RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Department of Computer Science & Engineering

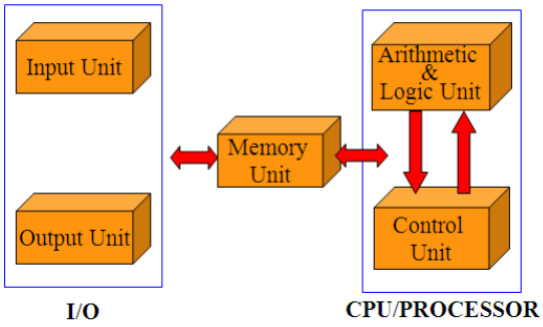
S4 CSE Computer Organization & Architecture

Question Bank

Prepared By: Dr. Preetha K G, Associate Professor, Department of Computer Science

MODULE 1

1. **Explain the functional units of computer**



- Input Unit: Data/ instructions are fed to a computer through input unit
  Examples: keyboard, mouse. Scanner, joystick
- Arithmetic & Logic Unit: ALU consist of necessary logic circuits like adder, comparator etc., to perform arithmetic and logic operations such as addition, multiplication, comparison of two numbers etc.
- Control Unit: Control unit co-ordinates activities of all units by issuing control signals. Control signals issued by control unit govern the data transfers and then appropriate operations take place. Control unit interprets or decides the operation/action to be performed.

- ALU &Control Unit together called Central Processing Unit (CPU). CPU is the brain of a computer system.
- **Memory Unit**: Memory unit stores the program instructions (Code), data and results of computations etc

  Examples: ROM,RAM
- **Output Unit**: Computer after computation returns the computed results, error messages, etc. via output unit.

  Examples: Monitor, printer, speaker

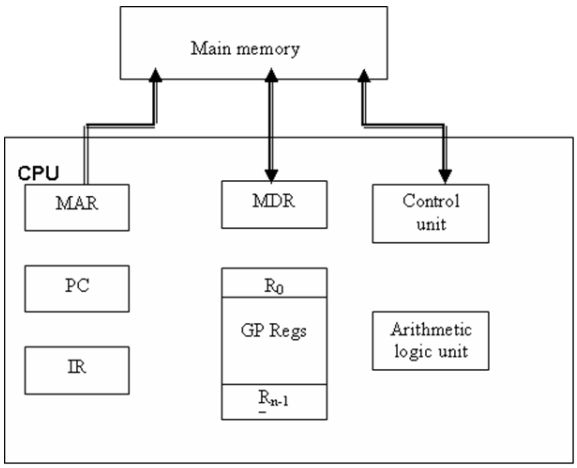
2. Describe the operational concepts of a computer.

- A set of instructions called a program reside in the main memory of computer. Data is also stored in the memory
- The CPU fetches those instructions sequentially one-by-one from the main memory, decodes them and performs the specified operation on associated data operands in ALU.
- Processed data and results will be stored in the memory and displayed on an output unit.
- All activities pertaining to processing and data movement inside the computer machine are governed by control unit.

3. What is a register? Describe the functions of various registers in CPU with a neat diagram./ Explain the internal architecture of a processor.

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

In addition to ALU CPU also contains CPU also contains Instruction Register (IR), the Program Counter(PC), the general-purpose registers, the Memory Address Register(MAR) and Memory Data Register(MDR).



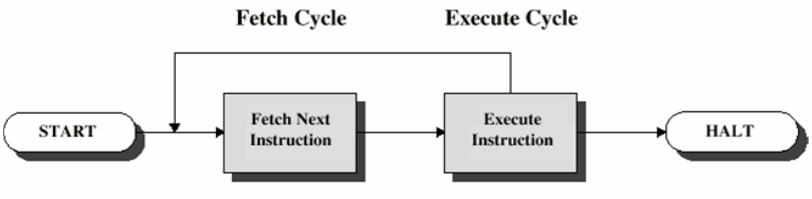
Functions

- **Instruction Register(IR**): contains the instruction that is being executed. Its output is available to the control circuits, that generate the timing signals for control of the actual processing circuits needed to execute the instruction.
- **Program Counter(PC):** It contains the memory address of the next instruction to be fetched and executed
- **Memory Address Register (MAR):** holds the address of the memory location to or from which data is to be transferred.
- **Memory Data Register(MDR):** contains the data to be written into or read-out of the addressed memory location.
- **General- purpose Registers:** are used for holding data, intermediate results of operations. They are also known as scratch-pad registers.

4. Illustrate the basic operational concepts in transferring data between main memory and processor.

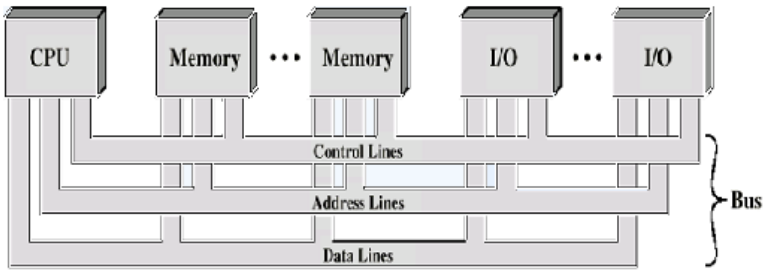
- Programs reside in the memory & usually get these through the input unit.
- Execution of the program starts when the PC is set to point at the first instruction of the program.
- Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.
- The addressed word is read out of the memory and loaded into the MDR.
- Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.

- If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.
- An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.
- When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.
- After one or two such repeated cycles, the ALU can perform the desired operation.
- If the result of this operation is to be stored in the memory, the result is sent to MDR.
- Address of location where the result is stored is sent to MAR & a write cycle is initiated.
- The contents of PC are incremented so that PC points to the next instruction that is to be executed.



5. List different types of buses used to interconnect various functional units in a computer.

- **Data Bus**: It is used for transmission of data. The number of data lines corresponds to the number of bits in a word.
- **Address Bus**: It carries the address of the main memory location from where the data can be accessed.
- **Control Bus**: It is used to indicate the direction of data transfer and to coordinate the timing of events during the transfer.



6. **Explain zero, one, two and three address instruction with an example for each.**

Three -address instruction format

This instruction format consists of three addresses along with an operation field. The three addresses include the address of the first operand, address of the second operand, address to store the result.

Format: Operation code Source1,source2, destination

Example: Add A,B,C

[C]← [A]+[B]

Two -address instruction format

This instruction format consists of two addresses along with an operation field. The two addresses include the address of the first operand, address of the second operand; the result is stored in one of the operand address.

Example: Add A, B

B← [A]+[B]

One -address instruction format

This instruction format consists of one address along with an operation field. The address is that of the first operand. The second operand and the result are stored in a CPU register called **accumulator**. A machine has only one accumulator; it need not be explicitly mentioned in the instruction.

Example: Add A

Zero -address instruction format

A **stack** is included in the CPU for performing arithmetic and logic instructions with no addresses. The operands are pushed onto the stack from memory and ALU operations are implicitly performed on the top elements of the stack.

Example: Add

Top of stack = top of stack + second top of stack

7. **What is an addressing mode? List different addressing modes used in an instruction execution.**

   The address generated by the CPU in-order to access the operand in the memory is termed as an effective address. The methods used to provide an access path to operands in memory and CPU registers is addressing mode.

   Various addressing modes are

   - Immediate
   - Register
   - Direct(Absolute)
   - Indirect
   - Index (Displacement)
   - Base with index
   - Base with index and offset
   - Relative
   - Auto increment
   - Auto decrement

8. **Describe auto increment addressing mode with the help of an example.**

   After accessing the operand the content of this register is automatically incremented to point the next item in the list.

   Increment R_i

   Add (R1)+, R2

9. **Identify the following addressing modes.**

   Move #500,R0-Immediate Addressing mode

   Add 4(R0),R1-Index addressing mode

   Add #16,R0-Immediate addressing mode

   Add 12(R0),R3-Index addressing mode

   Move R1,Sum1-Combination of register and direct

   Add [R2],[R0]- Indirect addressing mode

   Add (R2)+,R0-Autoincrement mode

   Add Loc,R2- Direct addressing mode

   Add (R3),R2-Indirect addressing mode

Add (R5),R1-Indirect addressing mode

Move R5, R4-Register Addressing mode

Move Loc1,Loc2-Direct addressing mode

Load 20(R1),R5-Index addressing mode

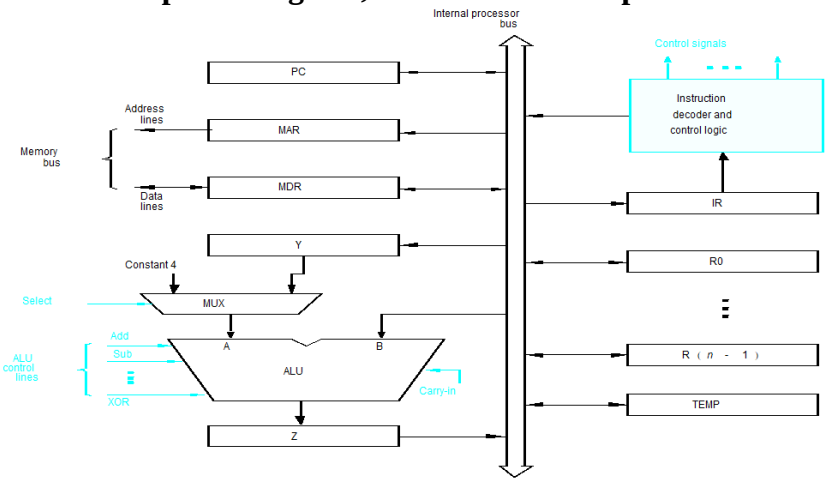
Store R5,30(R1,R2)-Base with index and offset

Sub (R1)+,R5-Autoincrement mode

Add (-R1),R5-Autodecrement mode

Jump 10(PC)- Relative addressing mode

10. With the help of a diagram, describe the data-path inside the processor.



- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

$$IR \leftarrow [[PC]]$$

- Assuming that the memory is byte addressable(4 byte long), increment the contents of the PC by 4 (fetch phase).

$$PC \leftarrow [PC] + 4$$

  - PC Keeps track of execution of a program
  - Contains the memory address of the next instruction to be fetched and executed.

- Constant 4 is used by the processor to increment the contents of PC.

- Carry out the actions specified by the instruction in the IR (execution phase).
- The data and address lines of external memory connected to the processor bus via MDR and MAR
- Register MDR has 2 inputs and 2 outputs.:
  - Contains data to be written into or read out of the addressed location.
  - Data can be loaded into MDR either from memory bus or from internal processor bus.
- MAR Holds the address of the location to be accessed.
  - I/P of MAR is connected to Internal bus and an O/p to external bus.
- The instruction decoder and control logic circuit is responsible for implementing the actions specified by the instruction loaded in IR
  - The decoder generate the control signals needed to select the registers
  - The registers, ALU and interconnecting bus are collectively referred to as the data path.
- MUX
  - Select either the output of the register Y or a constant value 4 to be provided as input A of the ALU.

11. Give the control sequence for execution of instruction Add[R3],R1 using a single bus organization

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4,Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

12. Write down the sequence of actions needed to fetch and execute the instruction

LOAD 10(R2),R1.

1. Pcout, MARin, Read,Select4,Add, Zin
2. Zout, Pcin, Yin,WMFC
3. MDRout, IRin
4. R2out,MARin, Read
5. WMFC, MDRout
6. IR10out,Yin
7. Select Y, Add, Zin
8. Zout, R1 in, End

13. Write down the sequence of actions needed to fetch and execute the instruction

STORE R1, 10(R2)

1. Pcout, MARin, Read,Select4,Add, Zin
2. Zout, Pcin, Yin,WMFC
3. MDRout, IRin
4. R2out,MARin, Read
5. WMFC, MDRout
6. IR10out,Yin
7. Select Y, Add, Zin
8. Zout, MARin
9. R1 out, MDRin, Write
10. WMFC, End

14. Give the sequence of control steps required to perform the operation ADD NUM, R1

1. Pcout, MARin, Read,Select4,Add, Zin
2. Zout, Pcin, Yin,WMFC
3. MDRout, IRin
4. IRNUMout,Yin,
5. SelectY, R1out, Add, Zin
6. Zout, R1in,End

15. Give the sequence of control steps to perform the operation ADD (NUM),R1

1. Pcout, MARin, Read,Select4,Add, Zin

2.Zout, Pcin, Yin,WMFC

3.MDRout, IRin

4. IRNUMout,MARin, Read

5. WMFC, MDRout,MARin,Read

6. WMFC, MDRout,Yin

7. Select Y, R1out, Add,Zin

8. Zout,R1in,END

16. Give the sequence of control steps required to perform the operation ADD R1, NUM

1. PCout, MARin, Read,Select4,Add, Zin

2.Zout, Pcin, Yin,WMFC

3.MDRout, IRin

4. IRNUMout, MARin, Read

5. WMFC, MDRout, Yin,

6. Select Y, R1out,Add, Zin

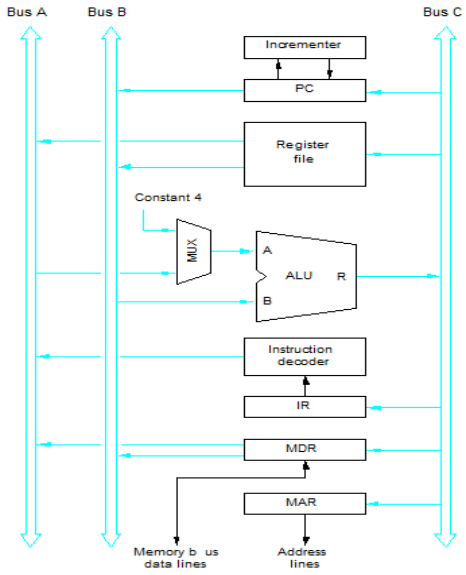
7. Zout, MDRin, Write

8. WMFC, END

17. Illustrate the advantages of using multiple bus organization over single bus organization.

- Multiple Bus Organization Improves Efficiency.
- Additional Buses Allow Expansion.
- More Buses Means More Compatibility
- Processor speed is high in multiple bus organization

18. Draw the diagram of a multi-bus organization with 3 buses. Write the control sequence for the instruction Add R4, R5, R6 for the above mentioned multi-bus organization.



Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

19. How is a branching instruction executed?

A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction. The offset X is usually the difference between the branch target address and the address immediately following the branch instruction

Control Sequence of a branch instruction

1. PC out, MKAR in, Read, Clear Y, Set carry in to ALU,ADD,Zin
2. Z out, PC in, WMFC
3. MDR out, IR in
4. PC out, Y in
5. Offset field of IR out, ADD, Zin
6. Zout, PC in ,End

MODULE 2

20. List the components of register transfer logic.

The set of registers in the system and their functions: A register also encompasses all type of registers including shift registers, counters and memory units.

The binary-coded information stored in the registers: The binary information stored in registers may be binary numbers, binary coded decimal numbers, alphanumeric characters, control information or any other binary coded information.

The operations performed on the information stored in the registers: The operations performed on data stored in registers are called micro operations. Examples are shift, count, add, clear and load

The control functions that initiate the sequence of operations: The control functions that initiate the sequence of operations consists of timing signals that sequence the operations one at a time.

21. What is microoperation?

Micro-operations: operations executed on data stored in one or more registers. For any function of the computer, a sequence of micro-operations is used to describe it

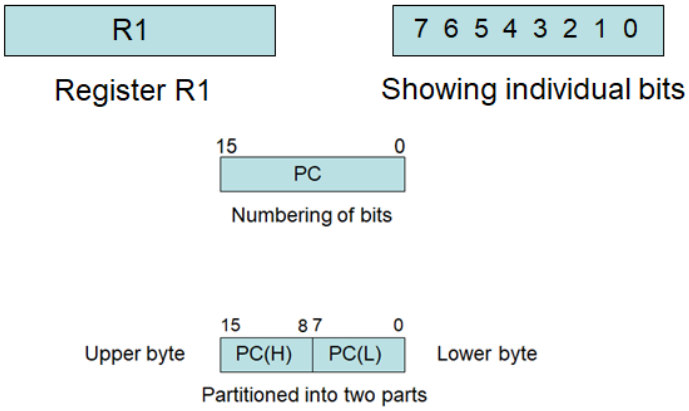
Example: Shift, count, clear, add & load

22. What are the different types of micro operations?

a. Register transfer microoperations
b. Arithmetic microoperations (on numeric data stored in the registers)
c. Logic microoperations (bit manipulations on non-numeric data)
d. Shift microoperations

23. How do you represent register in resister transfer logic.

- Rectangular box with name of the register inside,
- The individual cells is assigned a letter with a subscript number,
- The numbering of cells from right to left can be marked on top of the box,
- 16 bit register is partitioned into 2 parts , bits 1 to 8 are assigned the letter L(for low) and bits 9 to 16 are assigned the letter H(for high)

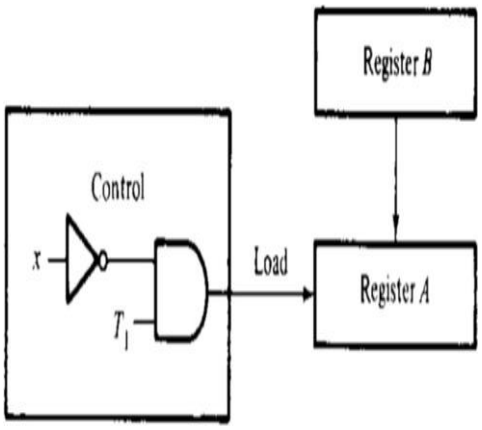


24. Explain the register transfer micro operation with the help of an example.

- Information transfer from one register to another is described by a *replacement operator:* R2 ← R1
- This statement denotes a transfer of the content of register R1 into register R2
- The transfer happens in one clock cycle
- The content of the R1 (source) does not change
- The content of the R2 (destination) will be lost and replaced by the new data transferred from R1
- We are assuming that the circuits are available from the outputs of the source register to the inputs of the destination register, and that the destination register has a parallel load capability

25. With a neat diagram explain how the the conditional transfer operation is implemented.

- Conditional transfer occurs only under a control condition
- Representation of a (conditional) transfer
  - X'T1: A ← B
- A binary condition (X'T1 equals to 0 or 1) determines when the transfer occurs
- The content of B is transferred into A only if if x is 0 and T_1 is 1`



26. Illustrate the basic symbols used in register transfer operation with examples.

Basic Symbols for Register Transfers		
Symbol	Description	Examples
Letters & numerals	Denotes a register	MAR, R2
Parenthesis ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two microoperations	R2 ← R1, R1 ← R2
Colon:	Terminates a control function	P:
Square Brackets [ ]	Specifies an address for memory transfer	MBR ← M [MAR]

27. List all possible arithmetic micro operations with an example.

- Addition Microoperation:R3 ←R1+R2
- Subtraction Microoperation: R3 ←R1-R2 OR R3 ←R1+R2'+1
- One's Complement Microoperation:R2 ←R2'
- Two's Complement Microoperation:R2 ←R2'+1
- Increment Microoperation:R2 ←R2+1
- Decrement Microoperation:R2 ←R2-1

28. Explain logical, circular and arithmetic shift operations with example.

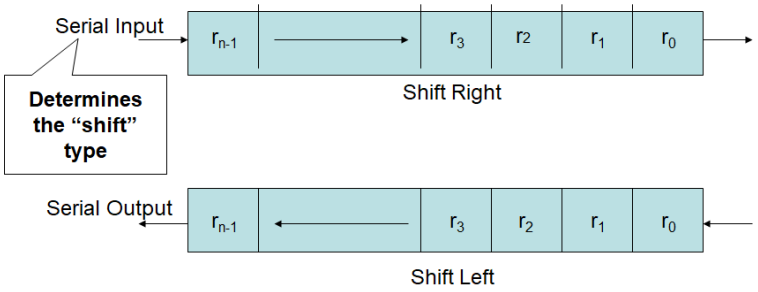
Logical Shift

- Transfers 0 through the serial input
- Logical Shift Right: R1 ← shr R1

The same

- Logical Shift Left: R2 ← shl R2

The same



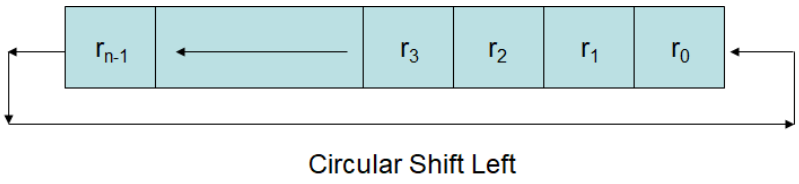
Circular Shift

- Circulates the bits of the register around the two ends without loss of information
- Circular Shift Right: R1 ← cir R1

The same

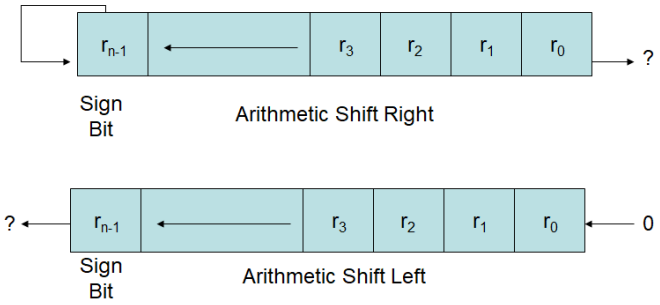
- Circular Shift Left: R2 ← cil R2

The same



Arithmetic Shift

It is also called signed shift**.** When shifting right with an **arithmetic right shift**, the least-significant bit is lost and the most-significant bit is *copied*.



29. Describe various processor organization methods.

Different Processor Organization are:-

a. Bus Organization

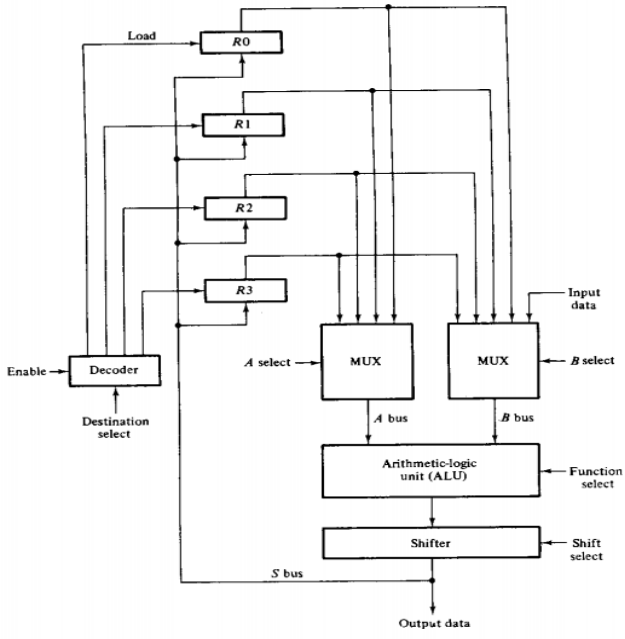
b. Scratchpad Memory

c. Processor with 2-port memory

d. Accumulator Register

Bus Organization

When a large number of registers are included in a processor, it is the most efficient way to connect them through common buses.

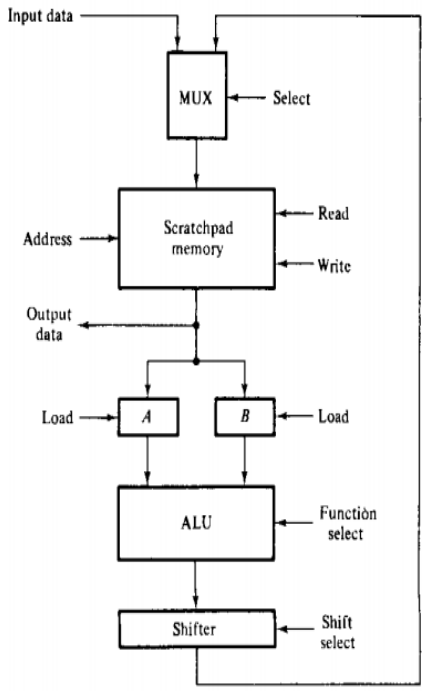


- Register connected to two MUX to form input buses A and B.
- The selection lines of each MUX select one register for the particular bus.
- The A and B buses are applied to a common ALU.

- The function selected in the ALU determines the particular operation that is to be performed.
- The shift micro-operations are implemented in the shifter.
- The result of the micro-operation goes through the output bus S into the inputs of all registers.
- The destination register that receives the information from the output bus is selected by a decoder.
- When enabled, this decoder activates one of the register load inputs to provide a transfer path between the data on the S bus and the inputs of the selected destination register.
- The output bus S provides the terminals for transferring data to an external destination.
- One input of MUX A or B can receive data from the outside
- The control unit that supervises the processor bus system directs the information flow through the ALU by selecting the various components in the unit.

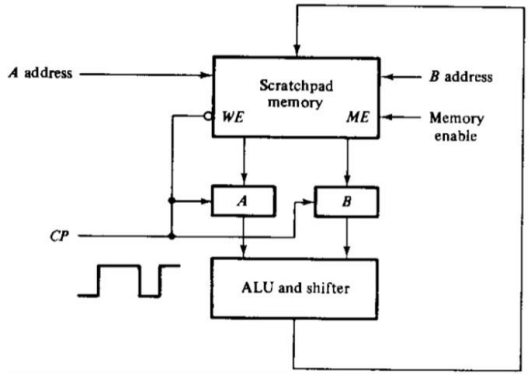
Scratch pad memory

- The register in a processor unit can be enclosed in a small memory unit.
- When included in a processor unit,a small memory is sometime called a scratchpad memory.
- The use of a small memory is a cheaper alternative to collecting processor registers through a bus system.
- The difference between the two system is the manner in which information is selected for transfer into the ALU



- Resource register is selected from memory and loaded into register A.
- A Second source register is selected from memory and loaded into register B.
- The information in A and B is manipulated in the ALU and shifter.
- Result of the operation is transferred to a memory register specifying its word address and activating the memory-write input control.
- Assume that the memory has eight words, so that an address must be specified with three bits.

Processor with 2- port memory



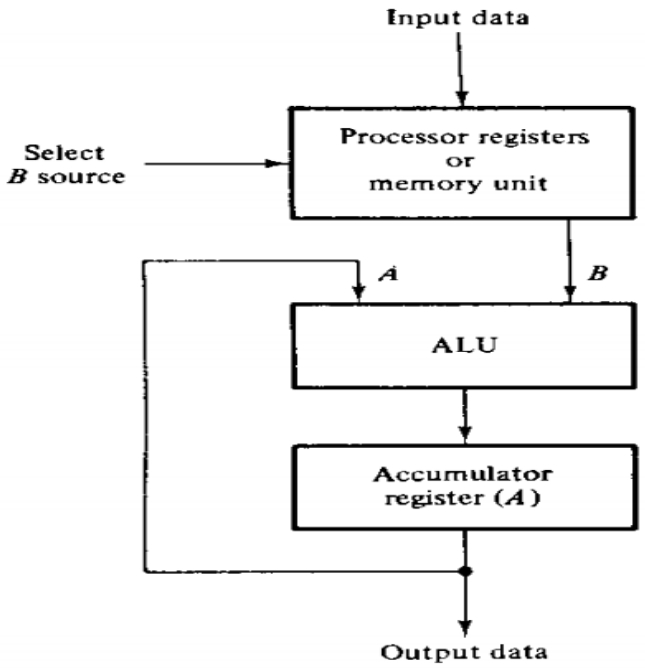
- Some processor employ a 2 port memory in order to **overcome the delay caused when reading two source registers**. A 2-port has two separate address lines to select two

words of memory simultaneously. The organization of a processor unit with a 2-port scratchpad memory is shown in figure above.

- In this way the two source register can be read at the same time.
- Memory has 2 sets of addresses. One for port A and the other for Port B.Data from any word in the memory are read into the register A by specifying an A address . Likewise data from any word in memory are read into B register by specifying a B address. The same address can be applied to the A address and B address.
- When enabled by the Memory Enable (ME) input, new data can be written into the word specified byb the B address. Thus the A and B addresses specify two source registers simultaneously and the B address always specifies the destination register.
- A and B registers are latches that accepts new information as the clock pulse CP is in 1 -state . When CP goes to 0, the latches are disabled and hold the information that was stored when CP was 1.

