary location T, before performing the operation.
   Format: Op X, Y; X ← X Op Y
   Example: SUB X, Y; X ← X - Y

3. **One address Instruction**
   It was generally used in earlier machine with the implied address been a CPU register known as accumulator. The accumulator contains one of the operand and is used to store the result. One-address instruction uses an implied accumulator (Ac) register for all data manipulation. All operations are done between the AC register and a memory operand. We use LOAD and STORE instruction for transfer to and from memory and Ac register.
   Format: Op X; Ac ← Ac Op X
   Example: MUL X; Ac ← Ac * X

4. **Zero address Instruction**
   It does not use address field for the instruction like ADD, SUB, MUL, DIV etc. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The name "Zero" address is given because of the absence of an address field in the computational instruction.
   Format: Op; TOS ← TOS Op (TOS – 1)
   Example: DIV; TOS ← TOS DIV (TOS – 1)

Example: To illustrate the influence of the number of address on computer programs, we will evaluate the arithmetic statement X=(A+B)*(C+D) using Zero, one, two, or three address instructions.

1. Three-Address Instructions:
   ADD R1, A, B; R1 ← M[A] + M[B]
   ADD R2, C, D; R2 ← M[C] + M[D]
   MUL X, R1,R2; M[X] ← R1 * R2

It is assumed that the computer has two processor registers R1 and R2. The symbol M[A] denotes the operand at memory address symbolized by A.

2. Two-Address Instructions:
   MOV R1, A; R1 ← M[A]
   ADD R1, B; R1 ← R1 + M[B]

   MOV R2, C; R2 ← M[C]
   ADD R2, D; R2 ← R2 + M[D]
   MUL R1, R2; R1 ← R1 * R2
   MOV X, R1; M[X] ← R1

3. One-Address Instruction:
   LOAD A; Ac ← M[A]
   ADD B; Ac ← Ac + M[B]
   STORE T; M[T] ← Ac
   LOAD C; Ac ← M[C]
   ADD D; Ac ← Ac + M[D]
   MUL T; Ac ← Ac * M[T]
   STORE X; M[X] ← Ac

   Here, T is the temporary memory location required for storing the intermediate result.

4. Zero-Address Instructions:
   PUSH A; TOS ← A
   PUSH B; TOS ← B
   ADD; TOS ← (A + B)
   PUSH C; TOS ← C
   PUSH D; TOS ← D
   ADD; TOS ← (C + D)
   MUL; TOS ← (C + D) * (A + B)
   POP X ; M[X] ← TOS

ADDRESSING MODES

- Specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.
- Computers use addressing mode techniques for the purpose of accommodating the following purposes:-
    - To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data and various other purposes.
    - To reduce the number of bits in the addressing field of the instructions.
        - Other computers use a single binary for operation & Address mode.
        - The mode field is used to locate the operand.
        - Address field may designate a memory address or a processor register.
        - There are 2 modes that need no address field at all (Implied & immediate modes).

Effective address (EA):

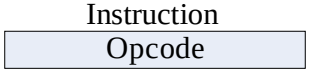
- The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode.
- The effective address is the address of the operand in a computational-type instruction.

The most well known addressing mode are:

- Implied Addressing Mode.
- Immediate Addressing Mode
- Register Addressing Mode
- Register Indirect Addressing Mode
- Auto-increment or Auto-decrement Addressing Mode
- Direct Addressing Mode
- Indirect Addressing Mode
- Displacement Address Addressing Mode
- Relative Addressing Mode
- Index Addressing Mode
- Stack Addressing Mode

Implied Addressing Mode:

- In this mode the operands are specified implicitly in the definition of the instruction.
  For example:- CMA - "complement accumulator" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction. In fact, all register reference instructions that use an accumulator are implied-mode instructions.



