Model Question Paper

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRD SEMESTER B.TECH DEGREE EXAMINATION, MONTH & YEAR
Course Code: CST 202
Course Name: Computer organisation and architecture
Max.Marks:100 Duration: 3 Hours

1. Give the significance of instruction cycle.

Executing a given instruction is a two-phase procedure. In the **first phase**, called **instruction fetch**, the instruction is fetched from the memory location whose address is in the PC. This instruction is placed in the instruction register (IR) in the processor.

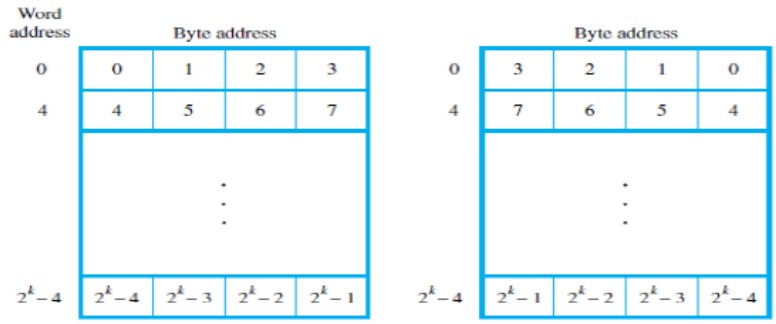
At the start of the **second phase**, called **instruction execute**, the instruction in IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor. This involves a small number of steps such as fetching operands from the memory or from processor registers, performing an arithmetic or logic operation, and storing the result in the destination location. At some point during this two-phase procedure, the contents of the PC are advanced to point to the next instruction. When the execute phase of an instruction is completed, the PC contains the address of the next instruction, and a new instruction fetch phase can begin.

Significance

- The instruction cycle of a computer system is necessary for understanding the flow of instructions and the execution of an instruction in a computer processor.
- It is responsible for the complete flow of instructions from the start of the computer system through its shutdown. The instruction cycle helps to understand the internal flow of the central processing unit, allowing any faults to be immediately resolved.
- It deals with a computer processor's basic operations and demands a detailed understanding of the many steps involved.
- All instructions for the computer processor system follow the fetch-decode-execute cycle.

2. Distinguish between big endian and little endian notations. Also give the significance of these notations

There are two ways that byte addresses can be assigned across words. The name **big-endian** is used when lower byte addresses are used for the more significant bytes (the leftmost bytes) of the word. The name **little-endian** is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word. In both cases, byte addresses 0, 4, and 8… are taken as the addresses of successive words in the memory of a computer with a 32-bit word length. These are the addresses used when accessing the memory to store or retrieve a word.



3. Compare I/O mapped I/O and memory mapped I/O.

Memory-mapped I/O and I/O-mapped I/O are the two major classifications that are based on the way, I/O devices are interfaced in a microprocessor-based system. The significant difference between memory-mapped I/O and I/O-mapped I/O is that in the case of memory-mapped I/O, the I/O device is mapped into memory thus address space is used by both memory and I/O device. As against, in case of I/O mapped I/O, I/O device is mapped into I/O space thus two separate address spaces are used for memory and I/O device.
In memory mapped I/O the memory-specific instructions are used by the I/O device. While in case I/O mapped I/O only IN and OUT instructions are accessed.

Basis for Comparison	**Memory mapped I/O**	**I/O mapped I/O**
Basic	I/O devices are treated as memory.	I/O devices are treated as I/O devices.
Data transfer instructions	Same for memory and I/O devices.	Different for memory and I/O devices.
Cycles involved	Memory read and memory write	I/O read and I/O write
Interfacing of I/O ports	Large (around 64K)	Comparatively small (around 256)
Control signal	No separate control signal is needed for I/O devices.	Special control signals are used for I/O devices.
Efficiency	Less	Comparatively more
Decoder hardware	More decoder hardware required.	Less decoder hardware required.
Data movement	Between registers and ports.	Between accumulator and ports.
Logical approach	Simple	Complex
Usability	In small systems where memory requirement is less.	In systems that need large memory space.