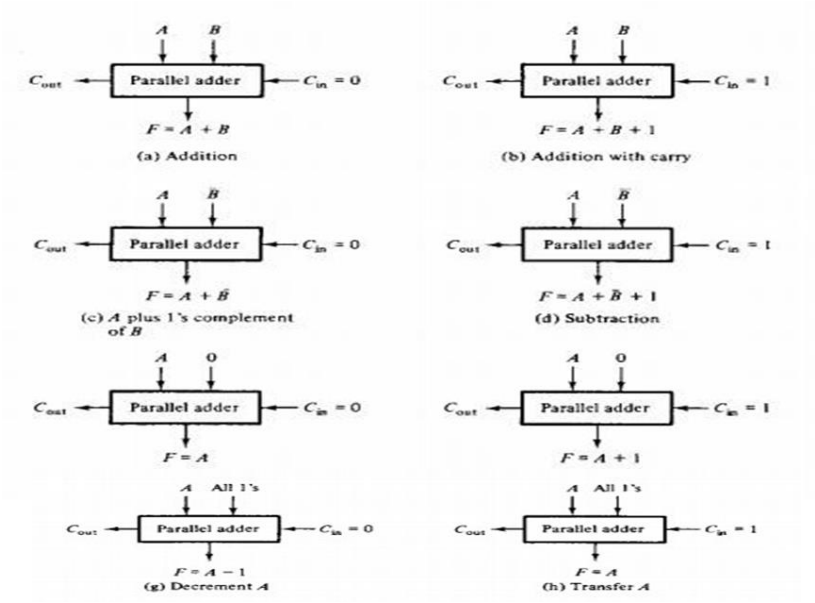
Processor with Accumulator register

Accumulator is the intermediate storage of arithmetic and logic data in a computer's CPU

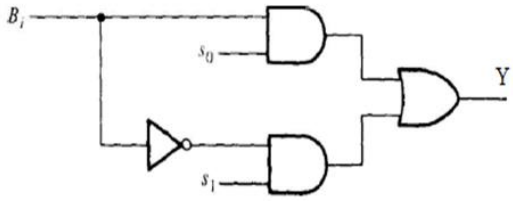


30. Describe the design of Arithmetic circuit

8 arithmetic operations acan be described in the following figure

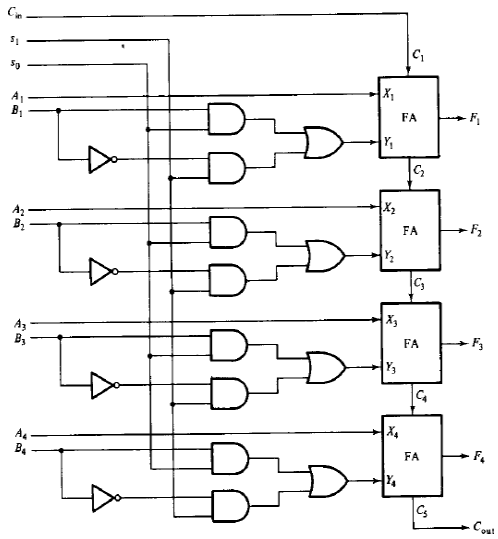


By changing the B input and C_{in} we get 8 operaton. So the input B is applied in four different form by using following circuit.



S1	S0	Y
0	0	0
0	1	B
1	0	B'
1	1	1

Y=S0Bi+S1Bi'



Function select			Y equals	Output equals	Function
S_1	S_0	C_{in}			
0	0	0	0	$F = A$	Transfer A
0	0	1	0	$F = A + 1$	Increment A
0	1	0	B	$F = A + B$	Add B to A
0	1	1	B	$F = A + B + 1$	Add B to A plus 1
1	0	0	$\bar{B}$	$F = A + \bar{B}$	Add 1's complement of B to A
1	0	1	$\bar{B}$	$F = A + \bar{B} + 1$	Add 2's complement of B to A
1	1	0	All 1's	$F = A - 1$	Decrement A
1	1	1	All 1's	$F = A$	Transfer A

31. Explain the design of logic circuit

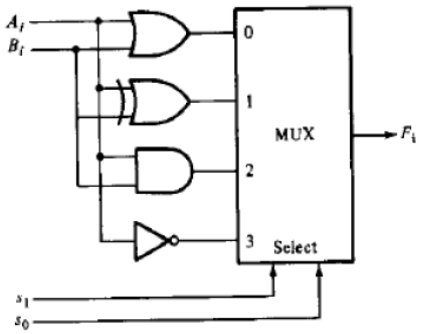
16 logic operations can be generated in one circuit and selected by means of four selection lines.

Can be obtained by means of AND, OR, and NOT (complement) operations

For three operations, we need two selection variables

But two selection lines can select among four logic operations, so we choose also the exclusive-OR (XOR) function for the logic circuit to be designed

The diagram(next slide) shows one typical stage designated by subscript i. The circuit must be repeated n times for an n-bit logic circuit



s_1	s_0	Output	Operation
0	0	$F_i = A_i + B_i$	OR
0	1	$F_i = A_i \oplus B_i$	XOR
1	0	$F_i = A_i B_i$	AND
1	1	$F_i = A_i'$	NOT

32. Give the function table and design of ALU

Function table

Selection				Output	Function
s_2	s_1	s_0	C_{in}		
0	0	0	0	$F = A$	Transfer A
0	0	0	1	$F = A + 1$	Increment A
0	0	1	0	$F = A + B$	Addition
0	0	1	1	$F = A + B + 1$	Add with carry
0	1	0	0	$F = A - B - 1$	Subtract with borrow
0	1	0	1	$F = A - B$	Subtraction
0	1	1	0	$F = A - 1$	Decrement A
0	1	1	1	$F = A$	Transfer A
1	0	0	X	$F = A \vee B$	OR
1	0	1	X	$F = A \oplus B$	XOR
1	1	0	X	$F = A \wedge B$	AND
1	1	1	X	$F = \overline{A}$	Complement A

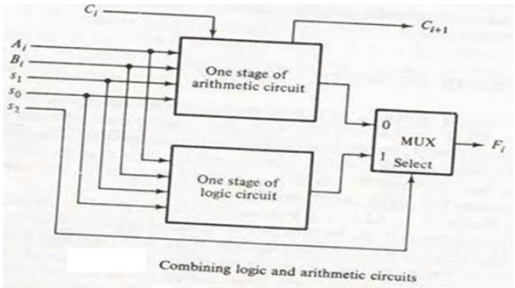
Design steps

1. Design the arithmetic section independent of the logic section.

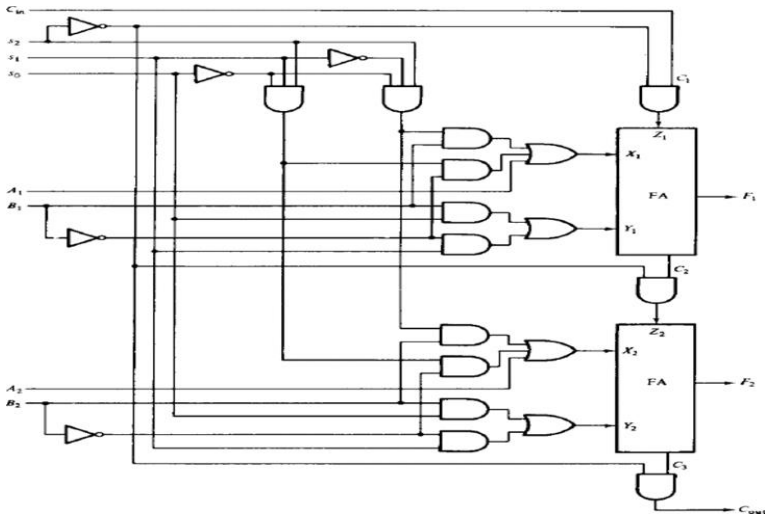
2. Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0.

3. Modify the arithmetic circuit to obtain the required logic operations.

One bit ALU design



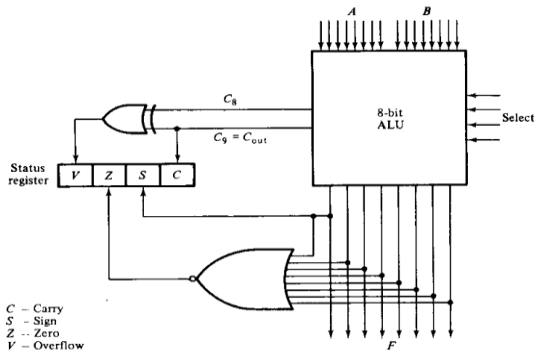
Combining logic and arithmetic circuits



33. Implement the status register. Give its design and function table.

Status register is a 4 bit register. The four bits are C (carry), Z (zero),S (sign) and V (overflow).These bits are set or cleared as a result of an operation performed in the ALU.

- Bit C is set if the output carry of an ALU is 1.
- Bit S is set to 1 if the highest order bit of the result in the output of the ALU is 1.
- Bit Z is set to 1 if the output of the ALU contains all O's.
- Bit V is set if the exclusive —OR of carries C8 and C9 is 1, and cleared otherwise. This is the condition for overflow when the numbers are in signed 2's complement representation. For an 8 bit ALU, V is set if the result is greater than 127 or less than -128.



Relation	Condition of status bits	Boolean function
$A > B$	$C = 1$ and $Z = 0$	CZ'
$A \geq B$	$C = 1$	C
$A < B$	$C = 0$	C'
$A \leq B$	$C = 0$ or $Z = 1$	$C' + Z$
$A = B$	$Z = 1$	Z
$A \neq B$	$Z = 0$	Z'

34. Explain the Design of a logic shifter circuit with diagram and function table.

Shifter may function in four different ways.

1. The shifter may transfer the information directly without a shift.
2. The shifter may shift the information to the right.
3. The shifter may shift the information to the left.
4. In some cases no transfer is made from ALU to the output bus.

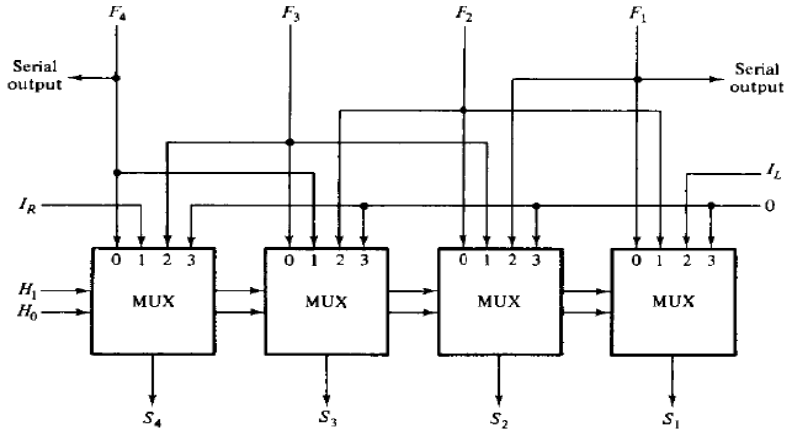
Shifter operation can be selected by two variables H1 H0

• If H1 H0 = 0 0 No shift is executed and the signal from F go directly to S lines

• If H1 H0 = 0 1 Shift Right is executed

• If H1 H0 = 1 0 Shift Left is executed

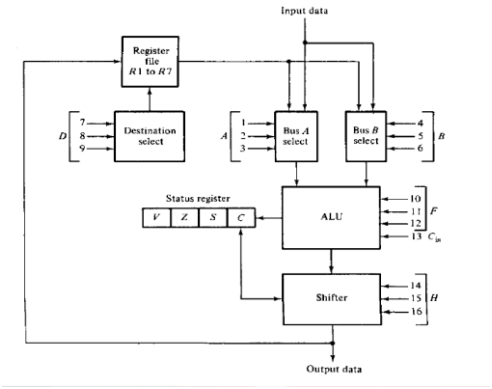
• If H1 H0 = 1 1 No operations



H1	H2	Operation	Function
0	0	S<--F	Transfer F to S
0	1	S←Shr F	Shift right F into S
1	0	S← Shl F	Shift left F into S
1	1	S←0	Transfer 0 into S

35. Design the complete Processor Unit

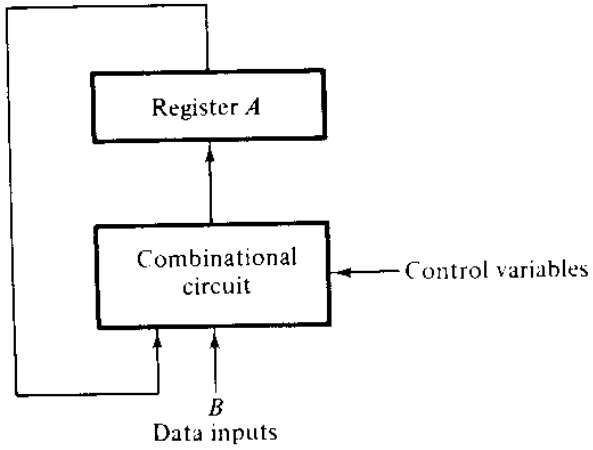
- The 3-bit binary code listed in the table specifies the code for each of the five fields A, B, D, input data, F, and H.
- The register selected by A, B, and D is the one whose decimal number is equivalent to the binary number in the code.
- When the A or B field is 000, the corresponding multiplexer selects the input data.
- When D = 000, no destination register is selected.
- The three bits in the F field, together with the input carry Cin, provide the 12 operations of the ALU
- There are two possibilities for F = A.
    - Carry bit C is cleared,
    - Carry bit C is set to 1.