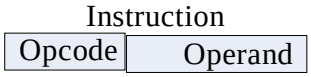
Advantage: no memory reference. Disadvantage: limited operand

Immediate Addressing mode:

- In this mode the operand is specified in the instruction itself. In other words, an

immediate-mode instruction has an operand field rather than an address field.

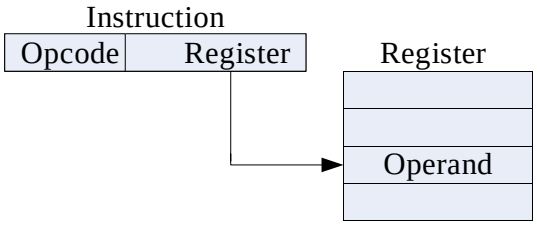
- This instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.
- These instructions are useful for initializing register to a constant value;
  For example MVI B, 50H



It was mentioned previously that the address field of an instruction may specify either a memory word or a processor register. When the address field specifies a processor register, the instruction is said to be in register-mode.
Advantage: no memory reference. Disadvantage: limited operand

Register direct addressing mode:

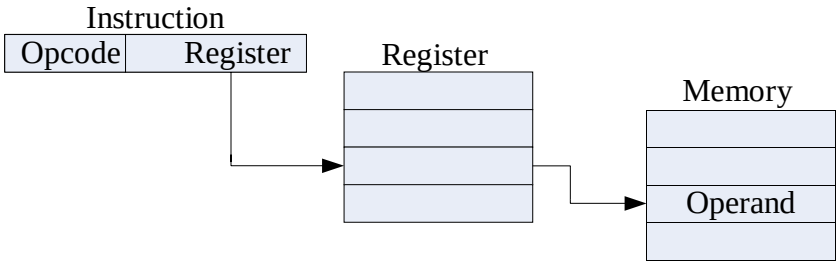
- In this mode, the operands are in registers that reside within the CPU.
- The particular register is selected from the register field in the instruction.
  For example MOV A, B



Effective Address (EA) = R
Advantage: no memory reference. Disadvantage: limited address space

Register indirect addressing mode:

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in the memory.
- In other words, the selected register contains the address of the operand rather than the operand itself.
- Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.
  For example LDAX B



Effective Address (EA) = (R)
Advantage: Large address space.

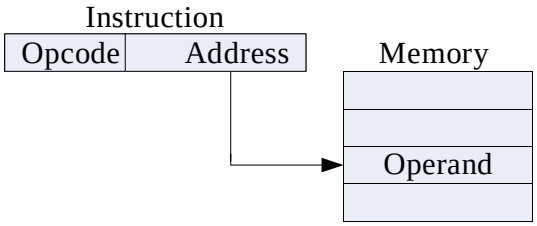
The address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.
Disadvantage: Extra memory reference

Auto increment or Auto decrement Addressing Mode:

- This is similar to register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- When the address stored in the registers refers to a table of data in memory, it is necessary to increment or decrement the registers after every access to the table.
- This can be achieved by using the increment or decrement instruction. In some computers it is automatically accessed.
- The address field of an instruction is used by the control unit in the CPU to obtain the operands from memory.
- Sometimes the value given in the address field is the address of the operand, but sometimes it is the address from which the address has to be calculated.

Direct Addressing Mode

- In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.
  For example LDA 4000H

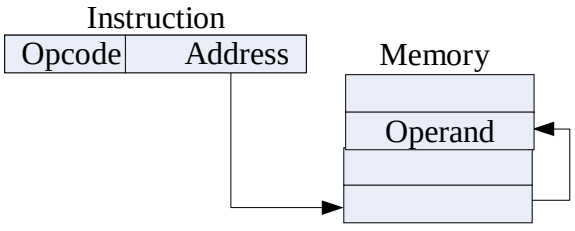


Effective Address (EA) = A

Advantage: Simple. Disadvantage: limited address field

Indirect Addressing Mode

- In this mode the address field of the instruction gives the address where the effective address is stored in memory.
- Control unit fetches the instruction from the memory and uses its address part to access memory again to read the effective address.