Basis for Comparison	Memory mapped I/O	I/O mapped I/O
Speed of operation	Slow	Comparatively fast

4. Give the importance of interrupts in I/O interconnection.

When a program enters a wait loop, it will repeatedly check the device status. During this period, the processor will not perform any function. There are many situations where other tasks can be performed while waiting for an I/O device to become ready. To allow this to happen, we can arrange for the I/O device to alert the processor when it becomes ready.

It can do so by sending a hardware signal called an **interrupt** to the processor. At least one of the bus control lines called an **interrupt request line** is usually dedicated for this purpose. Since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions. Indeed by using interrupts such waiting periods can ideally be eliminated.

The routine executed in response to an interrupt request is called **Interrupt Service Routine.**

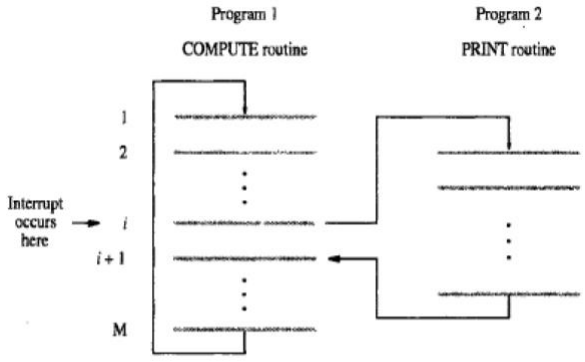


Figure 4.5 Transfer of control through the use of interrupts.

5. Justify the significance of status register.

The relative magnitude of two numbers may be determined by subtracting one number from the other and then checking certain bit conditions in the resultant difference. This status bit conditions are stored in a status register.

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. It is cleared if the output carry is 0.

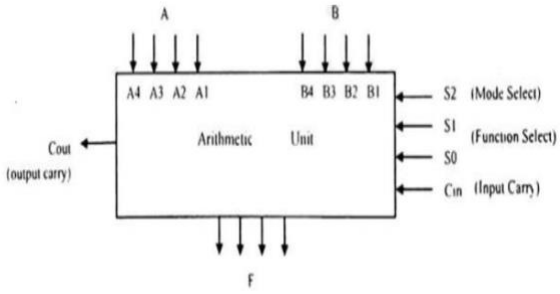
Bit S is set to 1 if the highest order bit of the result in the output of the ALU is 1. If the highest order bit is 0, this bit is cleared.

Bit Z is set to 1 if the output of the ALU contains all 0's. That is if the result is zero Z bit is 1, and if the result is nonzero Z bit is 0.

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.

6. How does the arithmetic circuitry perform logical operations in an ALU.

An ALU is a multioperation, combinational logic digital function. It can perform a set of basic arithmetic operations and a set of logic operations. The ALU has number of selection lines to select a particular operation in the unit. The selection lines are decoded within the ALU.



The four data inputs from A are combined with the four inputs B to generate an operation at the F outputs. The mode select input S2 distinguishes between arithmetic and logic operations. The two function select inputs S1 and S0 specify the particular arithmetic or logic operation to be generated. With three selection variables it is possible to specify four arithmetic operations and four logical operations. The input and output carries have meaning only during an arithmetic operation.
The design of ALU can be carried out in 3 stages:

1. Design of Arithmetic section

2. Design of Logic section

3. Modification of arithmetic section so that it can perform both arithmetic and logic operations.

7. Illustrate divide overflow with an example.

In a computer system, the division operation can lead to a quotient with an overflow because the registers cannot hold a number that exceeds the standard length. To understand this better, consider a system with a standard 5-bit register.

One register is used to hold the divisor and the other to hold the dividend. In case the quotient consists of 6 bits, 5 bits of the quotient will be stored in a 5-bit register. Therefore, the overflow bit needs a flip-flop to store the sixth bit.

The divide overflow condition occurs in case the high-order half bits of the dividend comprises a number that is greater than or equal to the divisor. One other point that needs to be considered in the division is that it is advisable to avoid division by zero. The overflow condition is generally detected when a flip-flop is set. This flip-flop is known as DVF.

