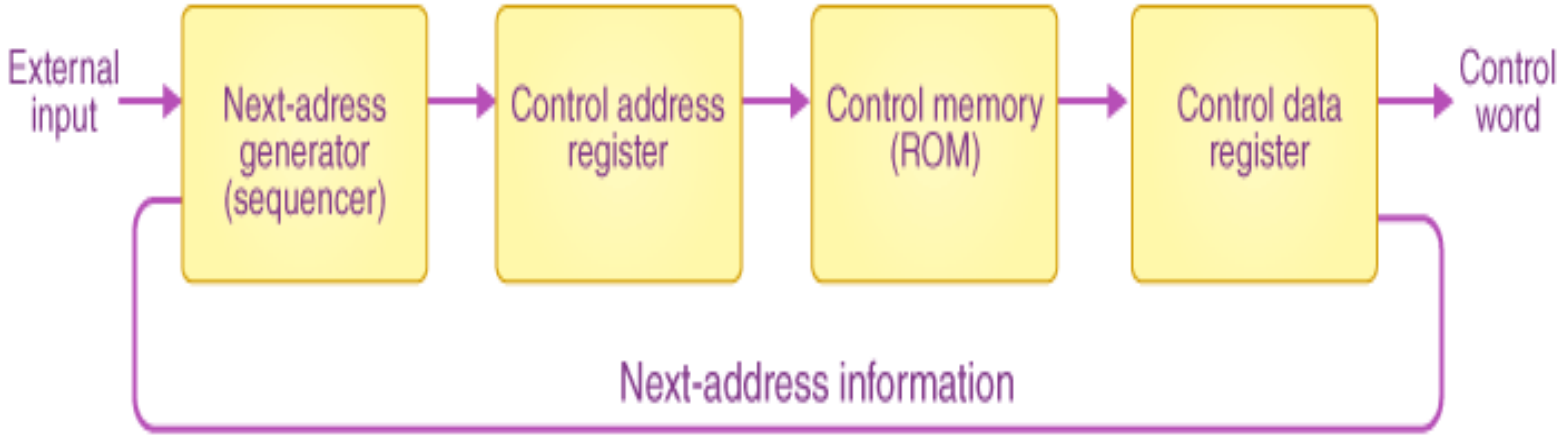
Computer Organization and Embedded Syste

Control and central processing unit

Control Memory

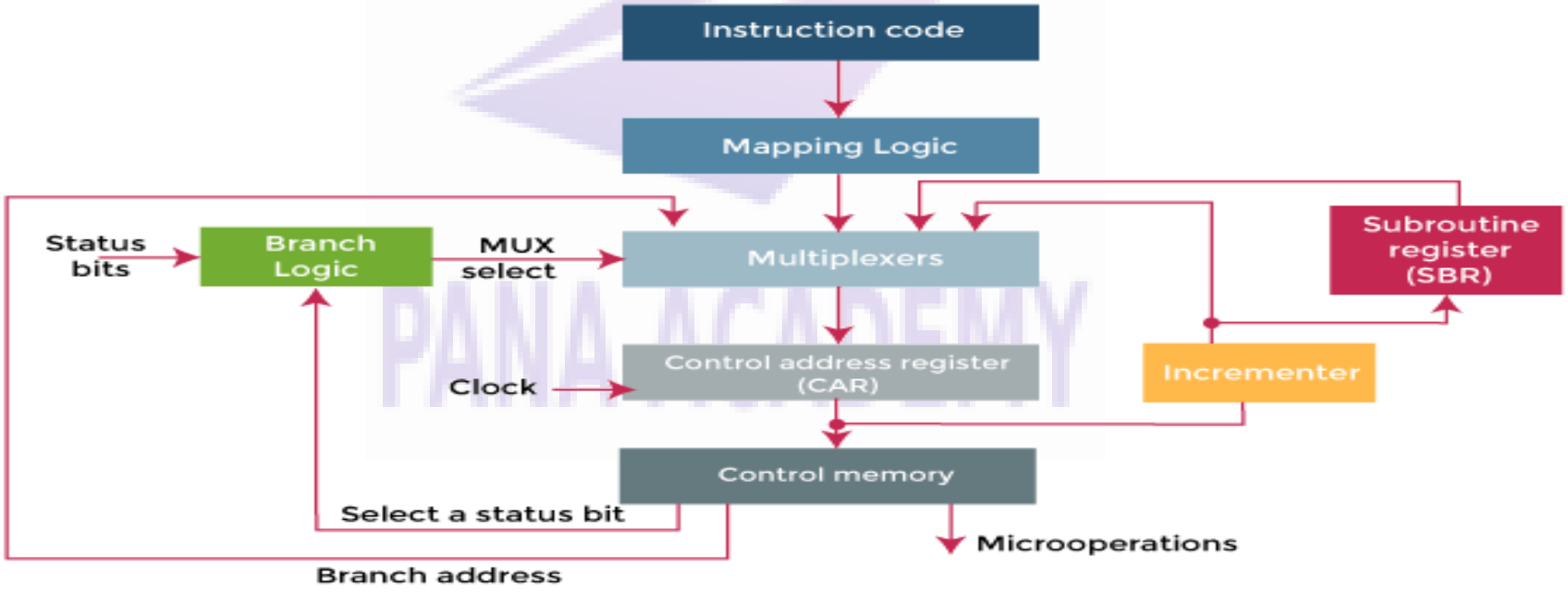
- CPU contains ALU, Register and Control Unit
- Here Control Unit is used to control flow of program execution
- **Micro-programmed Control Unit means the control logic or control signals are implemented with the help of program**
- It is a software
- Control memory mainly contains micro-programs



Microprogrammed control unit of a Basic Computer

Address Sequencing

- Micro-programmed control unit is used for address sequencing here
- Every micro-programs contain list of microinstruction
- And each instruction contains micro operands
- So how we get next instruction address is address sequencing
- Micro-program sequencer is required to generate next microinstruction address



Selection of Address for Control Memory

4 Approaches

Incrementing CAR by 1

- After increment operation, MUX transfer or load next address to CAR

Subroutine Register

- It is a function, that contains list of instruction used to perform specific task
- Mainly used for repitition purpose
- Whenever a function is called, control goes to function definition, all statements are executed
- **Once the execution is over, control comes to next statement in main function**
- **So, then the next statement is stored in subroutine register**
- **Then MUx transfer next address to CAR**

Unconditional/Conditional Branch

- Unconditional means without checking the condition the corresponding address will be loaded in to the CAR
- In conditional, branch logic checks for bits i.e. zero bit, status bit etc , if satisfies , instruction address will be loaded into CAR from MUX

Mapping instruction

- Here in this case , for example 1011 address for instruction has to loaded into CAR.
- Then, it will be mapped first for example 0101100 code, then will be loaded by MUX to CAR