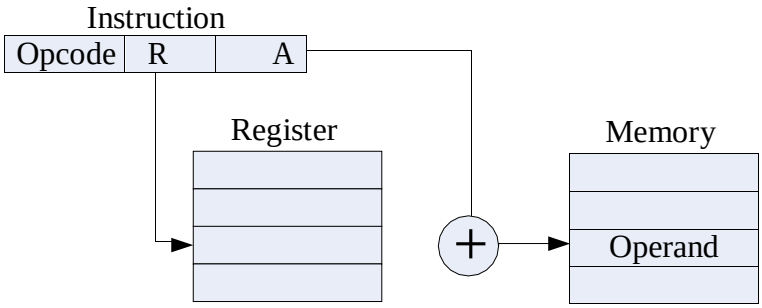


Effective Address (EA) = (A)

Advantage: Flexibility. Disadvantage: Complexity

Displacement Addressing Mode

- A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing.
- The address field of instruction is added to the content of specific register in the CPU.



Effective Address (EA) = A + (R)

Advantage: Flexibility. Disadvantage: Complexity

Relative Addressing Mode

- In this mode the content of the program counter (PC) is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number (either a +ve or a –ve number).
- When the number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction.
  Effective Address (EA) = PC + A

Indexed Addressing Mode

- In this mode the content of an index register (XR) is added to the address part of the instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value.
- Note: If an index-type instruction does not include an address field in its format, the instruction is automatically converted to the register indirect mode of operation.
  Effective Address (EA) = XR + A

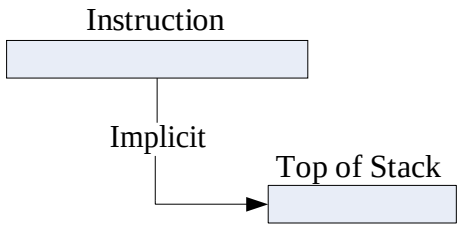
Base Register Addressing Mode

- In this mode the content of a base register (BR) is added to the address part of the instruction to obtain the effective address.
- This is similar to the indexed addressing mode except that the register is now called a base register instead of the index register.
- The base register addressing mode is used in computers to facilitate the relocation of programs in memory i.e. when programs and data are moved from one segment of memory to another.
  Effective Address (EA) = BR + A

Stack Addressing Mode

- The stack is the linear array of locations. It is some times referred to as push down list or

last in First out (LIFO) queue. The stack pointer is maintained in register.