8. Write notes on arithmetic pipeline

Pipelining is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other

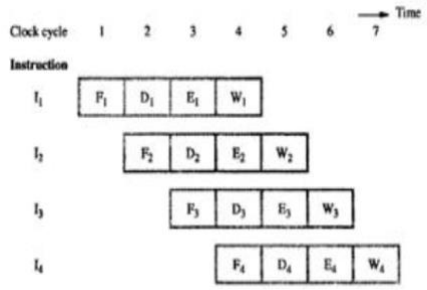
segments. A pipeline can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. A pipeline processor may process each instruction in 4 steps:

F Fetch: Read the instruction from the memory

D Decode: Decode the instruction and fetch the source operands

E Execute: Perform the operation specified by the instruction

W Write: Store the result in the destination location.



9. Briefly explain the role of micro program sequence.

Micro program sequencer is a control unit which does the tasks of Micro-program sequencing.There are two important factors must be considered while designing the micro programsequencer.

Micro-program sequencer is attached to the control memory.It inspects certain bits in the microinstruction to determine the next address for controlmemory.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.

10. Differentiate between horizontal and vertical micro instructions

Horizontal Micro-Instructions

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. However,

in extreme cases, each horizontal microinstruction controls all the hardware resources of the system.

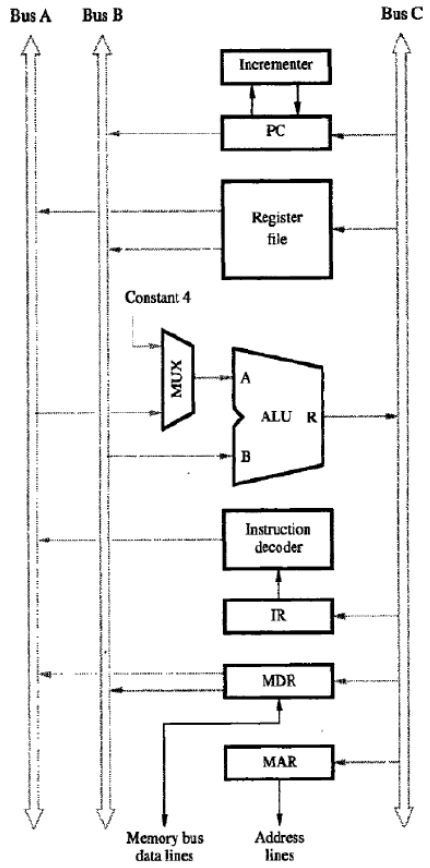
Vertical Micro-Instructions

We can reduce the length of the horizontal micro-instruction so easily by implementing another method known as vertical micro-instructions. In this case, Most signals are not needed simultaneously and many others are mutually exclusive. In a vertical microinstruction, a code is used for each action to be performed and the decoder translates this code into individual control signals. The vertical microinstruction resembles the conventional machine language format comprising one operation and a few operands.

11.(a) What is the significance of addressing modes in computer architecture

The addressing modes help us specify the way in which an operand's effective address is represented in any given instruction. Some addressing modes allow referring to a large range of areas efficiently, like some linear array of addresses along with a list of addresses. The addressing modes describe an efficient and flexible way to define complex effective addresses.

11.(b) Write the control sequence for the instruction DIV R1,[R2] in a three bus structure



Buses A and B are used to transfer the source operands to the A and B inputs of the ALU, where an arithmetic or logic operation may be performed. The result is transferred to the destination over bus C. If needed, the ALU may simply pass one of its two input operands unmodified to bus C. We will call the ALU control signals for such an operation R=A or R=B. The three-bus arrangement obviates the need for registers Y and Z.

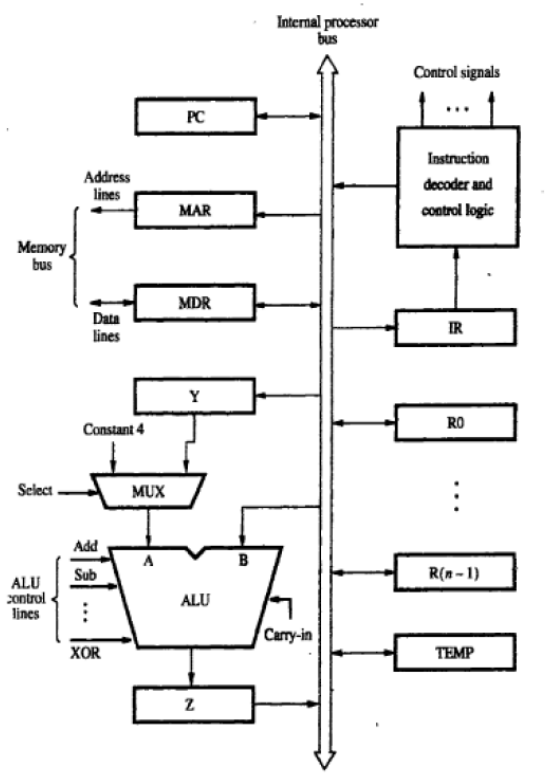
A second feature is the introduction of the Incrementer unit, which is used to increment the PC by 4. Using the Incrementer eliminates the need to add 4 to the PC using the main ALU. The source for the constant 4 at the ALU input multiplexer is still useful. It can be used to increment other addresses, such as the memory addresses in LoadMultiple and StoreMultiple instructions.
Consider the three-operand instruction