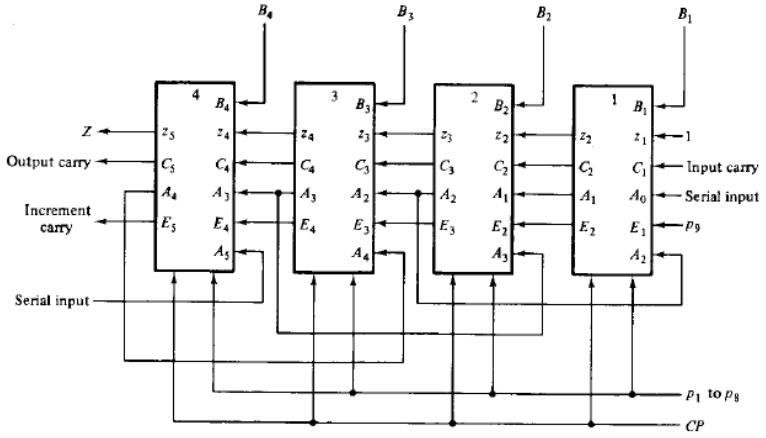
Binary code	Function of selection variables					
	A	B	D	F with $C_{in} = 0$	F with $C_{in} = 1$	H
0 0 0	Input data	Input data	None	$A, C \leftarrow 0$	$A + 1$	No shift
0 0 1	$R1$	$R1$	$R1$	$A + B$	$A + B + 1$	Shift-right, $I_R = 0$
0 1 0	$R2$	$R2$	$R2$	$A - B - 1$	$A - B$	Shift-left, $I_L = 0$
0 1 1	$R3$	$R3$	$R3$	$A - 1$	$A, C \leftarrow 1$	0's to output bus
1 0 0	$R4$	$R4$	$R4$	$A \vee B$	—	—
1 0 1	$R5$	$R5$	$R5$	$A \oplus B$	—	Circulate-right with C
1 1 0	$R6$	$R6$	$R6$	$A \wedge B$	—	Circulate-left with C
1 1 1	$R7$	$R7$	$R7$	$\overline{A}$	—	—

36. Explain the Design of accumulator

- Accumulator refers to both the A register and its associated combinational circuit.
- The external inputs to the accumulator are the data inputs from B and the control variables that determine the micro operations for the register.
- The next state of register A is a function of its present state and of the external inputs.
- Accumulator can also perform data processing operations.



Control variable	Microoperation	Name
p_1	$A \leftarrow A + B$	Add
p_2	$A \leftarrow 0$	Clear
p_3	$A \leftarrow \overline{A}$	Complement
p_4	$A \leftarrow A \wedge B$	AND
p_5	$A \leftarrow A \vee B$	OR
p_6	$A \leftarrow A \oplus B$	Exclusive-OR
p_7	$A \leftarrow$ shr A	Shift-right
p_8	$A \leftarrow$ shl A	Shift-left
p_9	$A \leftarrow A + 1$	Increment
	If $(A = 0)$ then $(Z = 1)$	Check for zero



- The inputs and outputs of each stage can be connected in cascade to form a complete accumulator.
- The number on top of each block represents the bit position.
- All blocks receive 8 control variables P1 to P8 and the clock pulses from CP.
- The other six inputs and four outputs are same as with the typical stage.
- The zero detect chain is obtained by connecting the z variables in cascade, with the first block receiving a binary constant I .
- The last stage produces the zero detect variable Z.
- Total number of terminals in the 4 bit accumulator is 25, including terminals for the A outputs.
- Incorporating two more terminals for power supply, the circuit can be enclosed within one IC package having 27 or 28 pins.
- The number of terminals for the control variable can be reduced from 9 to 4 if a decoder is inserted in the IC.
- In such cases, IC pin count is also reduced to 22 and the accumulator can be extended to 16 microoperations without adding external pins (That is, with 4 bits we can identify 16 operations).

MODULE 3

37. List the limitations of repeated addition method for multiplication.

a. Least sophisticated method

b. Just use adder over and over again

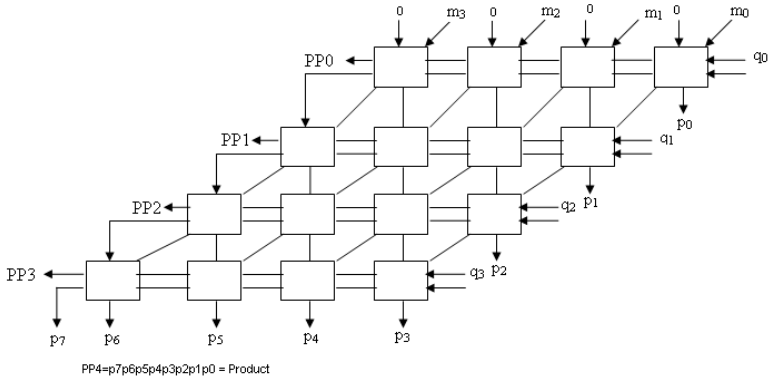
c. If the multiplier is n bits, can have as many as

d. 2n iterations of addition -- O(2n)

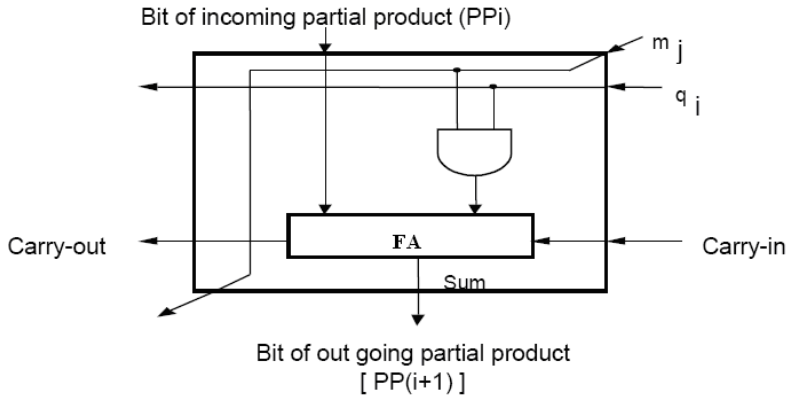
e. Not used in an ALU

38. Describe array processor with a neat diagram.

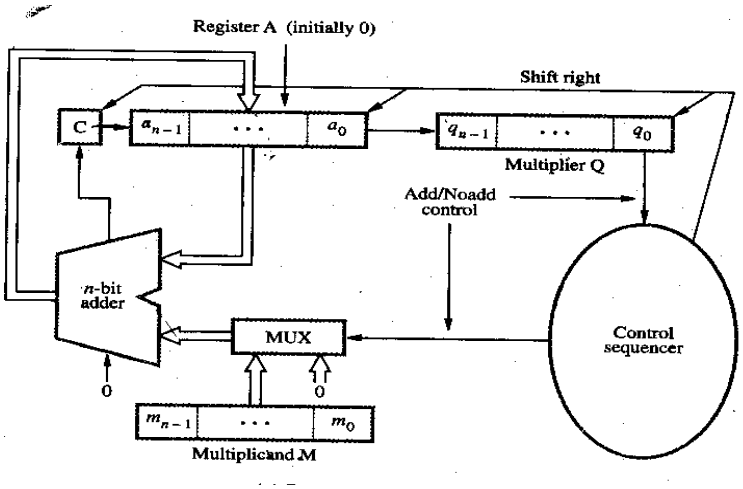
Binary multiplication of positive operands can be implemented in a combinational two dimensional array.

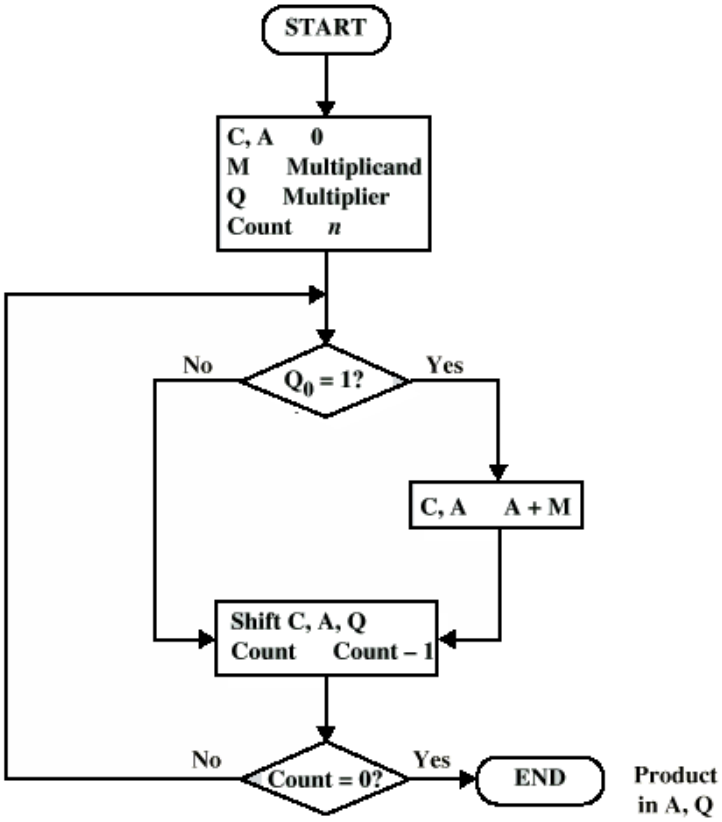


The AND gate in each cell determines whether a multiplicand bit **mj i**s added to the incoming partial product bit, based on the value of the multiplier bit **qj**. Each raw i adds the multiplicand to the incoming partial product **ppi** to generate pp(i+1) if qi=1. If qi=0 ppi is passed downward unchanged.



39. Expalin sequential multiplier with an example.

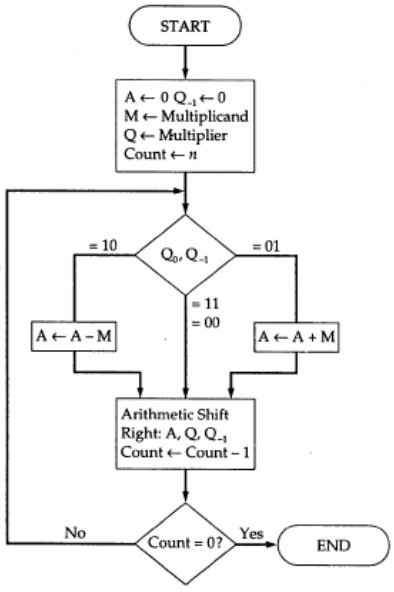




C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

40. Illustrate Booth algorithm with a suitable example.

When moving from 0 to 1 then -1 is selected and moving from 1 to 0 then +1 is selected as the multiplier scanning from left to right. This algorithm clearly extends to any number of blocks of 1s in a multiplier, including the situation in which a single 1 is considered as a block. If the first bit is 1 then consider the previous bit is 0.



The Multiplicand is placed in M register and multiplier is loaded into register Q. Registers A and Q_{-1} is cleared initially. A bit of multiplier is examined together with the bit in Q_{-1}. If

these bits are same (0-0,1-1) then all bits of A,Q,Q_{-1} are shifted right 1 bit. If two bits are differ then the multiplicand is added to or subtracted from A depending on 0-1 or 1-0 then right shift occurs. In either case the right shift A_{n-1} to A_{n-2} occurs and A_{n-1} maintains the same bit for maintaining the sign. This is called Arithmetic shift.

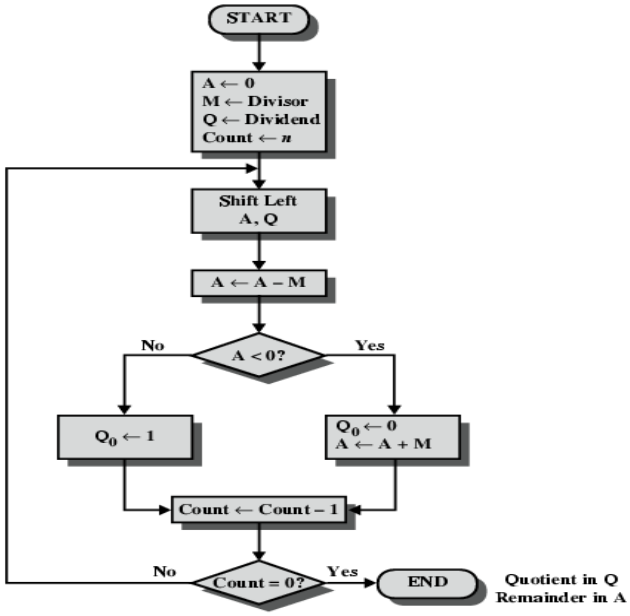
A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

41. What are the advantages of Booth Algorithm

- It handles both positive and negative multiplier uniformly.
- it reduce the number of partial product,
- The speed gained by skipping 1's depends on the data.

42. Illustrate restoring division algorithm with an example.

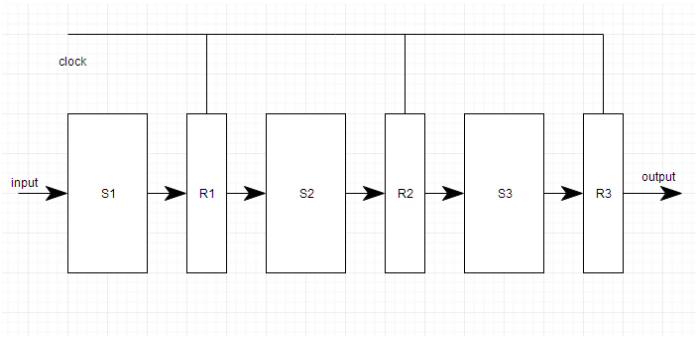
Restoring Division: Register A is initially loaded with 0 and it consists of n+1 bits, where n is the number of bits in the dividend. Dividend is loaded in register Q and Register M is loaded with the divisor. After division is complete n bit quotient is in register Q and remainder is in register A. extra bit on A and M accommodates the sign bit during subtractions.



A	Q	M=00011
00000	1000	Initial value
00001	000-	Shift
11110	0000	Subtract/set q_0
00001		Restore
00010	000-	Shift
11111	0000	Subtract/set q_0
00010		Restore
00100	000-	Shift
00001	0001	Subtract/set q_0
00010	001-	Shift
11111	0010	Subtract/set q_0
00010		**Restore**

43. What is Pipelining?

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end. In pipeline system, each segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit performs operations on it. The output of combinational circuit is applied to the input register of the next segment.