Computer Configuration

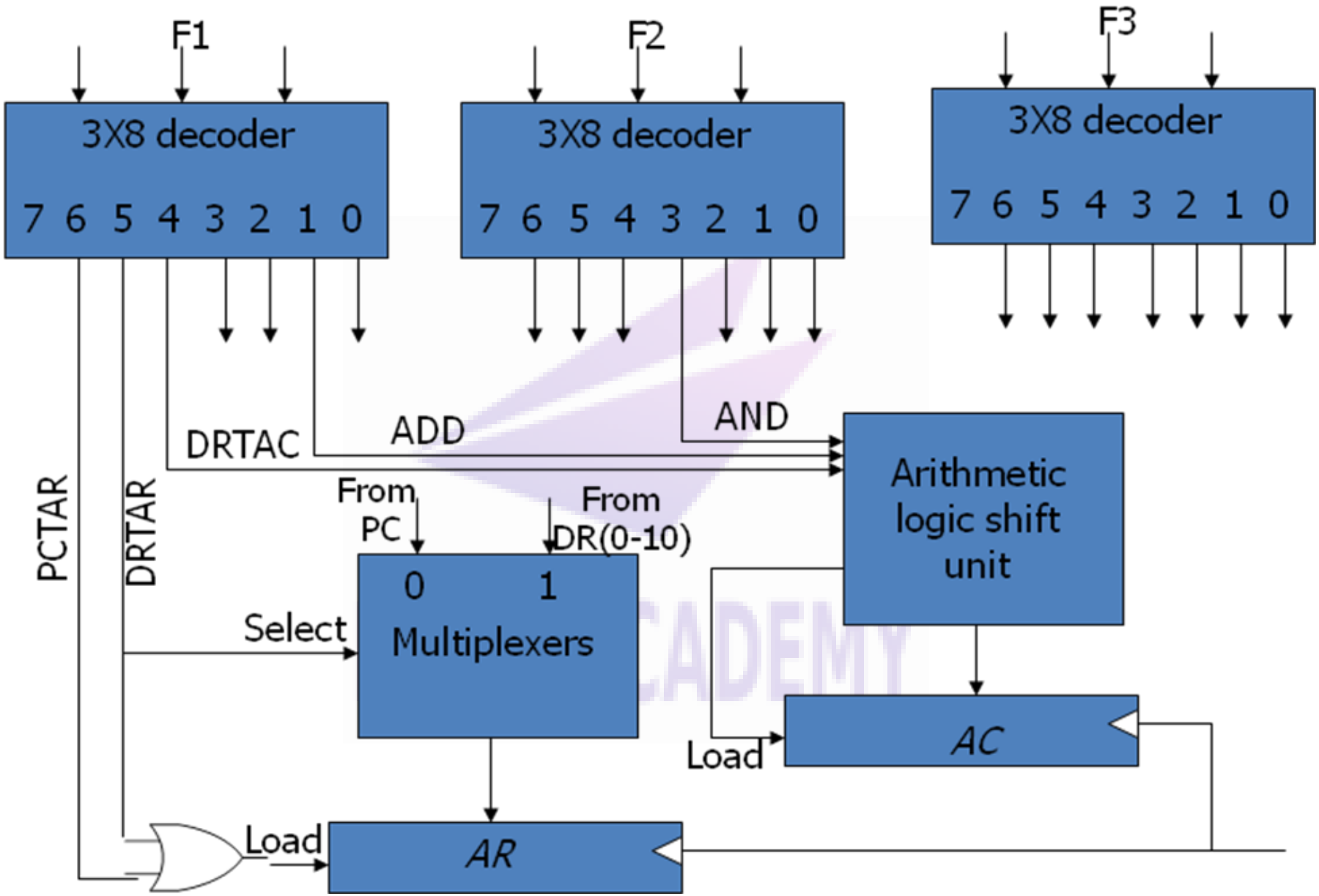
- Once the configuration and its micro-programmed control unit is established, the designer task is to generate the microcode for the control memory
- This microcode generation is called microprogramming
- Two memory units i.e. main memory for storing instruction and data and control memory for storing the microprogram
- Four registers are associated with the processor unit
  - Program counter pc, address register AR, dataregister DR and accumulator register AR
- The control unit has a control address register CAR and a sub routine register SBR
- The control memory and its register are organized as a microprogrammed control unit
- The transfer of information among the registers in the processor is done through multiplexer rather than a common bus

Design of Control Unit

- The bits of the microinstruction are usually divided into fields, with each field defining a distinct, separate function.
- The various fields encountered in instruction formats provide:
  - Control bits to initiate micro-operations in the system
  - Special bits to specify the way that the next address is to be evaluated
  - An address field for branching
- **The number of control bits that initiate microoperations can be reduced by grouping *mutually exclusive* variables into fields by encoding the *k* bits in each field to provide 2^k *microoperations.***
- Each field requires a decoder to produce the corresponding control signals.
  - Reduces the size of the microinstruction bits
  - Requires additional hardware external to the control memory

  - Increases the delay time of the control signals
- Figure shows the three decoders and some of the connections that must be made from their outputs.
- Outputs 5 or 6 of decoder F1 are connected to the load input of AR so that when either one of these outputs is active; information from the multiplexers is transferred to AR.
- The transfer into *AR* occurs with a clock pulse transition only when output 5 (from DR (0-10) to AR i.e. DRTAR) or output 6 (from PC to AR i.e. PCTAR) of the decoder are active.
- The arithmetic logic shift unit can be designed instead of using gates to generate the control signals; it comes from the outputs of the decoders.

Decoding of micro-operation fields

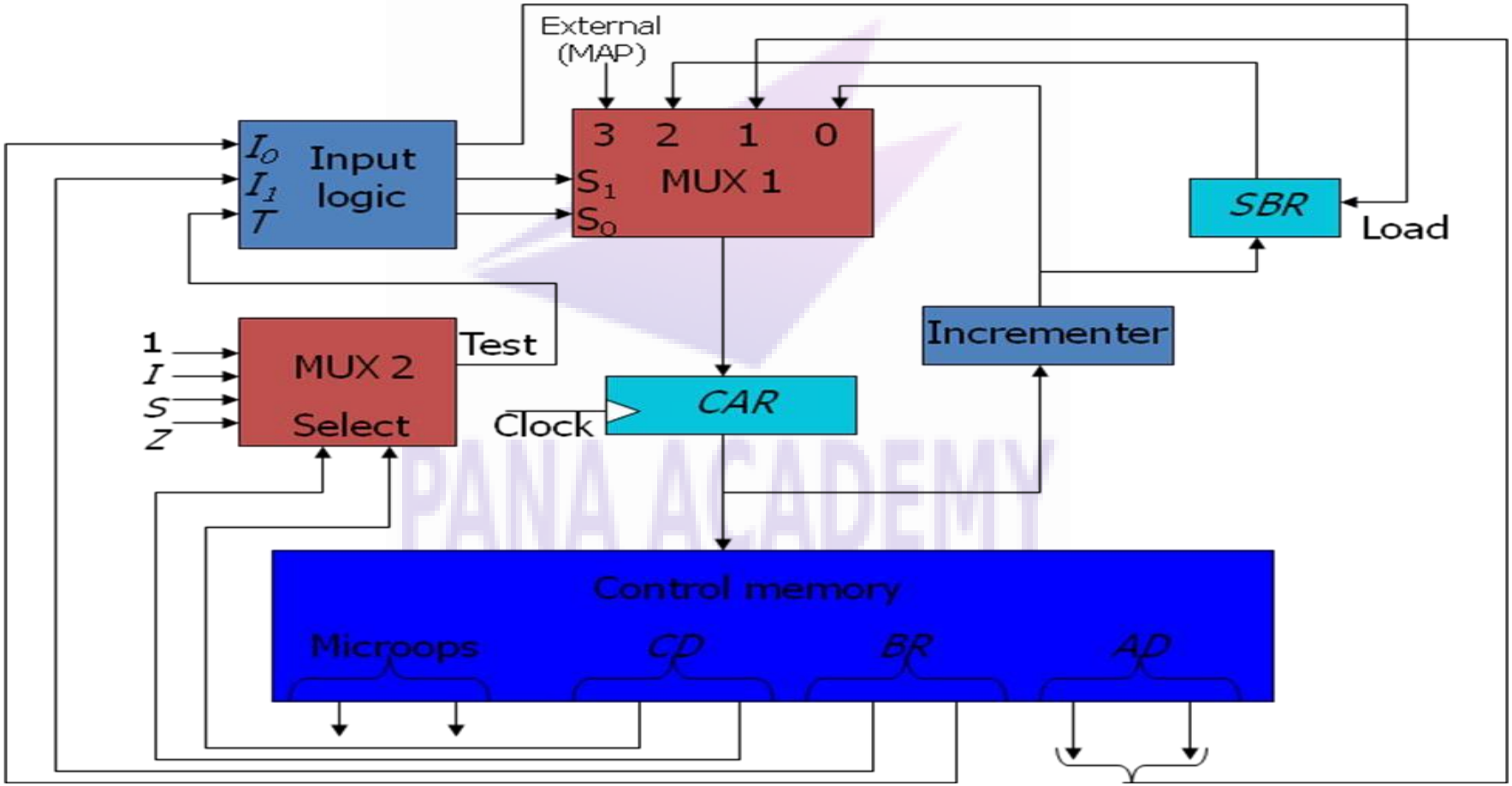


Micro-program Sequencer

- The basic components of a microprogrammed control unit are the *control memory* and *the circuits that select the next address*.
- The address selection part is called a *microprogram sequencer*.
- **A microprogram sequencer can be constructed with *digital functions* to suit a particular application.**
- To guarantee a wide range of acceptability, an *integrated circuit sequencer* must provide an internal organization that can be adapted to a wide range of application.
- **The purpose of a microprogram sequencer is to present an address to the control memory so that a microinstruction may be read and executed.**

- The block diagram of the microprogram sequencer is shown in figure.
  - The control memory is included to show the interaction between the sequencer and the attached to it.
  - There are two multiplexers in the circuit; first multiplexer selects an address from one of the four sources and routes to CAR, second multiplexer tests the value of the selected status bit and result is applied to an input logic circuit.
  - The output from CAR provides the address for control memory, contents of CAR incremented and applied to one of the multiplexers input and to the SBR.
  - Although the diagram shows a *single subroutine register*, a typical sequencer will have a *register stack* about four to eight levels deep. In this way, a push, pop operation and stack pointer operates for subroutine call and return instructions.

- The CD (Condition) field of the microinstruction selects one of the status bits in the second multiplexer.
- The Test variable (either 1 or 0) i.e. T value together with the two bits from the BR (Branch) field go to an input logic circuit.
- The input logic circuit determines the type of the operation.