Add R4,R5,R6

The control sequence for executing this instruction is given as below

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

12. Explain the concept of a single bus organization with help of a diagram. Write the control sequence for the instruction ADD [R1],[R2].
Figure shows the organization in which the arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus. This bus is internal to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices.



The data and address lines of the external memory bus are connected to the internal processor bus via the memory data register, MDR, and the memory address register, MAR, respectively. Register MDR has two inputs and two outputs. Data may be loaded into MDR either from the memory bus or from the internal processor bus. The data stored in MDR may be placed on either bus. The input

of MAR is connected to the internal bus, and its output is connected to the external bus. The control lines of the memory bus are connected to the instruction decoder and control logic block. Three registers Y, Z, and TEMP registers are used by the processor for temporary storage during execution of some instructions. The multiplexer MUX selects either the output of register Y or a constant value 4 to be provided as input A of the ALU. The constant 4 is used to increment the contents of the program counter.

Consider the instruction ADD [R1],[R2]. which adds the contents of a memory location pointed to by R1 to contents of a memory location pointed to by R2 and stores the result in a memory location pointed to by R2.

The control sequence for executing this instruction is given as below

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Zin
2	Zout, PCin, Yin, WMFC
3	MDRout, IRin
4	R1out, MARin, Read
5	WMFC
6	MDRout, Yin
7	R2out, MARin, Read
8	WMFC
9	MDRout, SelectY, Add, Zin
10	R2out, MARin, Zout, Write, WMFC

Executing this instruction requires the following actions:

1. Fetch the instruction.

2. Fetch the operands (the contents of the memory location pointed to by R1 and R2).

3. Perform the addition.

4. Load the result into memory pointed by R2.

Instruction execution proceeds as follows.

Step 1: The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to selectthe constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z.

Step 2: The updated value is moved from register Z back into the PC, while waiting for the memory to respond.

Step 3:The word fetched from the memory is loaded into the IR.

Step 4: The instruction decoding circuit interprets the contents of the IR. This enables the control circuitry to activate the control signals for remaining steps, which constitute the execution phase. The contents of register R1 are transferred to the MAR in step 4, and a memory read operation is initiated.

Step 5: wait for memory read operation to complete.

Step 6: When the Read operation is completed, the memory operand is available in register

MDR, and the addition operation is performed. The contents of MDR are gated to the bus, and

thus also to the B input of the ALU, and store in register Y.

Step 7: Store the content of R2 in MAR and initiate memory read operation

Step 8: wait for memory read operation to complete.

Step 9: out the content of MDR which is the data read from the memory specified by the address in R2. Select the other operand from Y. Add signal is generated and the result is stored in Zin

Step 10: The result is stored in the address specified by the content of R2.

13. Explain various register transfer logics.

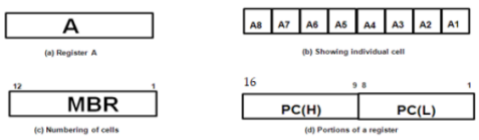
1)inter register transfer micro-operations

• These micro-operations do not change the information content when the binary information moves from one register to another.

• The registers in a digital system are designated by capital letters. Examples A, B, R1, R2 etc.

• The register that holds the address of the memory unit is called Memory Address Register (MAR). The cells of an n bit register are numbered in sequence from 1 to n, starting either from the left or from the right.

• A register can be represented in 4 ways which is shown in the figure.