Effective Address (EA) = TOS

Let us try to evaluate **Example.**

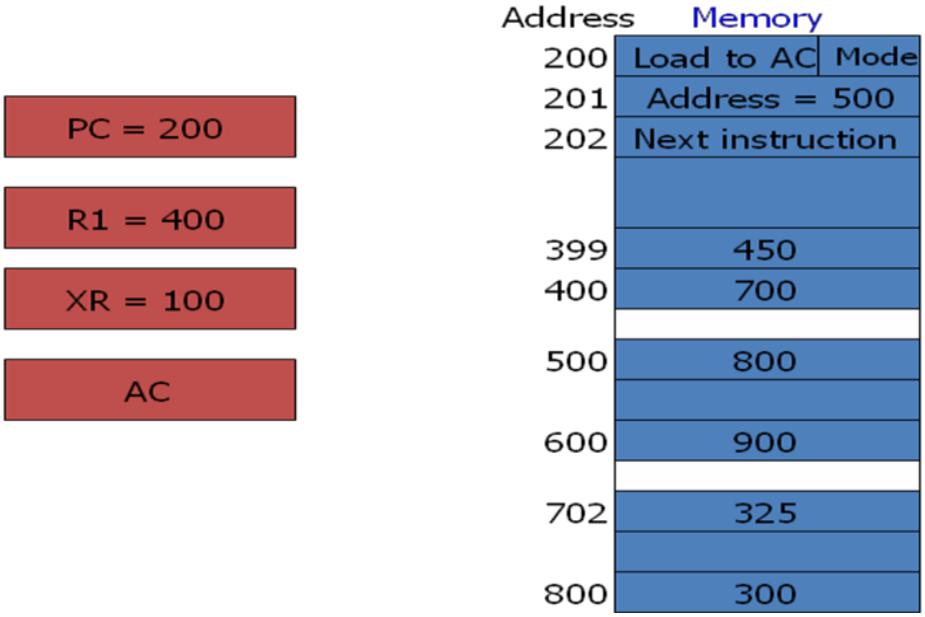


Fig: Numerical Example for Addressing Modes

Addressing Mode	Effective Address	Content of AC
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

Fig: Tabular list of Numerical Example

DATA TRANSFER AND MANIPULATION

Data transfer instructions cause transfer of data from one location to another without changing the binary information. The most common transfer are between the

- Memory and Processor registers
- Processor registers and input output devices
- Processor registers themselves

Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

DATA MANIPULATION INSTRUCTIONS

Data manipulation instructions perform operations on data and provide the computational capabilities for the computer. These instructions perform arithmetic, logic and shift operations.

Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

LOGICAL AND BIT MANIPULATION INSTRUCTIONS

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

SHIFT INSTRUCTIONS

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

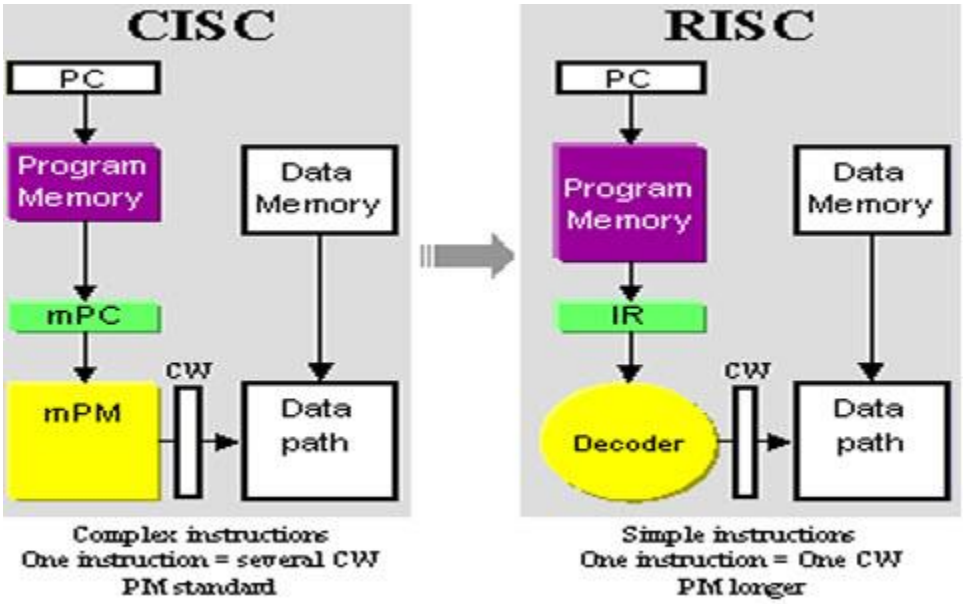
PROGRAM CONTROL INSTRUCTIONS

The program control instructions provide decision making capabilities and change the path taken by the program when executed in computer. These instructions specify conditions for altering the content of the program counter. The change in value of program counter as a result of execution of program control instruction causes a break in sequence of instruction execution. Some typical program control instructions are:

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

RISC AND CISC

- Important aspect of computer – design of the instruction set for processor.
- Instruction set – determines the way that machine language programs are constructed.
- Early computers – simple and small instruction set, need to minimize the hardware used.
- Advent of IC – cheaper digital software, instructions intended to increase both in number of complexity.
- Many computers – more than 100 or 200 instructions, variety of data types and large number of addressing modes.