Design of Input Logic

- **The input logic in a particular sequencer will determine the type of operations that are available in the unit.**
- Typical sequencer operations are: **increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack, and other address sequencing operations.**
- The truth table for the input logic circuit is shown in Table.
- Therefore, the simplified Boolean functions for the input logic circuit can be given as:
  - S1 = I1
  - S0 = I1.I0+ (I1)’ I0T
  - L = (I1)’ I0T

- The bit values for S1 and S0 are determined from the stated function and the path in the multiplexer that establishes the required transfer.

BR Field	Input I_1	I_0	T	MUX 1 S_1	S_0	Load *SBR* L
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	1
1	0	1	0	1	0	0
1	1	1	X	1	1	0

Microinstruction Format

Computer Instruction Format

- The computer instruction format is depicted in 3-5(a).
- It consists of three fields:
  - A 1-bit field for indirect addressing symbolized by *I*
  - A 4-bit operation code (*opcode*)
  - An 11-bit address field

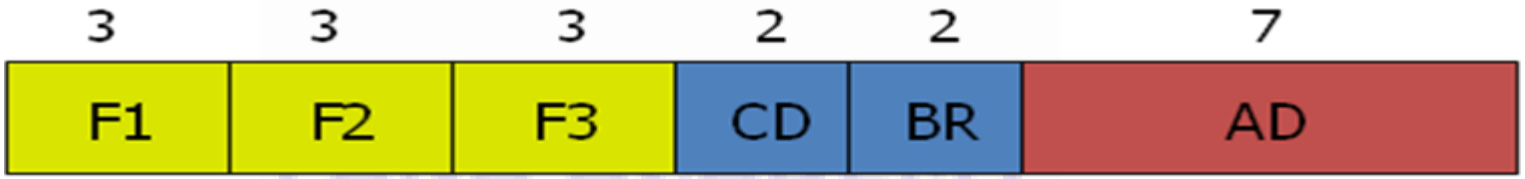
- Figure below lists four of the 16 possible memory-reference instructions.

Symbol	Opcode	Description
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	If(AC<0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

EA is the effective address

Microinstruction Format

- The microinstruction format for the control memory is shown below
- The 20 bits of the microinstruction are divided into four functional parts.
  - The three fields F1, F2, and F3 specify *microoperations* for the computer.
  - The CD field selects *status bit conditions*.
  - The BR field specifies *the type of branch*.
  - The AD field contains a *branch address*.



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

CPU Structure and Function

CPU must perform

Fetch Instructions

The CPU must read instructions from Memory.

Interpret Instructions

The instruction must have be decoded to determine what action is required.

Fetch data

The execution of an instruction may require reading from memory or I/O module.

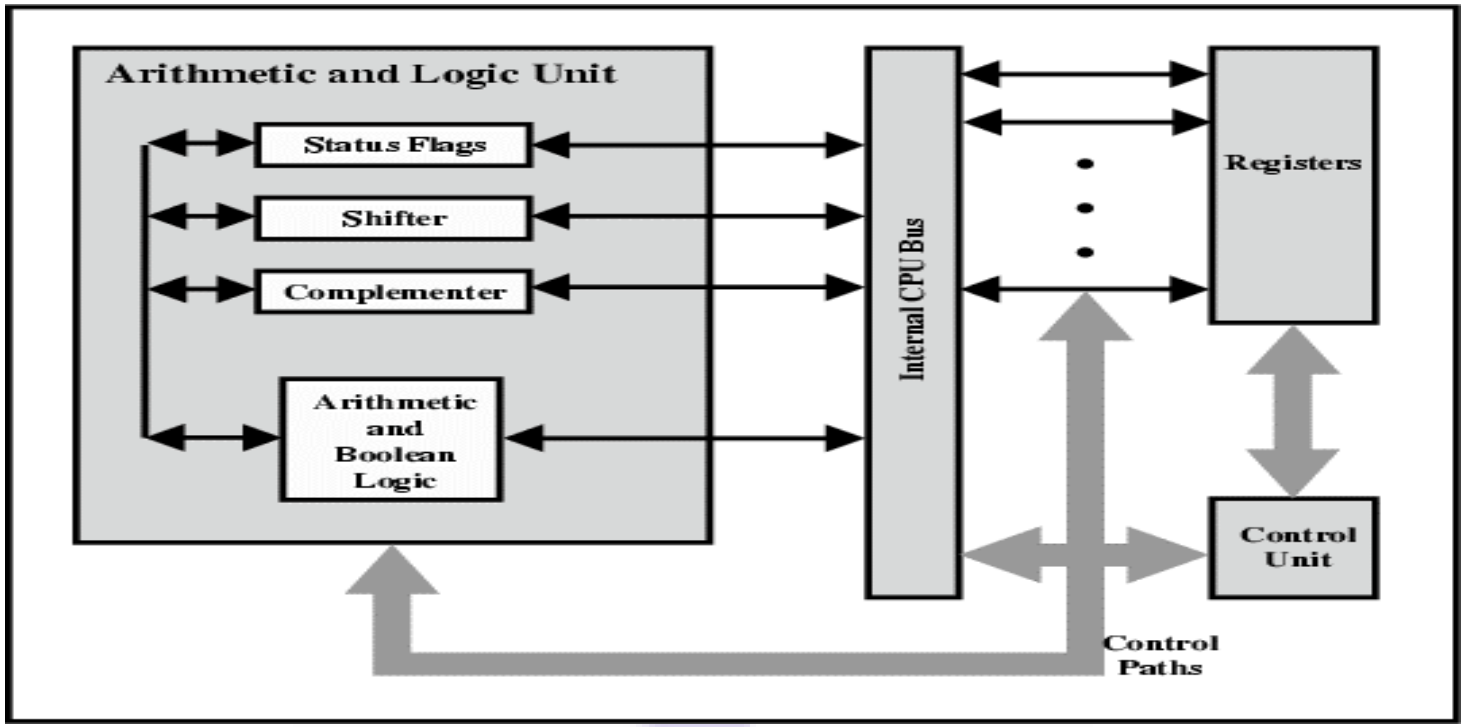
Process data

The execution of an instruction may require performing some arithmetic or logical operation on data.

Write Data

The results of an execution may require writing data to memory or an I/O module.

Internal structure of CPU



Components of the CPU

- Arithmetic and Logic Unit (ALU): does the actual computation or processing of data
- Control Unit (CU): controls the movement of data and instructions into and out of the CPU and controls the operation of the ALU.

The registers in the CPU serve two functions

User – Visible Registers

- These enable the machine or assembly language programmer to minimize main memory references by optimizing use of registers.

Control and status Registers

- These are used by the control unit to control the operation of the CPU and by privileged, operating system programs to control the execution of programs.

User-Visible Registers

- General Purpose registers - for variety of functions
- Data registers - hold data
- Address registers - hold address information
- Segment pointers - hold base address of the segment in use
- Index registers - used for indexed addressing and may be auto indexed
- Stack Pointer - a dedicated register that points to top of a Push, pop, and other stack instructions need not contain an explicit stack operand.
- Condition Codes (flags)

Control and Status Registers

Four registers are essential to instruction execution

Program Counter (PC)

- Contains the address of an instruction to be fetched.

Instruction Register (IR)

- Contains the instruction most recently fetched.

Memory Address Register (MAR)

- Contains the address of a location in memory.

Memory Buffer Register (MBR)

- Contains a word of data to be written to memory or the word most recently read

- The CPU design include a register or set of registers, often known as the program status word (PSW), that contain status information.
- The PSW typically contains condition codes plus other status information.
- Common fields or flags include the following : Sign - sign bit of last arithmetic operation
  - Zero - set when result of last arithmetic operation is 0
  - Carry - set if last op resulted in a carry into or borrow out of a high-order bit
  - Equal - set if a logical compare result is equality
  - Overflow - set when last arithmetic operation caused overflow
  - Interrupt Enable/Disable - used to enable or disable interrupts
  - Supervisor - indicates if privileged instructions can be used