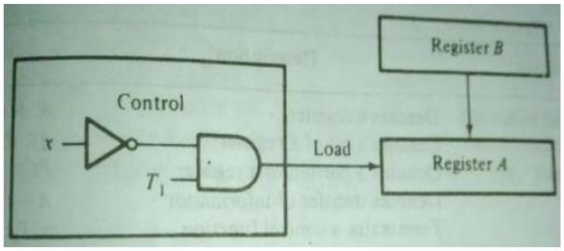
1. Figure (a) shows the most common way to represent a register is with a rectangular box in which name of the register is specified within the box.
2. In figure (b), it shows the individual cells of a register is assigned a letter with a subscript number and is marked from right to left.
3. The numbering of cells from right to left can be marked on top of the box as in figure (c).
4. A 16 bit register is partitioned into two parts, one high order part (H) consisting of eight high ordered cells and one lower order part (L) consisting of eight low ordered cells as in figure (d).

Conditional transfer

• The condition that determines when the transfer is to occur is called a control function. A control function is a Boolean function that can be equal to 0 or 1. The control function can be specified with the following statement: x'T1: A

• It means that the transfer operation be executed by the hardware only when the Boolean function x'T1=1.That is, variable x=0 and timing variable T1=1.

• Example: Hardware implementation of a controlled transfer x'T1: A B transfers the contents of register B to register A as shown in figure. Contents of source register B do not change after the transfer.



The basic symbols of the register transfer logic are listed in the following table.

Symbol	Description	Examples
Letters(and numerals)	Denotes a register	A,MBR,R2
Subscript	Denotes a bit of register	A_2,B_6
Parenthesis ( )	Denotes a portion of a register	PC(H),MBR(OP)
Arrow ←	Denotes transfer of information	A←B
Colon :	Terminates a control function	$x'T_1$:
Comma ,	Separates two micro operations	A←B,B←A
Square brackets [ ]	Specifies an address for memory transfer	MBR←M[MAR]

14.(a) Design a 4 bit combinational logic shifter with 2 control signals H1 and H2 that perform the following operations (bit values given in parenthesis are the values of control variable H1 and H2 respectively.) : Transfer of 0's to S (00), shift right (01), shift left (10), no shift (11).

The shift unit attached to the processor transfers the output of the ALU onto the output bus. Shifter may function in 4 different ways.

1. The shifter may transfer the information directly without a shift.

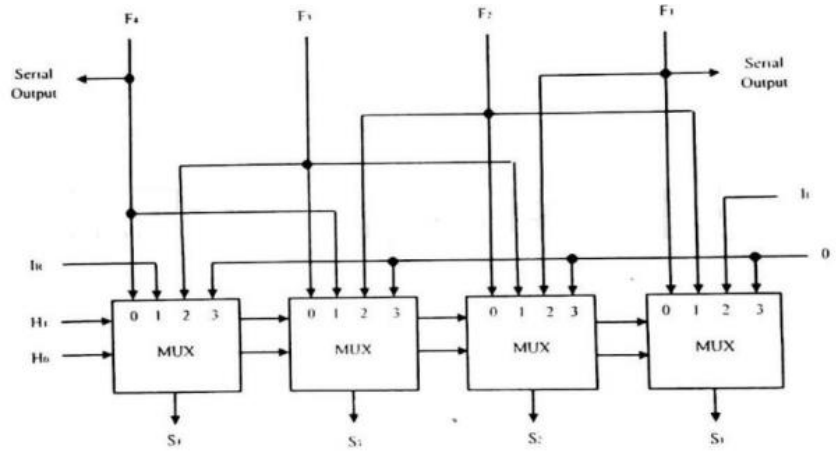
2. The shifter may shift the information to the right.

3. The shifter may shift the information to the left.

4. In some cases no transfer is made from ALU to the output bus.

• A shifter is a bi-directional shift-register with parallel load. The information from ALU can be transferred to the register in parallel and then shifted to the right or left.

• A combinational logic shifter can be constructed with multiplexers. The following figure will show the same.



In this configuration, a clock pulse is needed for the transfer to the shift register, and another pulse is needed for the shift. Another clock pulse may also in need of when information is passed from shift register to destination register.