COMPLEX INSTRUCTION SET COMPUTERS (CISC)

- The trend into computer hardware complexity was influenced by various factors:
  - Upgrading existing models to provide more customer applications
  - Adding instructions that facilitate the translation from high-level language into machine language programs
  - Striving to develop machines that move functions from software implementation into hardware implementation
- A computer with a large number of instructions is classified as a *complex instruction set computer* (CISC).
- One reason for the trend to provide a complex instruction set is the desire to simplify the compilation and improve the overall computer performance.
- The essential goal of CISC architecture is to attempt to provide a single machine instruction for each statement that is written in a high-level language.
- Examples of CISC architecture are the DEC VAX computer and the IBM 370

computer. Other are 8085, 8086, 80x86 etc.

The major characteristics of CISC architecture

- A large number of instructions– typically from 100 to 250 instructions
- Some instructions that perform specialized tasks and are used infrequently
- A large variety of addressing modes—typically from 5 to 20 different modes
- Variable-length instruction formats
- Instructions that manipulate operands in memory
- Reduced speed due to memory read/write operations
- Use of microprogram – special program in control memory of a computer to perform the timing and sequencing of the microoperations – fetch, decode, execute etc.
- Major complexity in the design of microprogram
- No large number of registers – single register set of general purpose and low cost

REDUCED INSTRUCTION SET COMPUTERS (RISC)

A computer uses fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory as often. It is classified as a *reduced instruction set computer* (RISC).

- RISC concept – an attempt to reduce the execution cycle by simplifying the instruction set
- Small set of instructions – mostly register to register operations and simple load/store operations for memory access
- Each operand – brought into register using load instruction, computations are done among data in registers and results transferred to memory using store instruction
- Simplify instruction set and encourages the optimization of register manipulation
- May include immediate operands, relative mode etc.

The major characteristics of RISC architecture

- Relatively few instructions
- Relatively few addressing modes
- Memory access limited to load and store instructions
- All operations done within the registers of the CPU
- Fixed-length, easily decoded instruction format
- Single-cycle instruction execution
- Hardwired rather than microprogrammed control

Other characteristics attributed to RISC architecture

- A relatively large number of registers in the processor unit
- Use of overlapped register windows to speed-up procedure call and return
- Efficient instruction pipeline – fetch, decode and execute overlap
- Compiler support for efficient translation of high-level language programs into machine language programs
- Studies that show improved performance for RISC architecture do not differentiate between the effects of the reduced instruction set and the effects of a large register file.
- A large number of registers in the processing unit are sometimes associated with RISC processors.
- RISC processors often achieve 2 to 4 times the performance of CISC processors.
- RISC uses much less chip space; extra functions like memory management unit or floating point arithmetic unit can also be placed on same chip. Smaller chips allow a semiconductor mfg. to place more parts on a single silicon wafer, which can lower the per chip cost dramatically.
- RISC processors are simpler than corresponding CISC processors, they can be designed more quickly.

COMPARISON BETWEEN RISC AND CISC ARCHITECTURES

S.N.	RISC	CISC
1	Simple instructions taking one cycle	Complex instructions taking multiple cycles
2	Only load and store memory references	Any instructions may reference memory
3	Heavily pipelined	Not/less pipelined
4	Multiple register sets	Single register set
5	Complexity is in compiler	Complexity is in micro-programming
6	Instructions executed by hardware	Instructions interpreted by micro-programming
7	Fixed format instructions	Variable format instructions
8	Few instructions and modes	Large instructions and modes