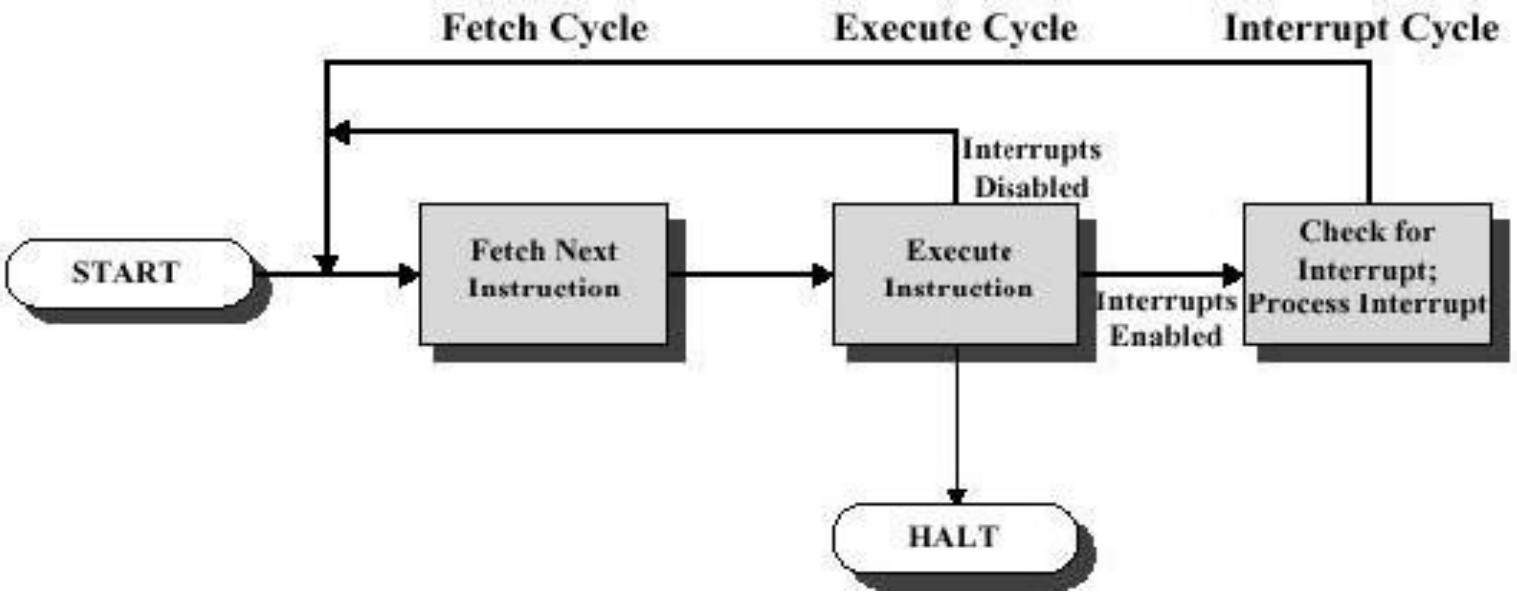
Other optional registers

- Pointer to a block of memory containing additional status info (like process control blocks)
- An interrupt vector
- A system stack pointer
- A page table pointer
- I/O registers

The Instruction Cycle

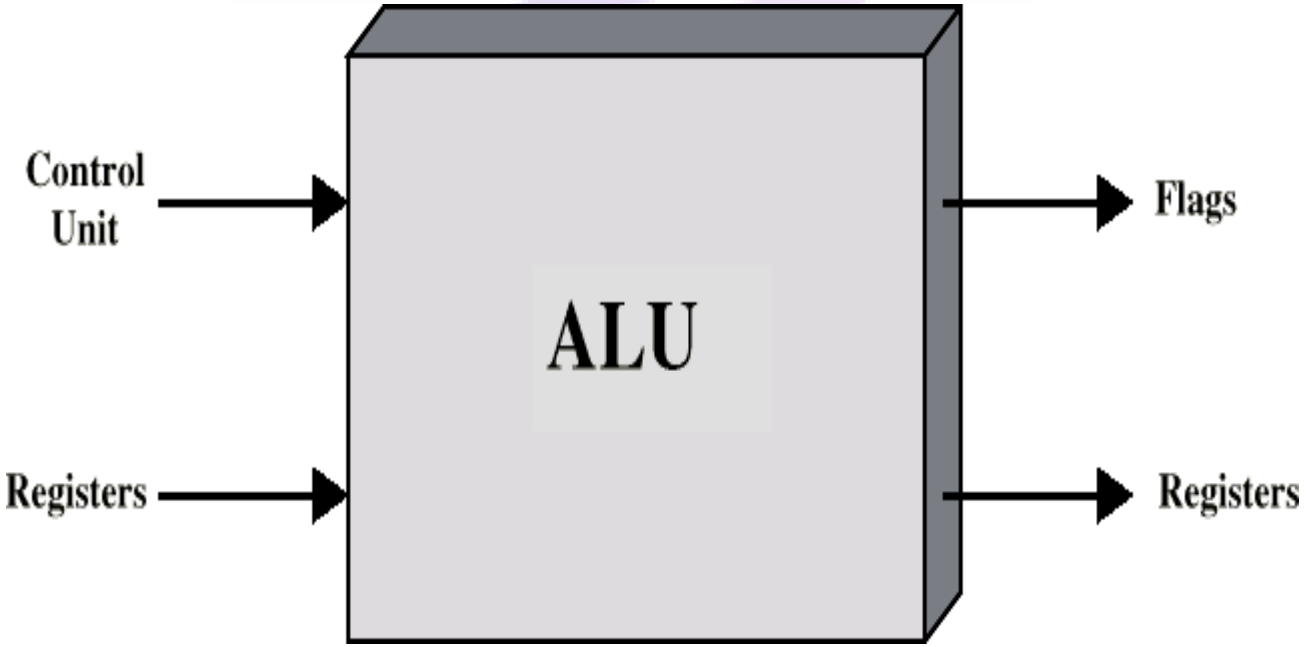
Basic instruction cycle contains the following sub-cycles.

- Fetch - read next instruction from memory into CPU
- Execute - Interpret the opcode and perform the indicated operation
- Interrupt - if interrupts are enabled and one has occurred, save the current process state and service the interrupt



Arithmetic and Logic Unit

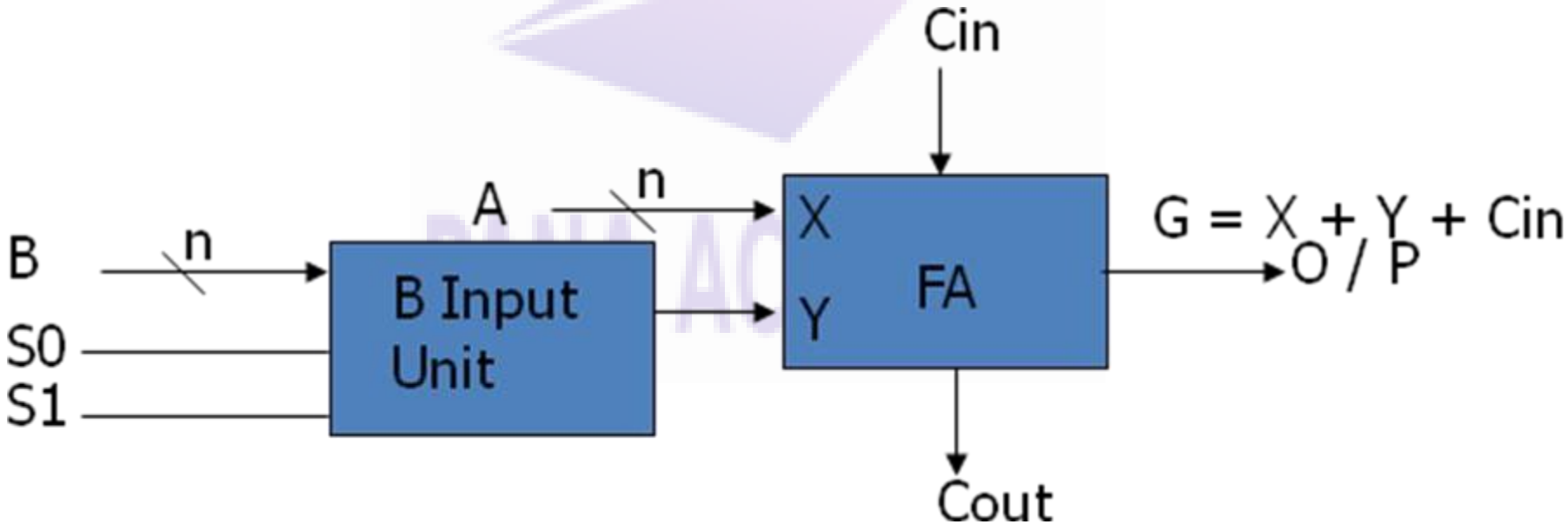
- is the combinational circuit of that part of computer that actually performs arithmetic and logical operations on data.
- are based on the use of simple digital logic device that can store binary digit and perform simple Boolean logic function.
- Data are presented to ALU in register and the result of operation is stored in register.



The design of ALU has three stages

Design the arithmetic section

- **The basic component of arithmetic circuit is a parallel adder which is constructed with a number of full adder circuits connected in cascade.**
- By controlling the data inputs to the parallel adder, it is possible to obtain different types of arithmetic operations.



Functional table for arithmetic section

Functional table for arithmetic unit:

Select		Input	Output		Microoperation	
S_1	S_0	Y	Cin = 0	Cin = 1	Cin = 0	Cin = 1
0	0	0	A	A+1	Transfer A	Increment A
0	1	B	A+B	A+B+1	Addition	Addition with carry
1	0	B'	A+B'	A+B'+1	Subtraction with borrow	Subtraction
1	1	-1	A-1	A	Decrement A	Transfer A

Design the logical section

- The basic components of logical circuit are AND, OR, XOR and NOT gate circuits connected accordingly.
- It consists of four gates and a multiplexer.
- Each of four logic operations is generated through a gate that performs the required logic.
- The two selection input S1 and S0 choose one of the data inputs of the multiplexer and directs its value to the output.
- Functional table lists the logic operations.