The number of clock pulses may reduce if the shifter is implemented with a combinational circuit. In such cases, only one clock pulse is required to transfer from source register to destination register. In a combinational logic shifter, the signals from the ALU to the output bus propagate through gates without the need for clock pulse. Shifter operation can be selected by two variables H1H0.

1. If H1 H0=00, no shift is executed and the signals from F go directly to the S lines.

2. If H1 H0=01, shift right is executed

3. If H1 H0=10, shift left is executed.

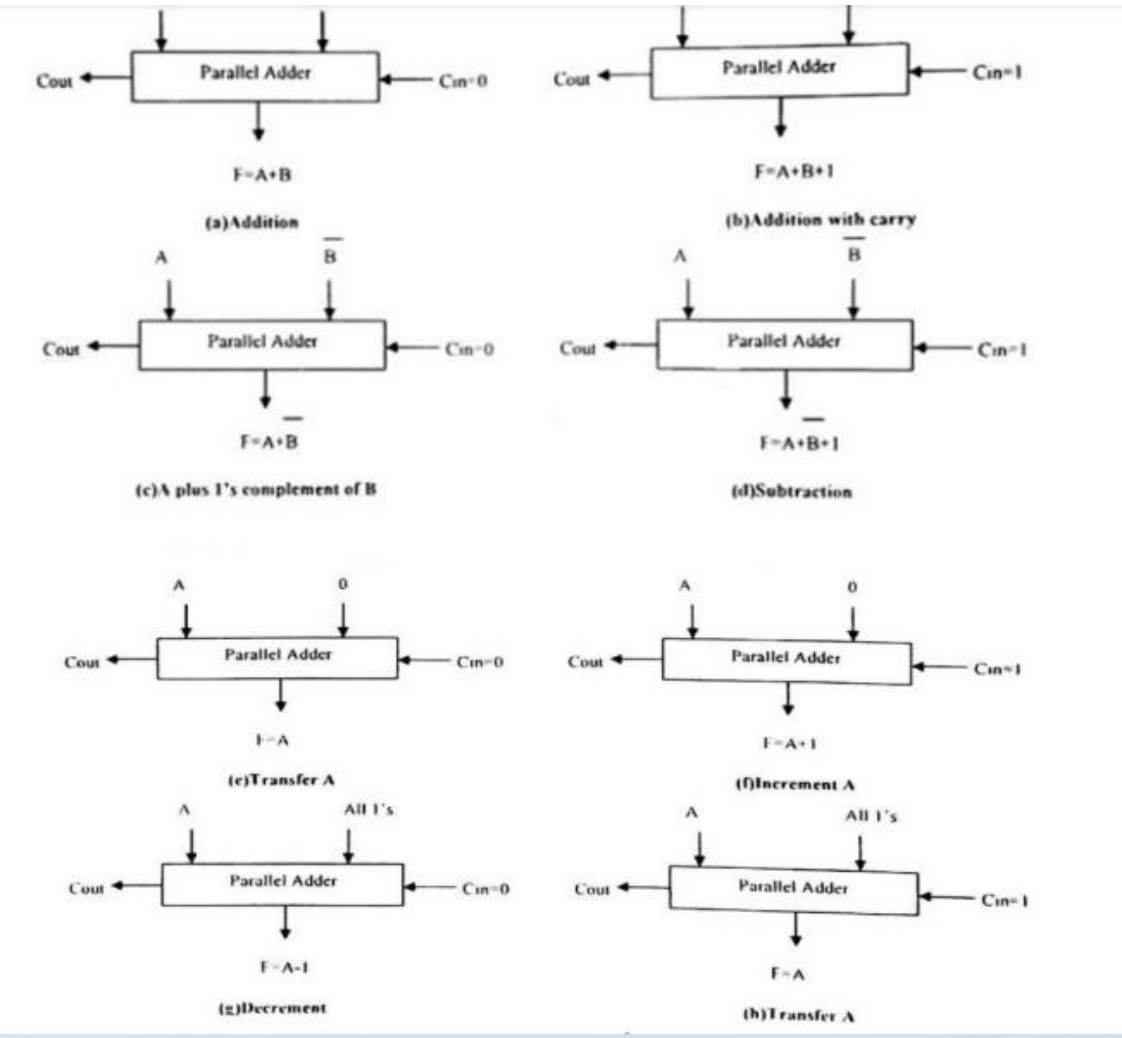
4. If H1 H0=11, no operation

The following summarizes the operation of a shifter.