44. List the advantages of pipeline.

- Multiple tasks operating simultaneously using different resources
- Pipelining increases throughput

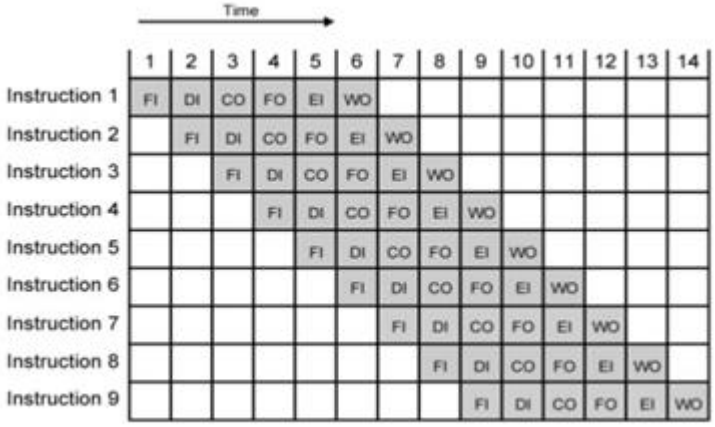
45. Different Types of Pipelines.

It is divided into 2 categories:

- Arithmetic Pipeline
- Instruction Pipeline

46. Explain instruction pipelining.

Instruction pipelining is a technique used in the design of modern microprocessors, microcontrollers and CPUs to increase their **instruction** throughput (the number of **instructions** that can be executed in a unit of time).



47. What are the various stages in instruction pipeline?

Stage 1 :**Instruction Fetch (FI)-**In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

Stage 2 **:Instruction Decode (DI)-**In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

Stage 3: **Calculate Operands(CO**)-In this stage, effective address of the operands are calculated.

Stage 4: **Fetch Operands (FO)-**memory operands are read from memory

Stage 5 :**Instruction Execute (EI)-**In this stage, ALU operations are performed.

Stage 6: **Write Back (WO)-**In this stage, computed/fetched value is written back to the register present in the instructions.

48. How pipelining speeding up the performance of a computer?

Speedup (S) of the pipelined processor over non-pipelined processor, when 'n' tasks are executed on the same processor is:

S= Performance of pipelined processor/ Performance of non pipelined processor

Consider a 'k' segment pipeline with clock cycle time as 'Tp'. Let there be 'n' tasks to be completed in the pipelined processor. Now, the first instruction is going to take 'k' cycles to come out of the pipeline but the other 'n – 1' instructions will take only '1' cycle each, i.e, a total of 'n – 1' cycles. So, time taken to execute 'n' instructions in a pipelined processor.

$ET_{pipeline}$=k+n-1 cycles= (k+n-1)Tp

In the same case, for a non-pipelined processor, execution time of 'n' instructions will be:

$ET_{non\ pipeline}$= n*k*Tp

As the performance of a processor is inversely proportional to the execution time, we have,

$S= ET_{nonipeline}/ET_{pipeline}$= n*k*Tp/ (k+n-1)Tp

= (n*k)/(k+n-1)

49. Explain Arithmetic Pipeline with an example.

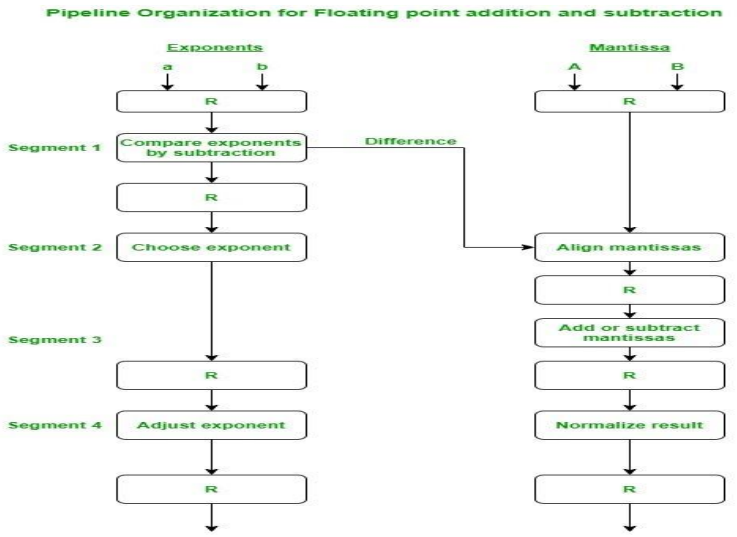
An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations.

Example: Floating point addition using arithmetic pipeline

The following sub operations are performed:

- Compare the exponents.
- Align the mantissas.
- Add or subtract the mantissas.

- Normalize the result



50. What is a pipeline hazards?

Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycles. Any condition that causes a stall in the pipeline operations can be called a hazard.

51. Define structural, data, and control hazard.

Structural hazard

a. two different instructions use same h/w in same cycle – They arise from the resource conflict
b. HW cannot support all combination of instructions in overlapped fashion (single person to fold and put clothes away)

Data hazard

c. They arise when the instruction depends on the result of previous instruction.
d. must appear as if the instructions execute in correct order

Control hazard

e. one instruction affects which instruction is next
f. They arise when the pipelining of branches & other instructions that change the PC

52. List the techniques used for overcoming hazard.

Structural hazard: Stall, Use of additional hardware

Data Hazard: Stall, Data Forwarding, Code reordering

Control Hazard: Stall, branch predict, delayed branch

53. Give examples of data hazard.

Read after Write (RAW): $Instr_J$ tries to read operand before $Instr_I$ writes it

Write after Read (WAR):$Instr_J$ tries to write operand *before* $Instr_I$ reads -Gets wrong operand

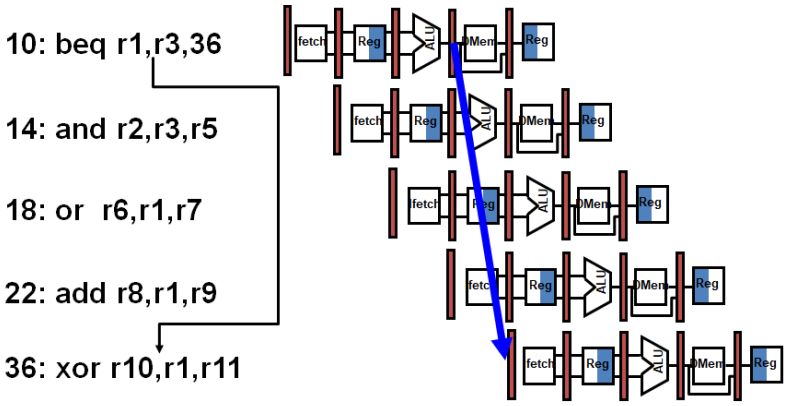
Write after Write (WAW): $Instr_J$ tries to write operand *before* $Instr_I$ writes it

- Leaves wrong result ($Instr_I$ not $Instr_J$)

- Read after Read (RAR).: It occurs when the instruction both read from the same register. No hazard occurs

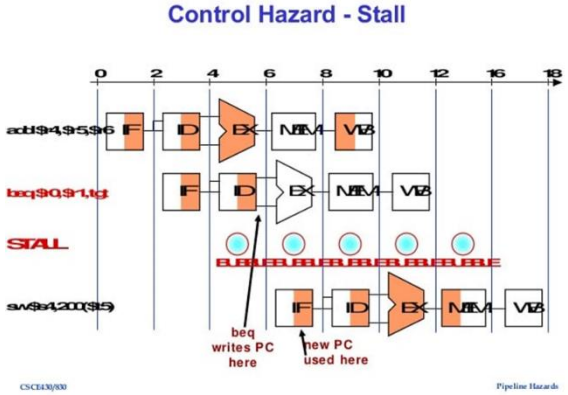
54. What is branch hazard? Describe the method for dealing with the branch hazard?

A control/branch hazard is when we need to find the destination of a branch, and can't fetch any new instructions until we know that destination.

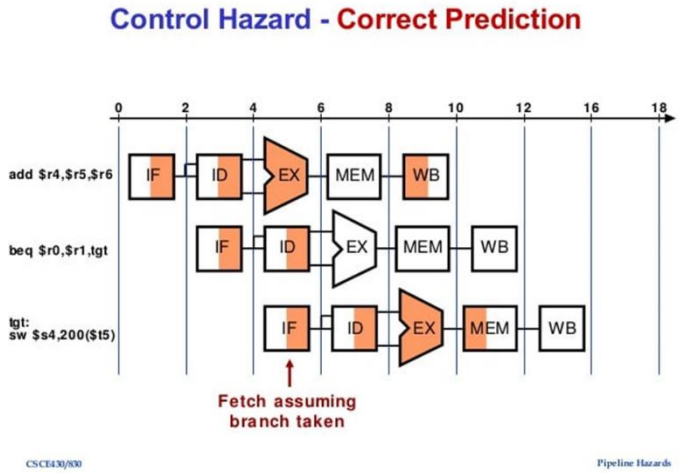


Solutions

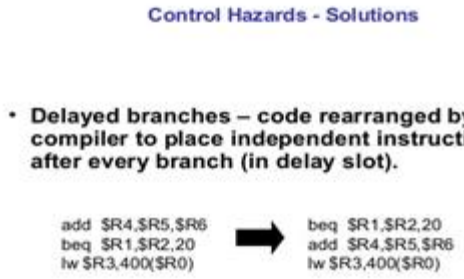
- **Stall:** Stop loading instruction until result is available. Stalling for each branch is not practical



- **Predict:** Assume an outcome and continue fetching (undo if prediction is wrong)
  - Loses cycles only on mis prediction

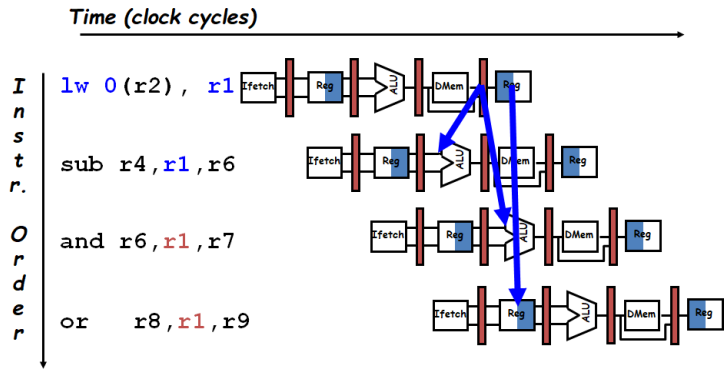


- **Delayed Branch**: Specify in architecture that the instruction immediately following branch is always executed.



55. Explain the methods for dealing with data hazard.

Forwarding: It adds special circuitry to the pipeline. This method works because it takes less time for the required values to travel through a wire than it does for a pipeline segment to compute its result.



Code reordeing: We need a special type of software to reorder code. We call this type of software a hardware-dependent compiler.

Stall insertion: it inserts one or more installs (no-op instructions) into the pipeline, which delays the execution of the current instruction until the required operand is written to the register file, but this method decreases pipeline efficiency and throughput.

56. Write the example of WAW hazard.

If instruction X tries to modify some data before it is written by instruction (X-1).

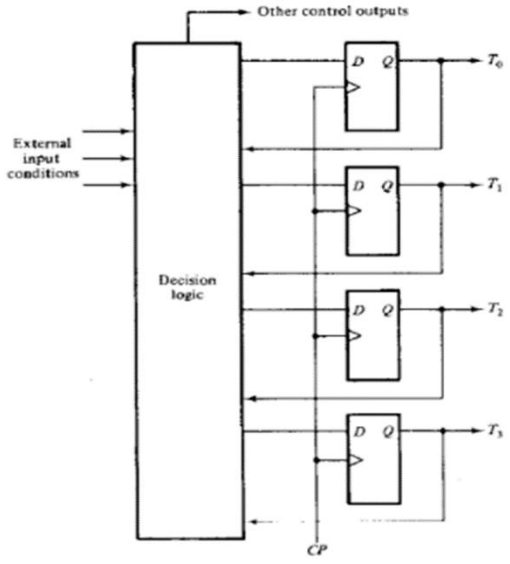
MODULE 4

57. What are the disadvantages of sequential control logic?

- Large number of states
- Excessive number of flip-flops and gates
- Design methods uses state and excitation tables but in practice they are cumbersome

58. **Explain the control organization in de**tail.

- **One flip-flop per state methods**

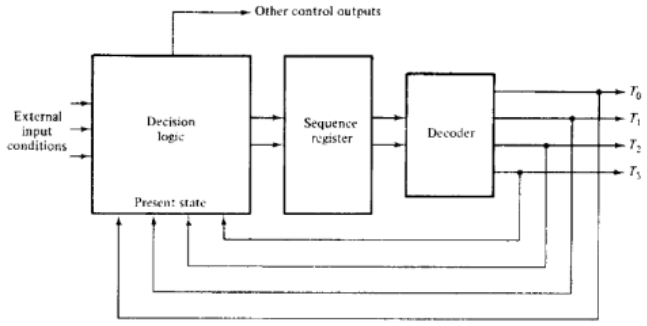


This method uses one flip-flop per state in the control sequential circuit. Only one flip-flop is set at any particular time: all others are cleared. A single bit is made to propagate from

one flipflop to the other under the control of decision logic. In such an array, each flip-flop represents a state and is activated only when the control bit is transferred to it.

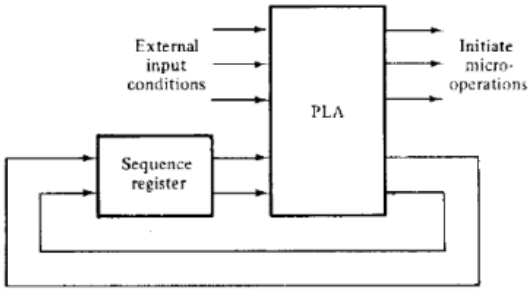
- **The advantage** of this method is the simplicity with which it can be designed.
- **The disadvantage** is that this method would increase system cost since more flip-flops are used.

- **Sequence register and decoder method**



This method uses a register to sequence the control states. The register is decoded to provide one output for each state. For n flip-flops in the sequence register, the circuit will have 2" states and the decoder will have 2" outputs.