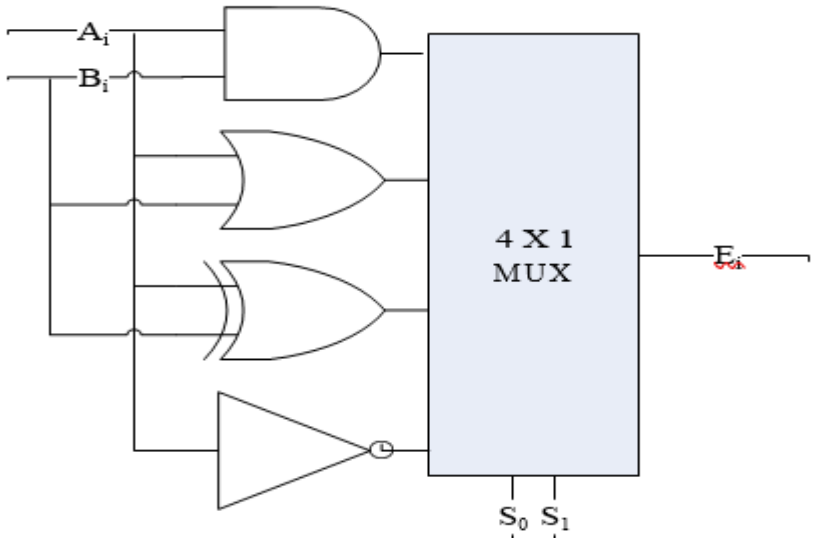


Fig: Block diagram of Logic Unit

Functional table

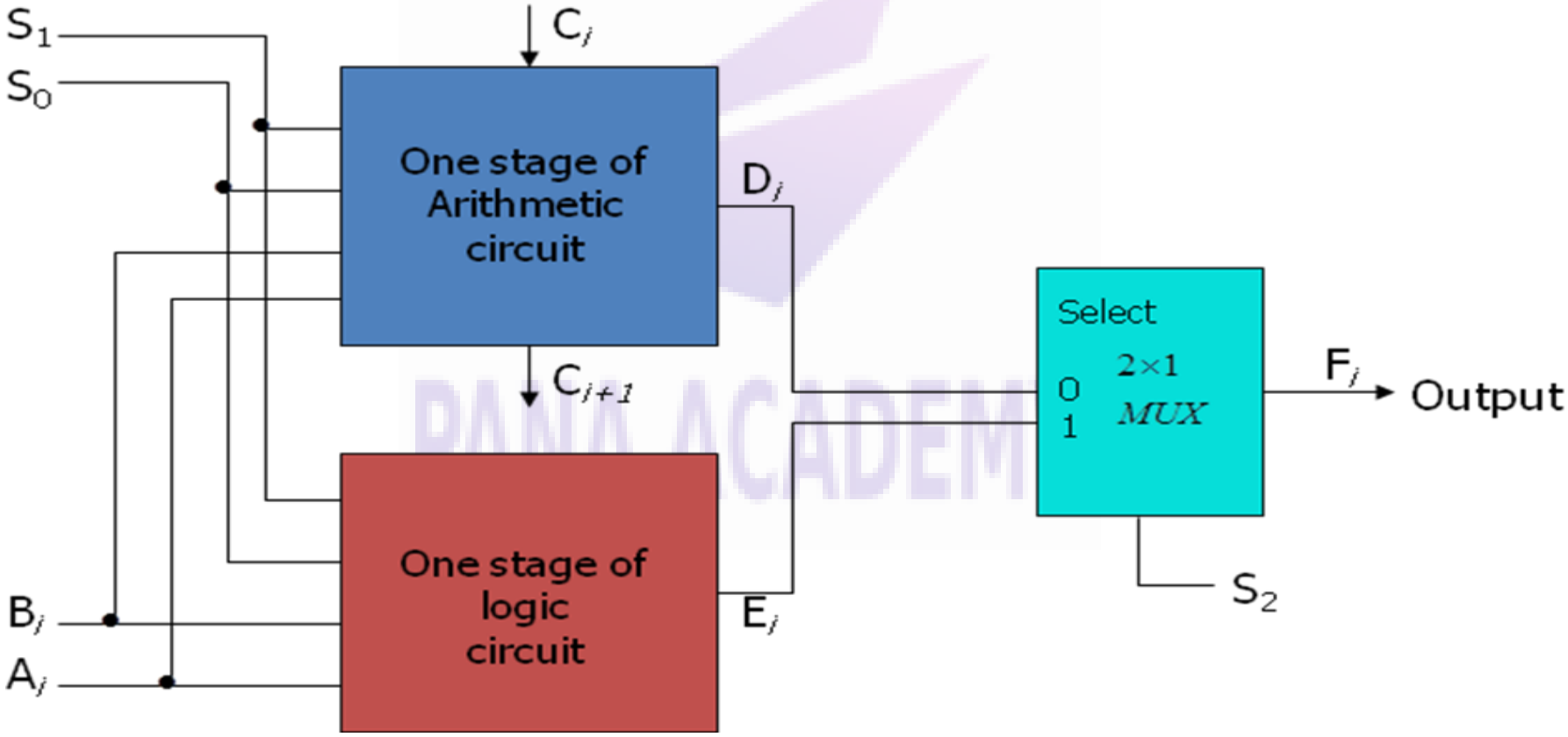
Functional table for logic unit:

S1	S0	output	Microoperation
0	0	Ai && Bi	AND
0	1	Ai ‖ Bi	OR
1	0	Ai XOR Bi	XOR
1	1	Ai’	NOT

Combine these 2 sections to form the ALU

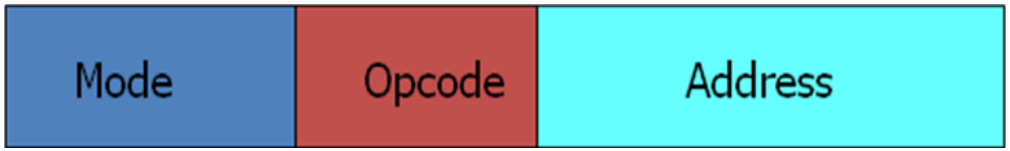
- A combined circuit of ALU where n data input from A are combined with n data input from B to generate the result of an operation at the G output line.
- ALU has a number of selection lines used to determine the operation to be performed.
- The selection lines are decoded with the ALU so that selection lines can specify distinct operations.

- The mode select S_2 differentiate between arithmetic and logical operations.
- The two functions select S_1 and S_0 specify the particular arithmetic and logic operations to be performed.
- With three selection lines, it is possible to specify arithmetic operation with S_2 at 0 and logical operation with S_2 at 1.



Instruction formats

- A computer usually has a variety of Instruction Code Formats.
- Control unit interpret the 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 2^k location directly and specify 2^m different operation.
- The bits of the instruction are divided into groups called fields.
- The most common fields in instruction formats are:
  - An **Operation code** field that specifies the operation to be performed.
  - An **Address field** that designates a memory address or a processor register.
  - A **Mode field** that specifies the way the operand or the effective address is determined.



- 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.

- **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

DISADVANTAGE

The instructions requires too many bits to specify 3 address

- **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 <- Op Y
- Example: SUB X, Y; X <- X - Y

- **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

- **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)

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 instruction
  - Other computers use a single binary for operation & Address
  - The mode field is used to locate the 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).

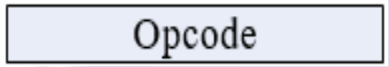
Effective address (EA):

- Is the memory address obtained from the computation dictated by the given addressing
- Is the address of the operand in a computational-type instruction.