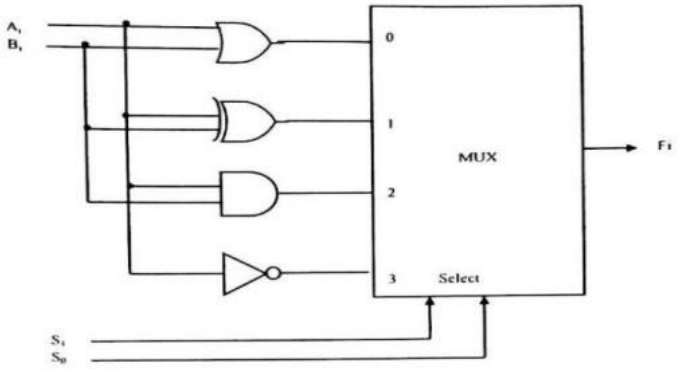
H1	H0	Operation	Function
0	0	S←F	Transfer F to S (no shift)
0	1	S←shr F	Shift right F into S
1	0	S←shl F	Shift left F into S
1	1	S←0	Transfer 0's into S

14.(b) Design an ALU unit which will perform arithmetic and logic operation with a given binary adder.

The basic component of arithmetic section of an ALU is a parallel adder. Parallel adder is constructed with a number of full adder circuits connected in cascade. By combining the data inputs to the parallel adder, it is possible to obtain different types of arithmetic operations. The following figure demonstrates the arithmetic operations obtained when one set of inputs to the parallel adder is controlled externally. The no of bits in the parallel adder may be of any value.



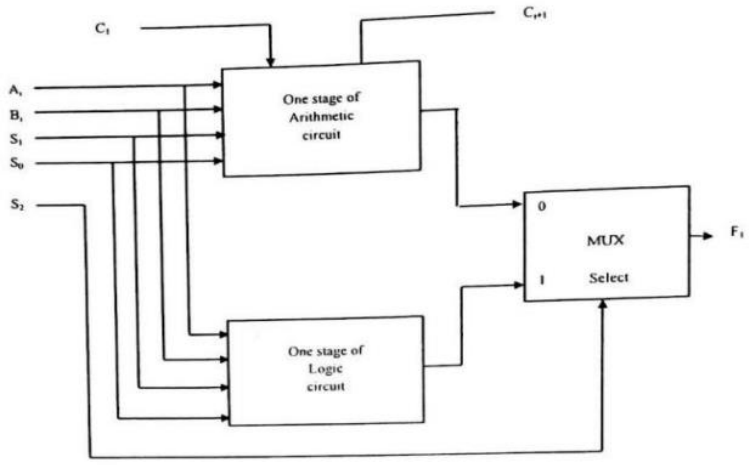
The logic microoperation manipulates bits of operands separately and treat each bit as a binary variable. The 16-logic operation can be generated in one circuit and selected by means of four selection lines.



All logic operations can be obtained by means of AND, OR and NOT operations. For three operations, we need two selection variables. But, with two selection variables, we can select four operations. So we can include one more operation XOR in our design. • The above diagram shows design of a logic circuit. In this one typical stage designated by subscript i. The circuit must be repeated n times for an n bit logic circuit. • The four gates generate the four logic operations OR, XOR, AND and NOT. The two selection variables in the multiplexer select one of the gates for the output. • The function table lists the output logic generated as a function of two selection variables.

S1	S0	Output	Operation
0	0	Fi=Ai+Bi	OR
0	1	Fi=Ai ⊕ Bi	XOR
1	0	Fi=AiBi	AND
1	1	Fi=Ai'	NOT

The logic circuit can be combined with the arithmetic circuit to produce one arithmetic logic unit.



Selection variables S1 and S0 can be made common to both sections provided we are using a third selection variable S2 to differentiate between the two. This configuration is illustrated in the following figure. The outputs of the logic and arithmetic circuits in each stage go through a multiplexer with selection variable S2. When S2=0, the arithmetic output is selected, but when S2= 1, the logic output is selected.