- **PLA control**



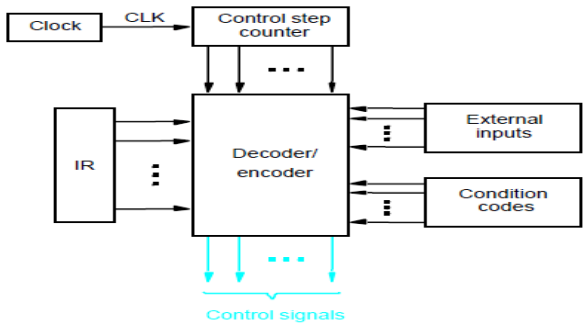
The external sequence register establishes the present state of the control circuit. The PLA outputs determine which micro-operations should be initiated depending on the external input conditions and the present state of the sequence register. At the same time other PLA outputs determine the next state of the sequence register. The sequence register is external to the PLA if the unit implements only combinational circuits

- **Micro-program control**

Control signals are generated by a program are similar to machine language programs. The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words (CW). A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the micro routine for that instruction, and the individual control words in this micro routine are referred to as microinstructions.

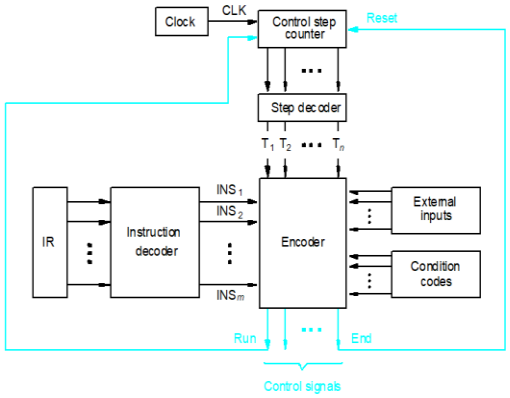
59. Explain hardwired control.

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.

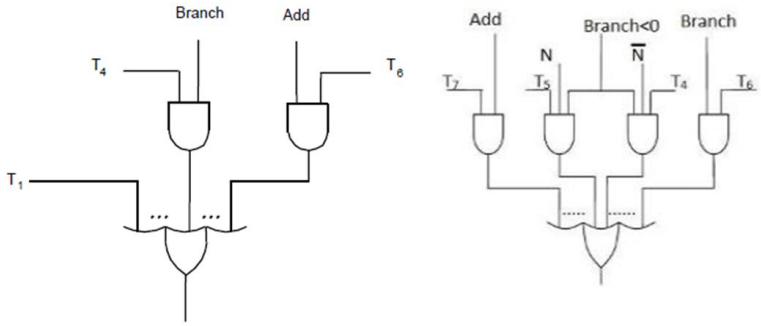
60. Explain the design a hardwired control for a simple control signal with an example.



Example

Zin=T1+T6.Add+T4.BR+…..

END= T7.Add+T5.BR+(T5.N+T4.N')BRN



61. Explain the advantages and disadvantages of hardwired control.

Advantages: High Speed

Disadvantages: requires changes in the wiring if the design has to be modified or changed-less flexible, Costly, and difficult to implement complex instructions

62. Define the terms Control word, Micro routine, Micro instructions, Micro program, Control store.

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

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

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

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).

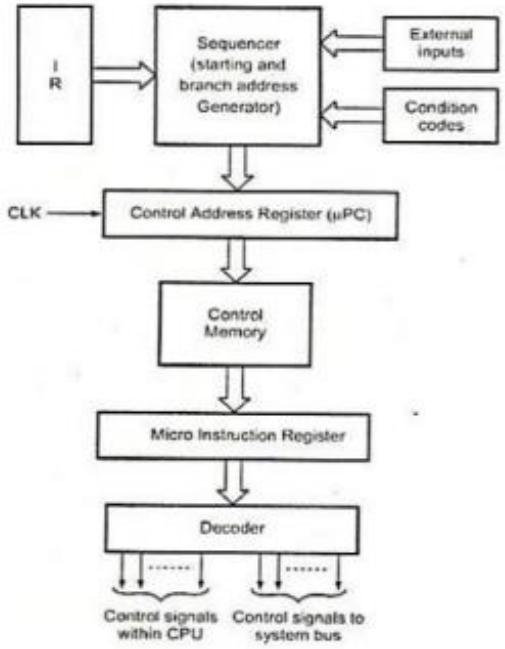
Control Store : the micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the Control Store.

63. Discuss the principle operation of micro programmed control unit.

Control variables that initiate micro-operations are stored in memory. The control memory is usually a ROM, since the control sequence is permanent and needs no alteration. The control variables stored in memory are read one at a time to initiate the sequence of micro-

operations for the system. Each micro operation is associated with a specific set of control lines which, when activated, causes that micro operation to take place.

64. Explain the components of microprogrammed control unit.



Control memory: In micro programmed control, the micro programs for all instructions are stored in the control memory (CM). The control signals to be activated at any time are specified by a microinstruction, which is fetched from Control memory (CM).

Control address register : The control address register holds the address of the next microinstruction to be read. When address is available in control address register, the sequencer issues READ command to the control memory.

Microinstruction register : After issue of READ command, the word from the addressed location is read into the microinstruction register. Now the content of the micro instruction register generates control signals and next address information for the sequencer.

Micro-program sequencer : A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program. The sequencer loads a new address into the control address register based on the next address information

65. List the advantages and disadvantages of microprogrammed control unit

Advantages

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A micro programmed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

66. Write the control sequence and micro instructions for the instruction Add (R3), R1

1. PCout, MARin, Read, Select4, Add, Zin
2. Zout, PCin, Yin, WMFC
3. MDRout, IRin
4. R3out, MARin, Read
5. R1out, Yin, WMFC
6. MDRout, Select Y, Add, Zin
7. Zout, R1in, End

Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

67. Explain Horizontal and vertical micro instructions.

A **horizontal microinstruction** is a minimally encoded scheme. The scheme of micro-instruction by assigning one bit position to each control signal is called horizontal micro-instructions.

Example: 011101001101001110

In a horizontal microinstruction every bit in the control field attaches to a controller. Horizontal microinstructions represent several micro-operations that are executed at the same time. A horizontal approach involves a wider control store, but is capable of greater speed.

Vertical microinstruction is a tightly encoded scheme. The vertical approach requires a narrow control store, but must be decoded in order to drive the actual control lines, thus introducing a delay in driving the control lines.

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	AS3 AS2 AS1	Read	Write	Carry-in	WMFC	End
0000: No action	000: No action	000: No action	000: ADD					
0001: PCout	001: PCin	001: MARin	001: INC					
0010: MDRout	010: IRin	010: MDRin	010: SUB					
0011: Zout	011: Zin	011: TEMPin	011: DEC					
0100: R0out	100: R0in	100: Yin	100: AND					
0101: R1out	101: R1in		101: OR					
0110: R2out	110: R2in		110: XOR					
0111: R3out	111: R3in		111: NOT					
1000: IRout								
1001: TEMPout								

68. Compare Horizontal and vertical micro instructions.

Horizontal	Vertical
Long formats.	Short formats
Ability to express a high degree of parallelism.	Limited ability to express parallel micro operations.
Little encoding of the control information.	Considerable encoding of the control information
Useful when higher operating speed is desired.	Slower operating speeds.

69. Explain the design of a micro program sequencer in detail.

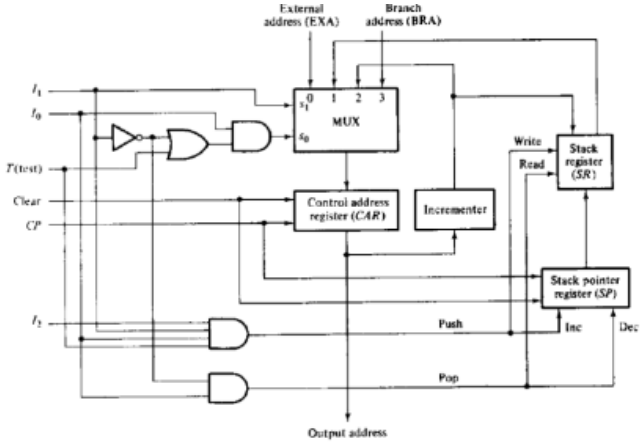
Micro program sequencer is a control unit which does the tasks of Micro-program sequencing.

Micro-program sequencer is attached to the control memory.

It inspects certain bits in the microinstruction to determine the next address for control memory.

A typical sequencer has the following address sequencing capabilities.

1. Increments the present address of control memory
2. Branches to an address which will be specified in the bits of microinstruction
3. Branches to a given address if a specified status bit is equal to 1.
4. Transfers control to a new address as specified by an external source
5. Has a facility for subroutines calls and returns.



It consists of a multiplexer that selects an address from four sources and routes it into a control address register (CAR). The output from CAR provides the address for control memory. The contents of CAR are incremented and applied to the multiplexer and to the stack register file. The register selected in the stack is determined by stack pointer. Inputs (I0-I2) specify the operation for the sequencer and input T is the test point for a status bit. Initially the address register is cleared to zero and clock pulse synchronizes the loading in to registers.

70. Compare and contrast hardwired and micro programmed control units.

Hardwired control	**Microprogrammed control**
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micrprogrammed control unit generates the control signals with the help of micro instructions stored in control memory
Faster	Slower

Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

MODULE 5

71. What are the different techniques for interfacing I/O device with CPU?

Memory mapped I/O and I/O mapped I/O

72. What is memory mapped I/O?

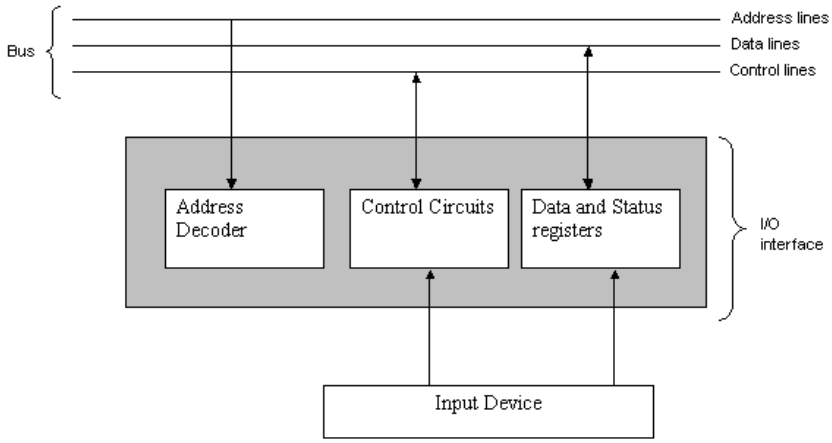
This is one of the techniques for interfacing I/O devices with CPU. In memory mapped I/O, the I/O devices assigned and identified by 16-bit addresses. To transfer the data between CPU and I/O devices memory related instructions (such as LDA, STA etc.) and memory control signals (MEMR, MEMW) are used. .

The CPU selects an I/O device by placing the address on the address lines; the corresponding I/O device recognizes its address and responds to the CPU's command. Any machine instruction used to access memory can be used to transfer data to and from I/O devices.

Example:

MOVE DATAIN, R_0

MOVE R_0, DATAOUT, Where DATAIN and DATAOUT are I/O buffer registers.



73. What is I/O mapped I/O?

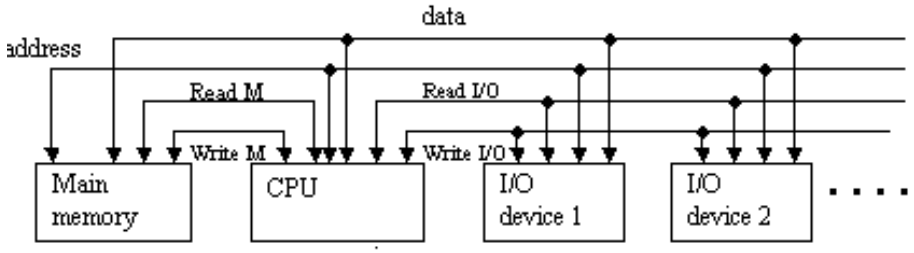
The memory and I/O address space are kept separate by Read/Write control lines.A memory reference instruction activates the Read M or Write M line and does not affect the I/O devices.

The I/O instruction such as for example IN or OUT activates the Read I/O and write I/O line and causes a word to be transferred between the addressed I/O and the CPU.

Therefore, the memory location and the I/O device may have the same address.

OUT data, device

IN data, device



74. What are the different data transfer modes between the CPU and the I/O devices.

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

75. Explain Programmed I/O.

It is a mode of data transfer between the CPU and the logic within the peripherals. The transfer is initiated using an I/O transfer (IOT) instruction and controlled entirely by the CPU.

Programmed I/O can be:

- Synchronous (Or Unconditional) transfer: Here no check is made by the CPU to determine the state of the peripherals like Ready or Busy. The transfer takes place when ever the CPU directs it i.e. the I/O device must be ready to receive the output data from the CPU or to input data into the CPU when ever CPU initiates such a transfer. Hence the transfer is Unconditional. The data transfers are controlled by a clock of fixed frequency, independent of the CPU
- Asynchronous (Or Conditional) transfer: The peripheral is queried about its status to see if it is ready to perform the transfer. The actual transfer takes place only when the peripheral shows Ready state. This method accomplishes synchronization between the CPU and slower peripherals whose timing may not be fixed.

76. List the advantages and disadvantages of Programmed I/O.

Advantages

- The programmed I/O mode is very low cost interface
- Can meet speeds of most of the peripherals.

Disadvantages: CPU speed is brought down to that of the slow peripheral device

77. Explain Interrupt driven I/O.

Here, the CPU does not wait for the peripheral device to become ready for the data transfer instead, it initiates the device and then goes away to handle other tasks. The peripheral sends the Ready signal to the CPU whenever it is ready. An interrupt request from an I/O device may cause the processor to suspend the execution of one program and start the execution of another program. Interrupts may sometimes alter the sequence of events. The programmer should be able to enable and disable interrupts. This is performed by machine instructions such as Interrupt Enable and Interrupt Disable.