The most well known addressing mode are:

- Implied Addressing
- 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

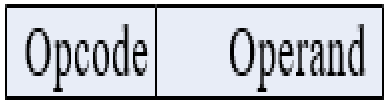
Implied Addressing Mode

- **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.
- Instruction 

- Advantage: no memory reference. Disadvantage: limited operand

Immediate Addressing mode:

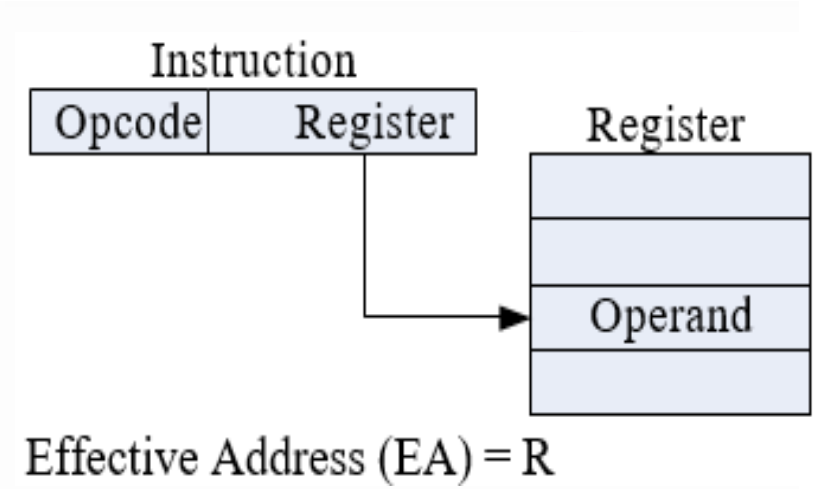
- In this mode the operand is specified in the instruction
- **In other words, an immediate-mode instruction has an operand field rather than an address field.**
- The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction
- These instructions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



- 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:

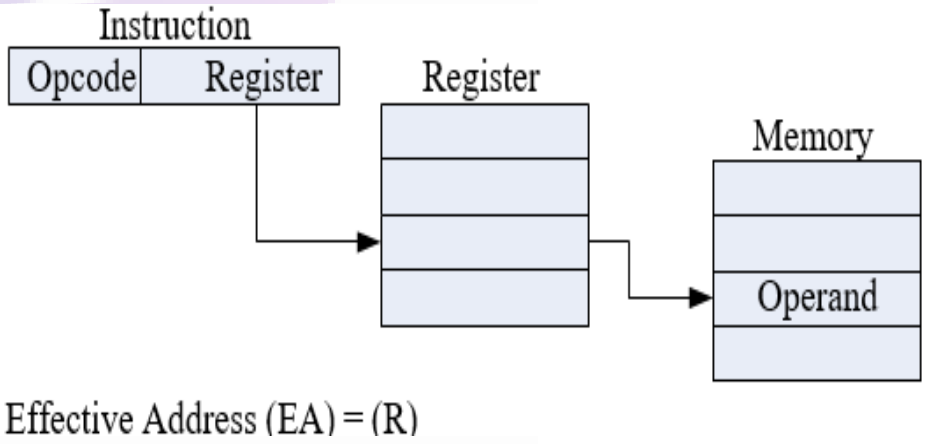
- **In this mode, the operands are in registers that reside within the CPU**.
- The particular register is selected from the register field in the instruction. For example MOV A, B



- **Advantage:** no memory reference.
- **Disadvantage:** limited address space

Register indirect addressing mode:

- Here, the instruction specifies a register in the CPU whose contents give the address of the operand in the memory i.e. 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



- **Advantage:** Large address space.
- **Disadvantage:** Extra memory reference

Auto increment or Auto decrement Addressing Mode:

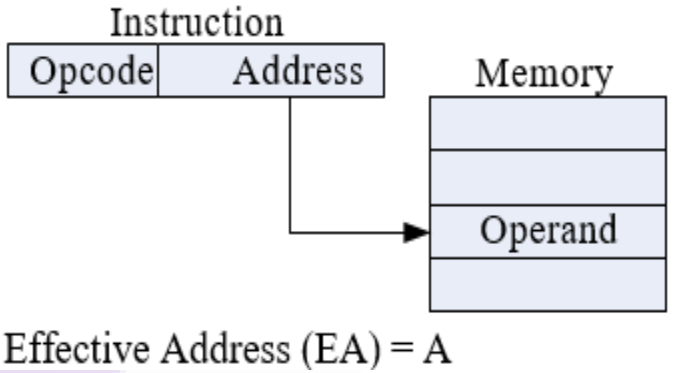
- This is similar to register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- When the address stored in the registers refers to a table of data in memory, it is necessary to increment or decrement the registers after every 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.

Lda – load accumulator

- For example LDA 4000H



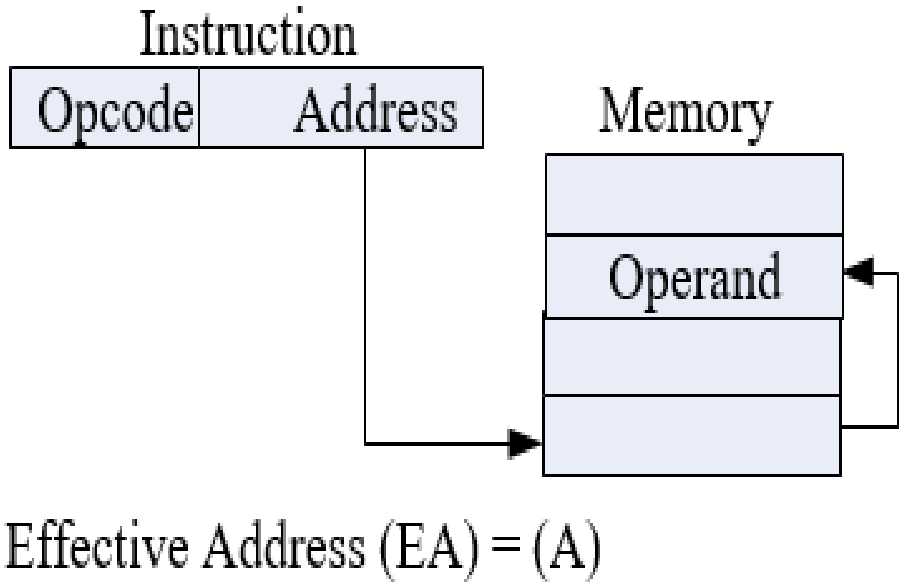
- **Advantage:** Simple.
- **Disadvantage:** limited address field

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.

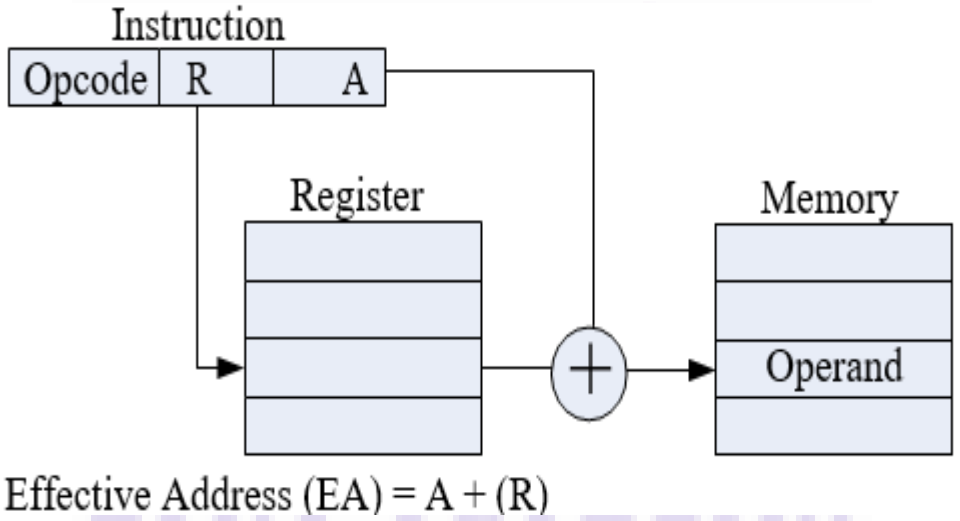
Mov ax,bx

- **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.
- **movl 12(r2),r3**



- **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.

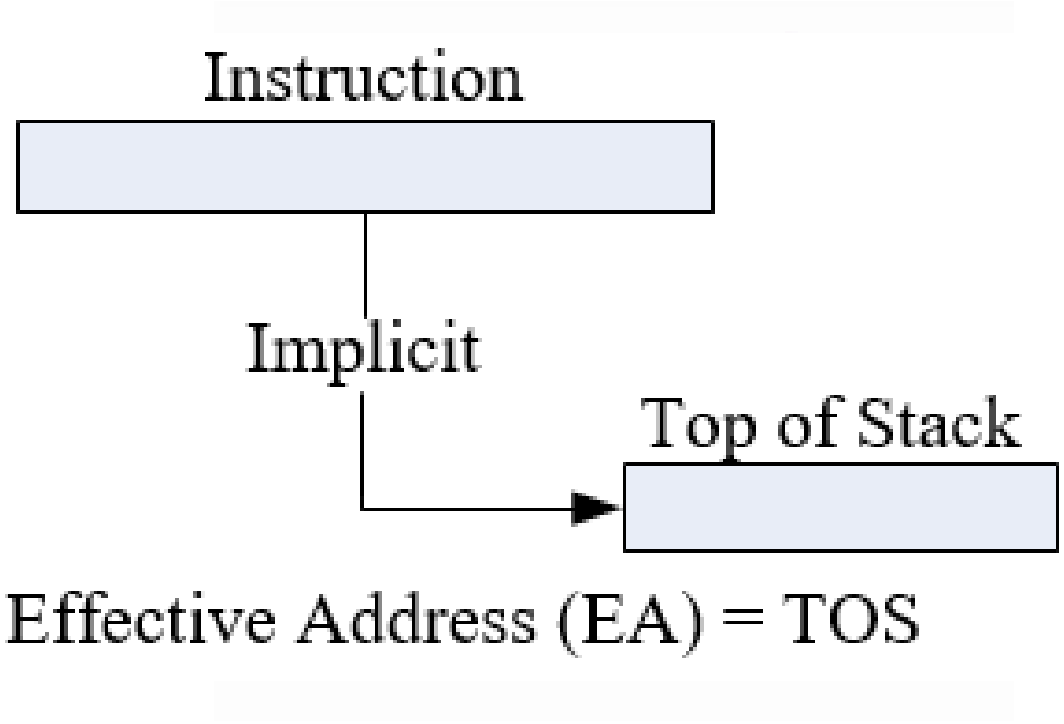
- The index register is a special CPU register that contains an index value.
- Note: If an index-type instruction does not include an address field in its format, the instruction is automatically converted to the register indirect mode of operation.
- Effective Address (EA) = XR + A

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 It is some times referred to as push down list or last in First out (LIFO) queue.
- The stack pointer is maintained in register.