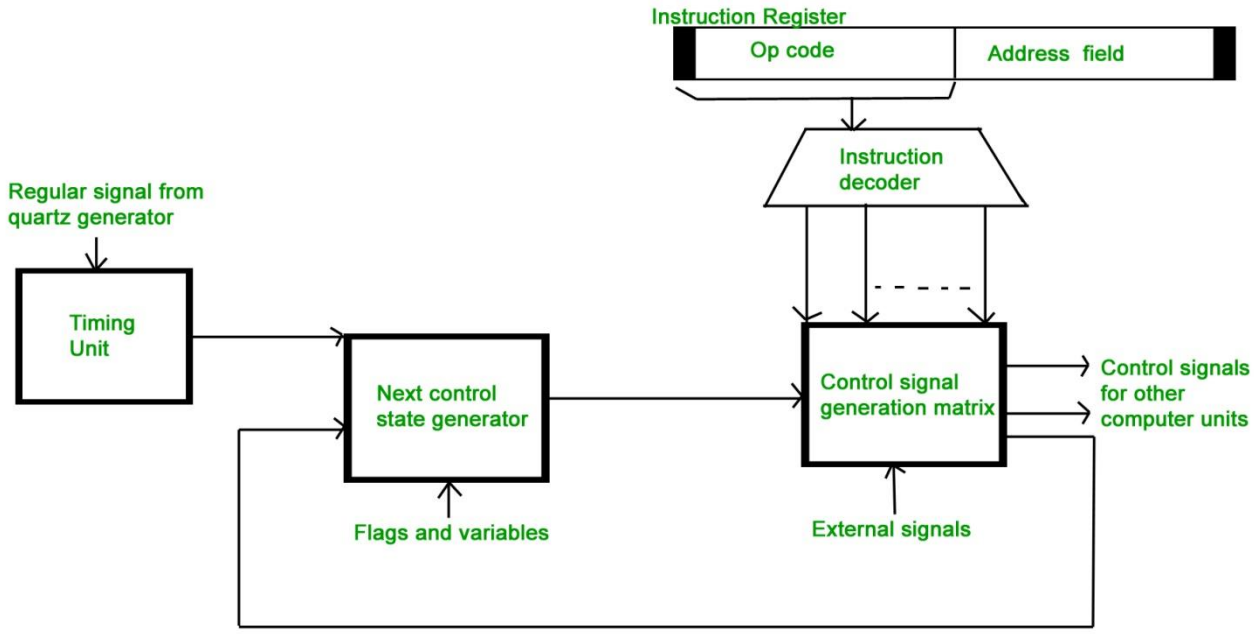
CONTROL UNIT

To execute an instruction, the control unit of the CPU must generate the required control signal in the proper sequence. There are two approaches used for generating the control signals in proper sequence as Hardwired Control unit and Micro-programmed control unit.

HARDWIRED CONTROL UNIT –

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as “hardwired”.

- Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
- Hardwired control is faster than micro-programmed control.
- A controller that uses this approach can operate at high speed.



MICRO-PROGRAMMED CONTROL UNIT –

- The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
- Control signals are generated by a program are similar to machine language programs.
- Micro-programmed control unit is slower in speed because of the time it takes to fetch microinstructions from the control memory.