78. **Explain the general sequence of events in interrupt I/O**

Interrupt request - The I/O requesting peripheral sends an interrupt request signal (INT REQ) to the CPU to indicate that it is ready for an I/O operation

Interrupt acknowledge - On receiving the interrupt request signal, the CPU completes the execution of the current instruction and sends an Interrupt acknowledgement signal (INT ACK) to the interrupt requesting peripheral

Status saving - Before servicing the interrupt request, the return address of the main program is stored in some pre-assigned memory locations, so that the CPU can resume its normal operation on completion of the subroutine.

Device identification - The interrupt request to the CPU is only through a single line bus, if the system has more than one device connected to it, the CPU must identify the interrupting peripheral, so that it can call the appropriate interrupt service routine (ISR) from the memory.

Peripheral service – having identified the peripheral which requested for the service, the program branches to the starting address of the specific subroutine in the memory and it services the peripheral.

Restore CPU status – after completion of the service subroutine, the data from the CPU registers that were held temporarily in memory are reloaded back to their respective locations and the CPU can take up its normal task

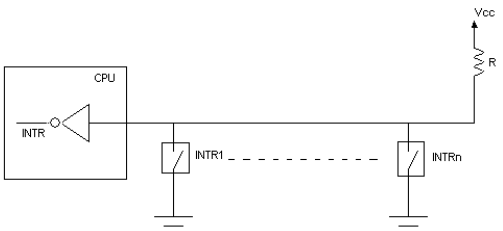
Resume main task – when all registers and program counter are restored with their original contents, the main program operation is resumed.

79. Explain the methods for identifying the interrupting device.

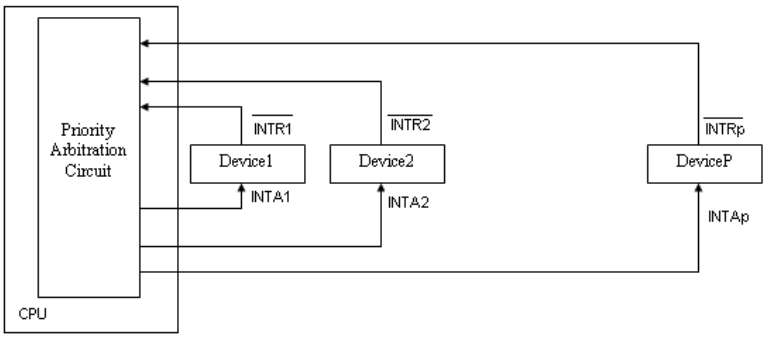
Polling and vectored interrupts are the methods to identifying the interrupting device.

Polling

All the devices are connected to a common INTR (Interrupt Request) line via switches to ground. When all the interrupt requests signals (INTR1 to INTRn) are inactive, the INTR line is maintained in the high voltage state. When any device requests an interrupt it closes its switch and the voltage on the line drops to zero, causing INTR line to go to 1.



A device that requests an interrupt identifies itself to the processor by sending a special code called vector to the processor. This code supplies the starting address of the ISR for that device. The processor reads this address, called the interrupt vector & loads into PC. This simplifies the circuit but it limits the number of devices that can be identified. One solution is to assign a code to a group of devices and the ISR identify the device by polling the devices in that group.



80. Explain interrupt nesting.

An interrupt request from a high priority device should be accepted while the processor is servicing another request from a low priority device. For this the priority level of the processor can be set to that of the device that is currently being serviced. Interrupt requests from the devices with priority greater than that of the processor will only be accepted.

Processor's priority is encoded in a few bits of the Program Status Word. Hardware is implemented by using a separate INTR and INTA line for each device. An interrupt request is sent to the priority arbitration logic and is accepted only if it is greater than the one that is currently assigned to the processor.

81. What is DMA?

Direct memory access refers to that data transfer, when large blocks of data are to be transferred at high speeds between I/O devices and main memory, without involving the

CPU. In order to perform DMA transfer, an additional interface circuitry is required known as **DMA controller.**

82. Describe DMA Controller registers.

The DMA controller has the following four registers to perform transfer:

a. **Address register** – used to hold the address of the next word to be transferred
b. **Data buffer register** – used to store the data to be transferred
c. **Data count register or word counter** – used to store the number of words that are remaining to be transferred. It is automatically decremented after the transfer of each word and tested for zero. When the data count reaches zero, the DMA transfer halts.
d. **Control register and circuits** – used to control the DMA transfer

83. What is the need for DMA transfer?

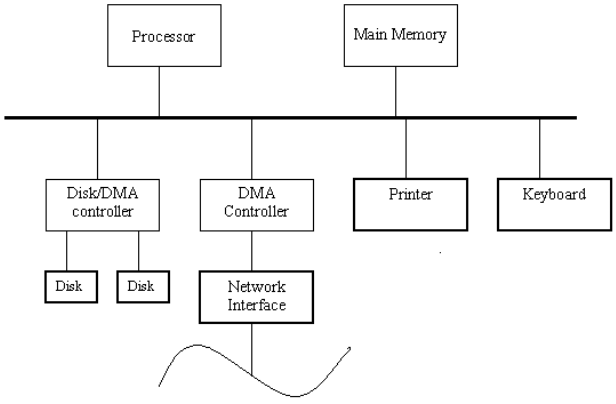
During the actual transfer between peripheral and memory, CPU is free from this activity and can do some other useful work.

84. Explain the action carried out by the processor after occurrence of an interrupt.

An interrupt is a signal that refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. The control returns to the original program after the service program is executed. Interrupts are used to coordinate slow external devices with fast CPU. One of the bus control lines called the interrupt request line is dedicated for this purpose. Here, the CPU does not wait for the peripheral device to become ready for the data transfer instead, it initiates the device and then goes away to handle other tasks. The peripheral sends the Ready signal to the CPU whenever it is ready.

85. Explain with the block diagram the DMA transfer in a computer system

The peripheral devices initiate the DMA transfer by sending a DMA request signal to the CPU when it is ready to perform a transfer.



There are two methods by which DMA transfer is initiated and operated:

Burst Mode

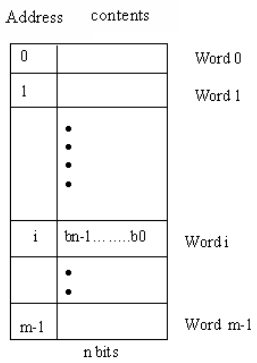
When the data is being transferred to the memory or from the memory, the CPU remains idle for relatively long periods of time or the CPU is in a kind of HOLD state. The CPU exits form this state only after the DMA request signal is withdrawn from the I/O device.The hold duration depends on the speed of the memory, I/O device, and number of bytes to be transferred.

Cycle-Stealing Mode

The CPU after receiving the DMA request signal from the peripheral, will attend to the current instruction, it is doing and suspends its operation for the next time slice (machine cycle). During the next time slice, the DMA transfer takes place and one word is transferred between the peripheral device and the main memory. On completion of the transfer, the CPU can resume to its normal operation. DMA transfers are essentially time-sliced with the operations of the main memory. Here, the DMA controller can be regarded as "stealing" memory cycles from the CPU. Hence, this technique is known as cycle stealing.

86. Explain Memory system

The main memory consists of a large number of storage cells, each of which can store a binary digit 0 or 1. This 1-bit representation of information is too small to be handled by a computer. So, a group of n bits are used while storing or retrieving. Each group of n bits is referred to as its word length. The word length can be 8 bits, 16 bits, 32, 64 or 128 bits depending on the size of the computer.



Each location in the main memory is given a distinct name or address. With this address it is possible to store or retrieve information from the memory location. The contents of memory locations can represent either instructions or operands.

87. Explain Big Endian and Little Endian Assignments.

Two ways that byte address can be assigned across words. Big Endian and Little Endian.

Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). In big endian, store the most significant byte in the smallest address.

Word address	Byte address			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11

Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. In little endian, you store the least significant byte in the smallest address.

Word address	Byte address			
0	3	2	1	0
4	7	6	5	4
8				

88. Define Memory access time, memory cycle time and transfer rate.

- **Memory Access time:** It is the time that elapses between the initiation of an operation and the completion of that operation. E.g.: time between Read and MFC
- **Memory Cycle time :** It is the minimum time delay required between the initiations of two independent memory operations. E.g.: two successive Read operations.

- **Transfer Rate:** It is the rate at which the data can be transferred to or out of memory unit.

TN=TA+N/R

TN= Average time to read / write n bits

TA=- Average access time

N= Number of bits

R= Transfer rate in bits/ sec.

89. Discuss the Classification of memory

Memory can be either primary memory or Secondary memory.

Two classifications of primary storage are read-only memory (ROM) and random-access memory (RAM).

a) READ-ONLY MEMORY (ROM)

In computers, it is useful to have instructions that are used often, permanently stored inside the computer. ROM enables us to do this without loosing the programs and data when the computer is powered down. Only the computer manufacturer can provide these programs in ROM; once done, we cannot change it. Consequently, we cannot put any of our own data or programs in ROM. Many complex functions, such as translators for high-level languages, and operating systems are placed in ROM memory. Since these instructions are hardwired, they can be performed quickly and accurately.

- **Programmable ROM (PROM):** This is a type of ROM that can be programmed using special equipment; it can be written to, but only once. This is useful for companies that make their own ROMs from software they write, because when they change their code they can create new PROMs without requiring expensive equipment. This is similar to the way a CD-ROM recorder works by letting you "burn" programs onto blanks once and then letting you read from them many times. In fact, programming a PROM is also called *burning*, just like burning a CD-R, and it is comparable in terms of its flexibility.
- **Erasable Programmable ROM (EPROM):** An *EPROM* is a ROM that can be erased and reprogrammed. A little glass window is installed in the top of the ROM package, through

which you can actually see the chip that holds the memory. Ultraviolet light of a specific frequency can be shined through this window for a specified period of time, which will erase the EPROM and allow it to be reprogrammed again.

- **Electrically Erasable Programmable ROM (EEPROM):** The next level of erasability is the *EEPROM*, which can be erased under software control. This is the most flexible type of ROM, and is now commonly used for holding BIOS programs.

b) RANDOM-ACCESS MEMORY (RAM)

RAM is another type of memory found inside computers. It may be compared to a chalkboard on which we can scribble down notes, read them, and erase them when finished. In the computer, RAM is the working memory. Data can be read (retrieved) or written (stored) in RAM by providing the computer with an address location where the data is stored or where you want it to be stored. When the data is no longer required, we may simply write over it. Thus you can use the storage location again for something else.

The RAM family includes two important memory devices **static RAM** (SRAM) and **dynamic RAM** (DRAM). The primary difference between them is the lifetime of the data they store. *SRAM* retains its contents as long as electrical power is applied to the chip. If the power is turned off or lost temporarily, its contents will be lost forever. *DRAM*, on the other hand, has an extremely short data lifetime-typically about four milliseconds. This is true even when power is applied constantly.

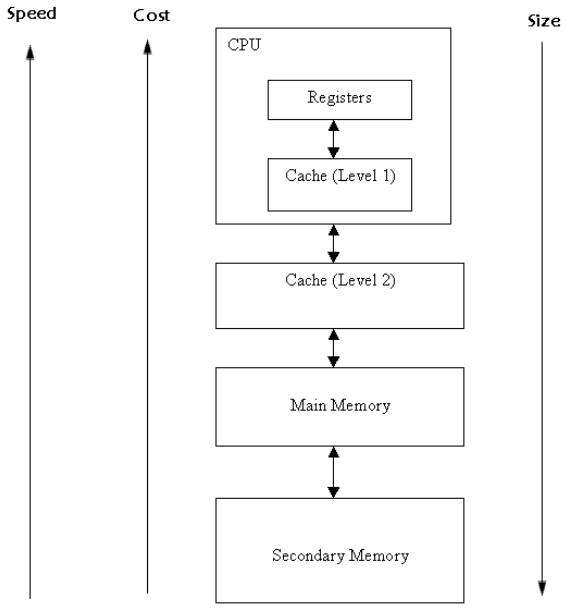
A simple piece of hardware called a *DRAM controller* can be used to make DRAM behave more like SRAM. The job of the DRAM controller is to periodically refresh the data stored in the DRAM. By refreshing the data before it expires, the contents of memory can be kept alive for as long as they are needed. So DRAM is as useful as SRAM after all.

SRAM devices offer extremely fast access times (approximately four times faster than DRAM) but are much more expensive to produce. Generally, SRAM is used only where access speed is extremely important. A lower cost-per-byte makes DRAM attractive whenever large amounts of RAM are required.

Secondary storage, or auxiliary storage, is memory external to the main body of the computer (CPU) where programs and data can be stored for future use. When the computer is ready to

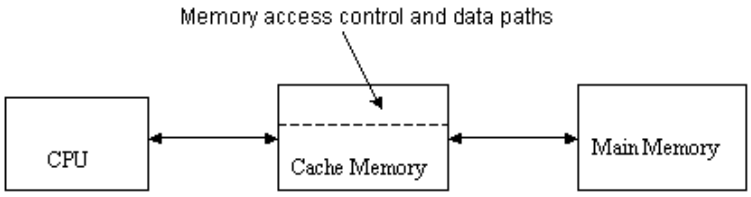
use these programs, the data is read into primary storage. Secondary storage media extends the storage capabilities of the computer system. Secondary storage is required for two reasons. First, the working memory of the CPU is limited in size and cannot always hold the amount of data required. Second, data and programs in secondary programs do not disappear when the power is turned off. Secondary storage is nonvolatile memory. This information is lost only when we erase it. Magnetic disks are the most common type of secondary storage. They may be either floppy disks or hard disks (hard drives).

90. Draw the picture of memory hierarchy.



91. What is the need of a cache memory?

Speed of main memory is very low compared to processor speed. The speed gap between processor and memory is resolved by using cache memories. The cache memory is a small and fast memory that is placed between the main memory and the CPU. It holds the currently active and more frequently used segments of program and its data.



In this approach, the CPU need not always fetch the program and data from the main memory instead it can get it from the cache memory. Hence, the total time of execution is considerably reduced and it is more economical.

92. Explain the basic principle of cache.

The cache working is based on the phenomenon **Locality of Reference.**

Principle of Locality of reference: Program access a relatively small portion of the address space at any instant of time.

Example: 90% of time in 10% of the code

The locality of reference occur in two ways

- **Spatial Locality**

Instructions are closed to recently executed instruction (based on their address) are likely to be executed soon.