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**

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**

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

Subroutine call and Return

- A subroutine call instruction consists of an operation code together with an address that specifies the beginning of the subroutine.
- The instruction is executed by performing two tasks:
  - The address of the next instruction available in the program counter (the return address) is stored in a temporary location (stack) so the subroutine knows where to return.
  - Control is transferred to the beginning of the subroutine.
- The last instruction of every subroutine, commonly called return from subroutine; transfer the return address from the temporary location into the program counter.
- This results in a transfer of program control to the instruction where address was originally stored in the temporary location.

Interrupt

- The interrupt procedure is, in principle, quite similar to a subroutine call except for three variations:
- The interrupt is usually initiated by an external or internal signal rather than from execution of an instruction.
- The address of the interrupt service program is determined by the hardware rather than from the address field of an instruction.
- An interrupt procedure usually stores all the information necessary to define the state of the CPU rather than storing only the program counter.

CISC & RISC

CISC

- 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 Other are 8085, 8086, 80x86 etc.

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

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.

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

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

Pipelining

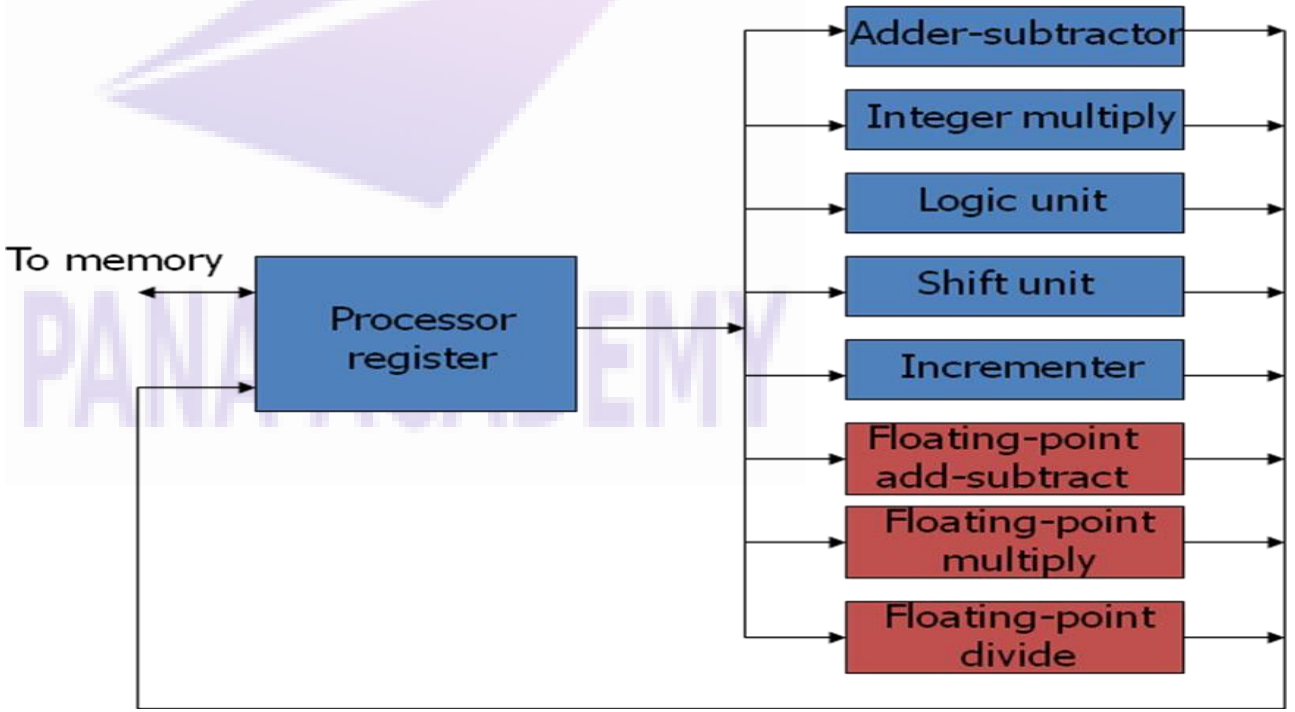
- is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.
- The overlapping of computation is made possible by associating a register with each segment in the pipeline.
- The registers provide isolation between each segment so that each can operate on distinct data simultaneously.
- Perhaps the simplest way of viewing the pipeline structure is to imagine that each segment consists of an input register followed by a combinational circuit.
  - The register holds the data.
  - The combinational circuit performs the suboperation in the particular segment.

Parallel Processing

- Is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system.
- The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, that is, the amount of processing that can be accomplished during a given interval of time.
- The amount of hardware increases with parallel processing, and with it, the cost of the system increases.
- Parallel processing can be viewed from various levels of complexity.
  - At the lowest level, we distinguish between parallel and serial operations by the type of registers e.g. shift registers and registers with parallel load

  - At a higher level, it can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously.

- Figure below shows one possible way of separating the execution unit into eight functional units operating in parallel.
  - A multifunctional organization is usually associated with a complex control unit to coordinate all the activities among the various components.



- There are a variety of ways that parallel processing can be classified.
  - Internal organization of the processors
  - Interconnection structure between processors
  - The flow of information through the system
- M.J. Flynn considers the organization of a computer system by the **number of instructions and data items** that are manipulated simultaneously.
  - Single instruction stream, single data stream (SISD)
  - Single instruction stream, multiple data stream (SIMD)
  - Multiple instruction stream, single data stream (MISD)
  - Multiple instruction stream, multiple data stream (MIMD)

SISD

- Represents the organization of a single computer containing a control unit, a processor unit, and a memory unit.
- Instructions are executed sequentially and the system may or may not have internal parallel processing capabilities.
- Parallel processing may be achieved by means of multiple functional units or by pipeline processing.