Some Important Terms –

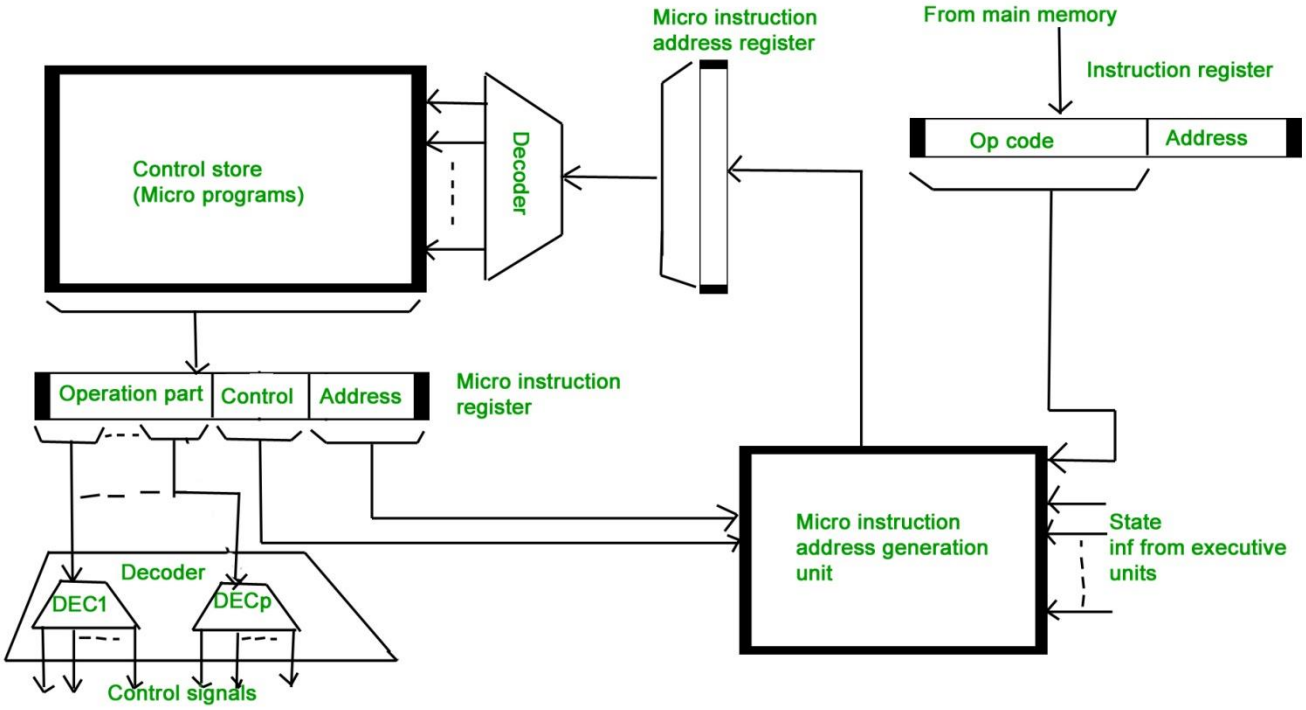
1.Control Word : A control word is a word whose individual bits represent various control signals.

2.Micro-routine : A sequence of control words corresponding to the control sequence of a machine instruction constitutes the micro-routine for that instruction.

3.Micro-instruction : Individual control words in this micro-routine are referred to as microinstructions.

4.Micro-program : A sequence of micro-instructions is called a micro-program, which is stored in a ROM or RAM called a Control Memory (CM).

5.Control Store : the micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the Control Store.



TYPES OF MICRO-PROGRAMMED CONTROL UNIT –

Based on the type of Control Word stored in the Control Memory (CM), it is classified into two types :

1. Horizontal Micro-programmed control Unit :

The control signals are represented in the decoded binary format that is 1 bit/CS. Example: If 53 Control signals are present in the processor than 53 bits are required. More than 1 control signal can be enabled at a time.

- It supports longer control word.
- It is used in parallel processing applications.
- It allows higher degree of parallelism. If degree is n, n CS are enabled at a time.
- It requires no additional hardware(decoders). It means it is faster than Verical Microprogrammed.
-

2. Vertical Micro-programmed control Unit :

The control signals re represented in the encoded binary format. For N control signals- Log2(N) bits are required.

- It supports shorter control words.
- It supports easy implementation of new conrol signals therefore it is more flexible.
- It allows low degree of parallelism i.e., degree of parallelism is either 0 or 1.
- Requires an additional hardware (decoders) to generate control signals, it implies it is slower than horizontal microprogrammed.

DIFFERENCE BETWEEN HARDWIRED CONTROL AND MICROPROGRAMMED CONTROL

Hardwired Control	Microprogrammed Control
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.
Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.