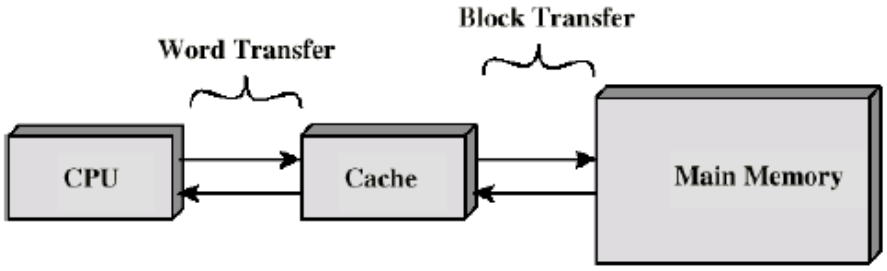
- **Temporal Locality**

Recently executed instructions are likely to be executed again very soon.

If the active segments are placed in the cache then execution time can be reduced then the total execution time is reduced. Due to temporal aspect, wherever information (data) are first needed this item should be brought into cache. Due to spatial aspect, instead of fetching an item fetch the assignment also.

93. Explain the basic operation of cache.

The cache memory is divided into number of blocks or cache lines.



When the CPU requests for an addressed word, it is transferred from the main memory to the cache. The memory-access control circuitry in the cache, checks whether the requested word currently exists in the cache. If the requested word exists in cache, then the **Read or Write hit** occurs and the required operation is performed on the appropriate location in the

cache. When the operation is Read, the main memory is not involved. However, if the operation is Write, then there are two techniques followed. In the first technique known as **write through**, the main memory and cache locations are updated simultaneously. In the second technique known as **write back**, only the cache location is updated and it is marked as updated using **dirty** or **modified bit.** The main memory will be updated during swapping. If the requested word does not exist in cache, then **a read or write miss** (cache miss) occurs. If the operation is Read**,** then there are two techniques followed. In the first technique the block of words that contain the requested word is loaded from the main memory into the cache and the requested word is forwarded to the CPU. In the second technique the requested word is directly sent to the CPU as soon as it is read from main memory. This technique is known as **load-through or early restart.** This reduces the processor waiting time. When the operation is Write, and the requested word is not in cache, then it is written directly into the main memory.

94. Define Hit Ratio of cache memory.

$$\text{Hit Ratio} = \frac{\text{Number of hits}}{\text{Total CPU references}}$$

$$\text{Hit Ratio} = \frac{\text{Number of hits}}{\text{Number of Hits} + \text{Number of Misses}}$$

95. What are the mapping functions commonly used in cache memory?

Direct mapping technique

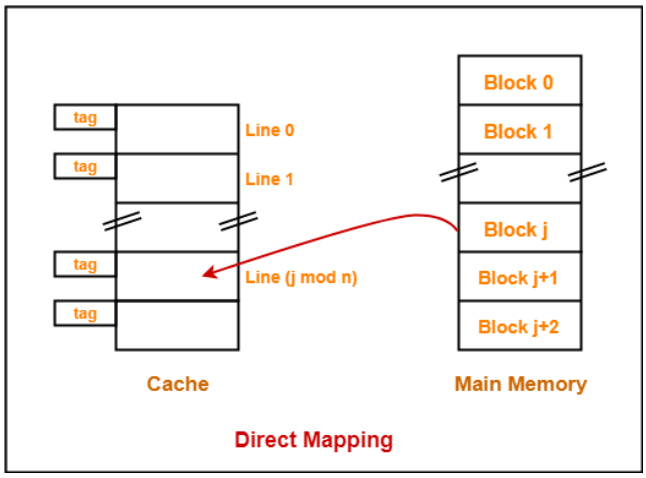
Associative mapping technique

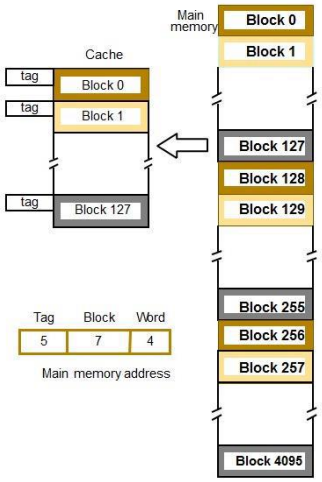
Set associative mapping technique

96. Explain direct mapping technique.

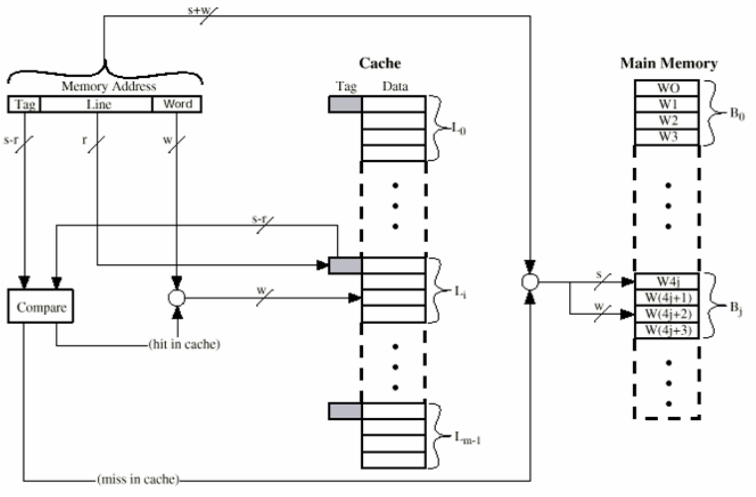
In direct mapped cache each main memory block is assigned to a specific line in the cache according to the formula i = j modulo N, where i is the cache line number assigned to main memory block j, and N is the number of lines in the cache. Direct mapping cache interprets a main memory address (comprising of s + w bits) as 3 distinct fields. Tag (the most significant s - r bits) is a unique identifier for each memory block. Line number (middle r bits) specifies which line will hold the referenced address. Line number identify one of the N = 2^r lines of cache.

Word (least significant w bits) specifies the offset of a byte within a line or block. Word specifies the specific byte in a cache line that is to be accessed.



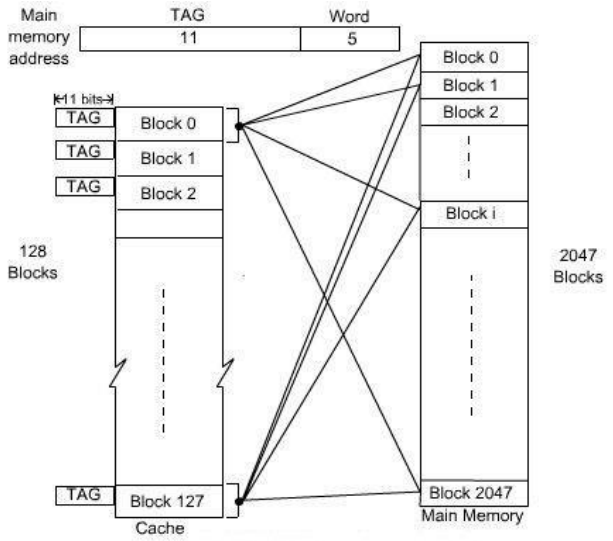


Block j of the main memory maps to j modulo 128 of the cache. (ie) Block 0, 128, 256 of main memory is maps to block 0 of cache memory. 1,129,257 maps to 1, & so on.



97. Explain Associative mapping.

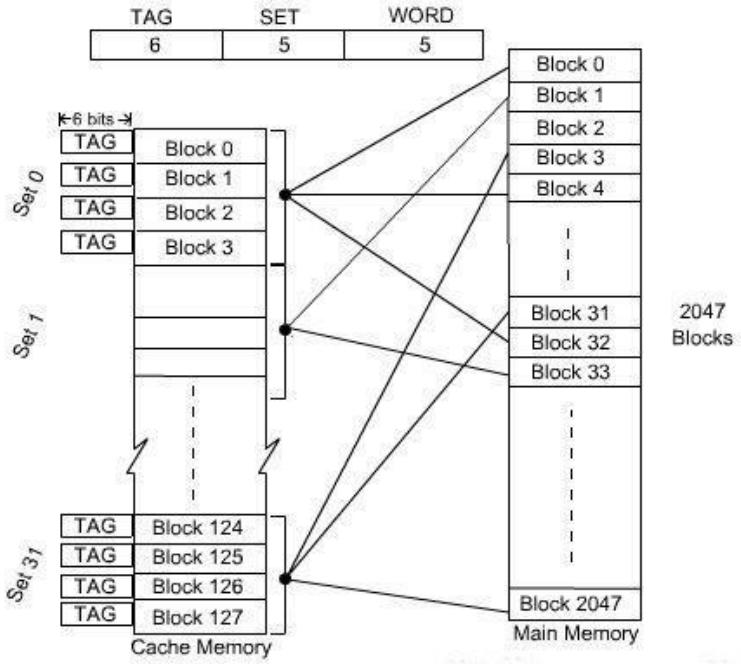
In associative mapping a block of the main memory can map to any available cache line. Associative mapping cache interprets a main memory address (comprising of s + w bits) as two distinct fields. Tag (the most significant s bits) is a unique identifier for each memory block. Word (least significant w bits) specifies the offset of a byte within a line or block. It might not be practical to use this complete flexibility of associative mapping technique due to searching overhead, because the TAG field of main memory address has to be compared with the TAG field of the entire cache block.



98. Explain Set associative mapping.

This mapping is compromise between direct mapping and fully associative mapping. It builds on the strengths of both: namely, the easy control of the direct mapped cache and

the more flexible mapping of the fully associative cache. In set associative mapping, cache is divided into a number of smaller direct-mapped areas called sets (*v*), with each set holding a number of lines (*k*).The cache is then described in the number of lines each set contains. If a set can hold *X* lines, the cache is referred to as an X-way set associative cache. A main memory block can be stored in any one of the *k* lines in a set such that *cache set number = j modulo v (*where *j* is the main memory block number).



The TAG field of associative mapping technique is divided into two groups, one is termed as SET bit and the second one is termed as TAG bit. Each set contains 4 blocks, total number of set is 32. The main memory address is grouped into three parts: low-order 5 bits are used to identifies a word within a block. Since there are total 32 sets present, next 5 bits are used to identify the set. High-order 6 bits are used as TAG bits.

99. What is meant by block replacement?

When the cache is full, there is a need for replacement algorithm for replacing the cache block with a new block.

100. Discuss various block replacement algorithms in cache.

Three types of replacement algorithms

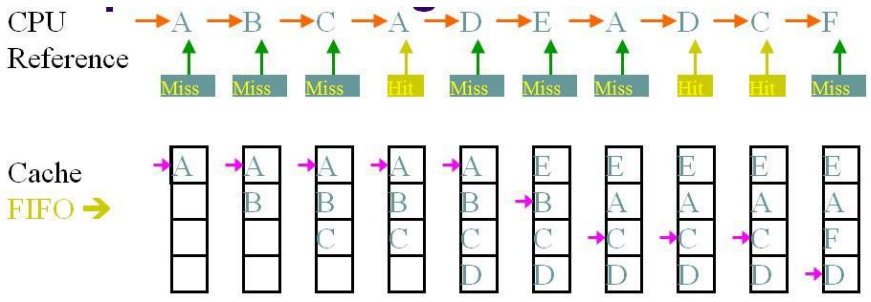
• **Random replacement policy.**

Choose the block to be overwritten at random. In this algorithm replace any cache line by using random selection. Simple and very effective in practice.

• **First in first Out (FIFO) replacement policy**

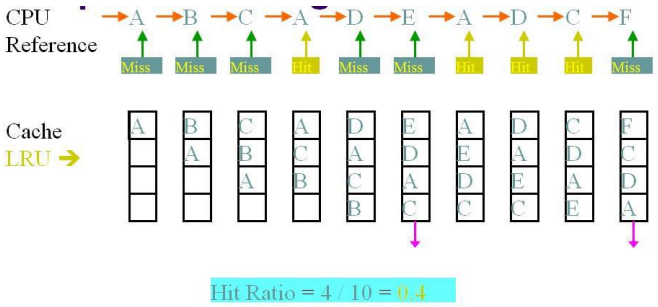
In this algorithm replace the cache block which is having the longest time stamp. While using this technique there is no need of updating when a hit occurs but when there is a miss occur then the block is put into an empty block and the counter values are incremented by one.

Example:



• **Least recently used (LRU) replacement policy.**

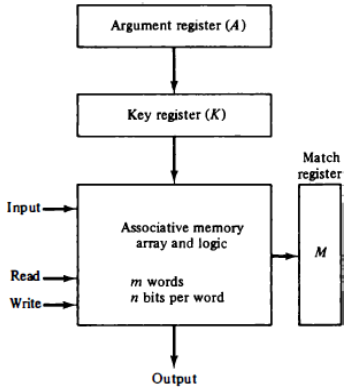
In the LRU, replace the cache block which is having the less reference with the longest time stamp. In this case also when a hit occurs when the counter value will be set to 0 but when the miss occurs there will be arising of two possibilities in which one possibility is that counter value is set as 0 and in another possibility, the counter value can be incremented as 1.



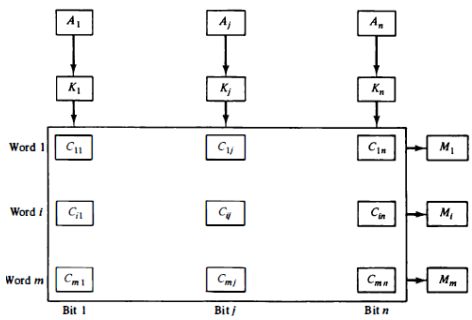
101. Discuss the concept of associative memory.

A memory unit accessed by content is called **an associative memory or Content Addressable Memory (CAM).**

This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word, or part of the word, is specified. The memory locates all words, which match the specified content and marks them for reading. The organization of associative memory is as shown in figure 8. The comperand register is also known as argument register. The masking register is known as key register.



The match register M has m bits, one for each memory word. Each word in memory is compared in parallel with the content of the argument register. The words that match the bits of the argument register set a corresponding bit in the match register. After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.



Example

A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match