SIMD

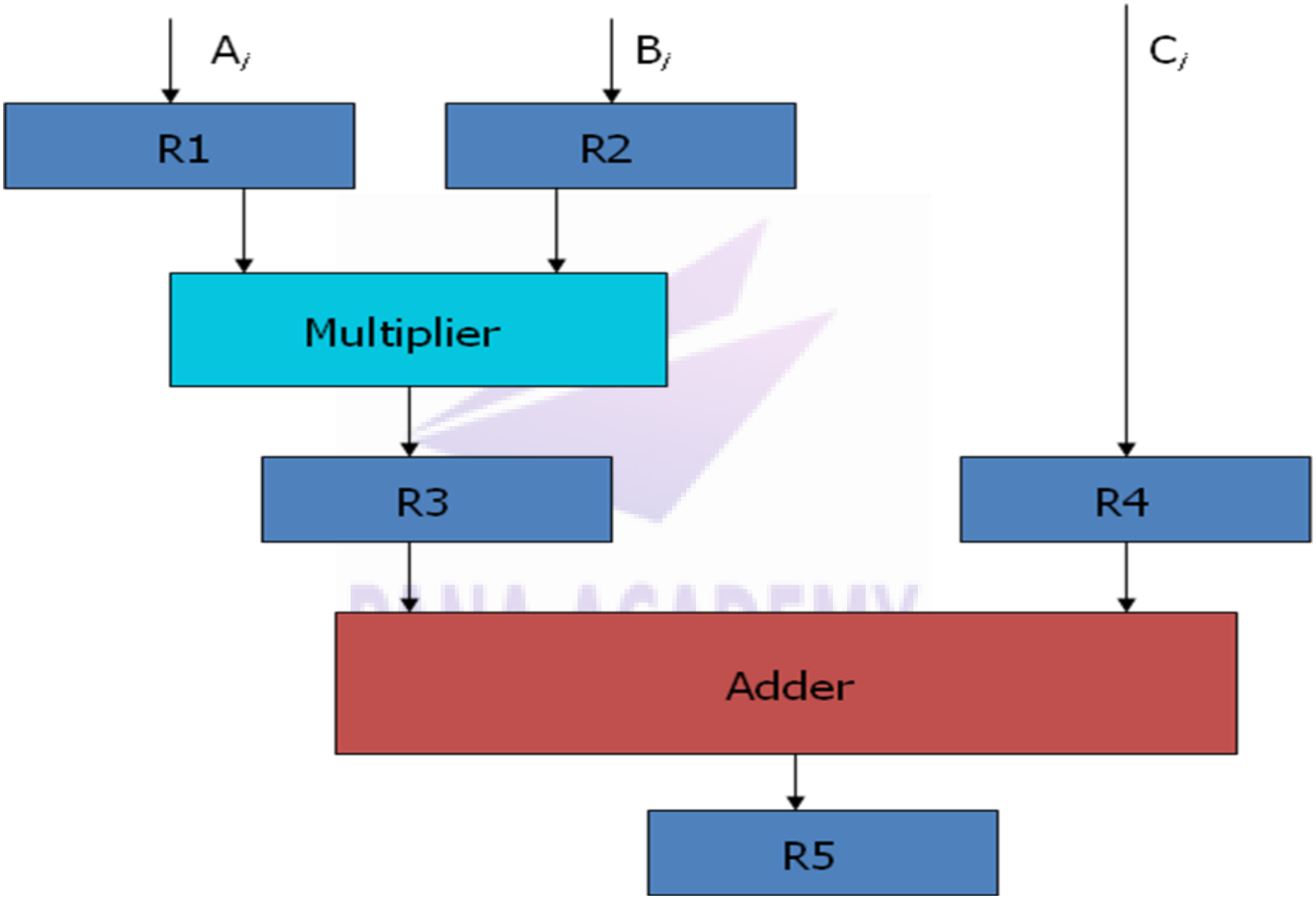
- Represents an organization that includes many processing units under the supervision of a common control unit.
- All processors receive the same instruction from the control unit but operate on different items of data.
- The shared memory unit must contain multiple modules so that it can communicate with all the processors simultaneously.

MISD & MIMD

- MISD structure is only of theoretical interest since no practical system has been constructed using this organization.
- MIMD organization refers to a computer system capable of processing several programs at the same e.g. multiprocessor and multicomputer system
- **Flynn's classification depends on the distinction between the performance of the control unit and the data-processing unit.**
- It emphasizes the behavioral characteristics of the computer system rather than its operational and structural interconnections.
- One type of parallel processing that does not fit Flynn's classification is pipelining.

- A clock is applied to all registers after enough time has elapsed to perform all segment activity.
- The pipeline organization will be demonstrated by means of a simple example:
  - To perform the combined multiply and add operations with a stream of numbers
  - $A_i * B_i + C_i$ for i = 1, 2, 3, …, 7
- Each sub-operation is to be implemented in a segment within a pipeline.
  - R1 <- A_i, R2 <- B_i Input A_i and B_i
  - R3 <- R1 * R2, R4 <- C_i Multiply and input C_i
  - R5 <- R3 + R4 Add C_i to product

- Each segment has one or two registers and a combinational circuit as shown below



- Which of the following allows simultaneous write and read operations?
  a. ROM
  b. EROM
  c. RAM
  d. None of the above
- Which of the following format is used to store data?
  a. Decimal
  b. Octal
  c. BCD
  d. Hexadecimal
- Which of the following memory of the computer is used to speed up the computer processing?
  a. Cache memory
  b. RAM
  c. ROM
  d. None of the above
- Which of the following register can interact with the secondary storage?
  a. PC
  b. MAR
  c. MDR
  d. IR

- In the case of, Zero-address instruction method the operands are stored in _____
  a) Registers
  b) Accumulators
  c) Push down stack
  d) Cache
- The addressing mode which makes use of in-direction pointers is _____
  a) Indirect addressing mode
  b) Index addressing mode
  c) Relative addressing mode
  d) Offset addressing mode
- The addressing mode/s, which uses the PC instead of a general purpose register is _____
  a) Indexed with offset
  b) Relative
  c) Direct
  d) Both Indexed with offset and direct
- The addressing mode, where you directly specify the operand value is _____
  a) Immediate
  b) Direct
  c) Definite
  d) Relative

- _____ addressing mode is most suitable to change the normal sequence of execution of instructions.
  a) Relative
  b) Indirect
  c) Index with Offset
  d) Immediate
- **Which addressing mode execute its instructions within CPU without the necessity of reference memory for operands?**
  a. Implied Mode
  b. Immediate Mode
  c. Direct Mode
  d. Register Mode
- The part of a processor which contains hardware necessary to perform all the operations required by a computer:
  a) Data path
  b) Controller
  c) Registers
  d) Cache

- If the control signals are generated by combinational logic, then they are generated by a type of _____ controlled unit.
  a) Micro programmed
  b) Software
  c) Logic
  d) Hardwired
- Which is the simplest method of implementing hardwired control unit?
  a) State Table Method
  b) Delay Element Method
  c) Sequence Counter Method
  d) Using Circuits
- A set of microinstructions for a single machine instruction is called _____
  a) Program
  b) Command
  c) Micro program
  d) Micro command

- A decoder is required in case of a _____
  a) **Vertical Microinstruction**
  b) Horizontal Microinstruction
  c) Multilevel Microinstruction
  d) All types of microinstructions
- What does the clock speed of a CPU measure?
  a. The number of CPU cores
  b. The processing speed of the CPU
  c. The amount of RAM in CPU
  d. The size of the CPU cache
- What is the purpose of CPU clock?
  a. To keep track of time and date
  b. To regulate the temperature of the CPU
  b. To synchronize and control the execution of instructions
  d. To display images on screen

- Which of the following is not a type of CPU cache?
  a. L1 cache
  b. L2 cache
  c. RAM cache
  d. L3 cache
- What is the purpose of pipelining in CPU design?
  a. To prevent overheating of the CPU
  b. To reduce power consumption
  c. To improve instruction execution throughput
  d. To connect to the internet
- What is a CPU core?
  a. A physical processor unit within CPU
  b. A type of computer monitor
  c. A network interface card
  d. A software application

- What is the purpose of CPU's stack pointer register?
  a. To store program instruction
  b. To point to the top of the call stack
  c. To control the CPU's clock speed
  d. To display images on screen
- The ALU gives the output of the operations and the output is stored in the _____
  a) Memory Devices
  b) Registers
  c) Flags
  d) Output Unit
- The process of division on memory spaces is called _____
  a) Paging
  b) Segmentation
  c) Bifurcation
  d) Dynamic Division
- Number of bits in ALU is _____
  a) 4
  b) 8
  c) 16
  d) 2

- **What kind of PSW flags remain unaffected by the data transfer instructions ?**
  a. Auxillary Carry Flags
  b. Overflow Flags
  c. Parity Flags
  d. All of the above
- The computer architecture aimed at reducing the time of execution of instructions is _____
  a) CISC
  b) RISC
  c) ISA
  d) ANNA
- The Sun micro systems processors usually follow _____ architecture.
  a) CISC
  b) ISA
  c) ULTRA SPARC
  d) RISC
- The iconic feature of the RISC machine among the following is _____
  a) Reduced number of addressing modes
  b) Increased memory size
  c) Having a branch delay slot
  d) All of the mentioned

- Both the CISC and RISC architectures have been developed to reduce the _____
  a) Cost
  b) Time delay
  c) Semantic gap
  d) All of the mentioned
- In CISC architecture most of the complex instructions are stored in _____
  a) Register
  b) Diodes
  c) CMOS
  d) Transistors
- Pipe-lining is a unique feature of _____
  a) RISC
  b) CISC
  c) ISA
  d) IANA
- A pipeline is similar to which of the following?
  a) a gas line
  b) house pipeline
  c) both a and b
  d) an automobile assembly line

- A processor performing fetching or decoding of instructions during the execution of another instruction is commonly known as?

  a. Super-scaling

  b. Parallel Computation

  c. Pipe-lining

  d. None of these
- A term for simultaneous access to a resource physical or logical

  a. Multiprogramming

  b. Multitasking

  c. Threads

  d. Concurrency
- A parallelism based on increasing processor word size

  a. Increasing

  b. Count based

  c. Bit based

  d. Bit level

- The fetch and execution cycles are interleaved with the help of _____
  a) Modification in processor architecture
  b) Clock
  c) Special unit
  d) Control unit
- Each stage in pipelining should be completed within _____ cycle.
  a) 1
  b) 2
  c) 3
  d) 4
- If a unit completes its task before the allotted time period, then _____
  a) It'll perform some other task in the remaining time
  b) Its time gets reallocated to a different task
  c) It'll remain idle for the remaining time
  d) None of the mentioned
- To increase the speed of memory access in pipelining, we make use of _____
  a) Special memory locations
  b) Special purpose registers
  c) Cache
  d) Buffers