USN No.

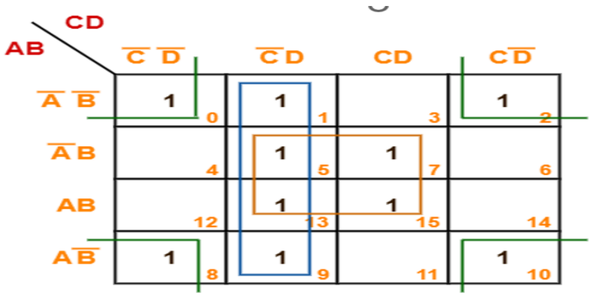
Internal Assessment Test 1– Nov 2024

Sub:	**Digital Design and Computer Organization**					Sub Code:	BCS302	Branch:	ISE
Date:	06-11-2024	Duration:	90 min's	Max Marks:	50	Sem / Sec:	03/ A,B,C		OBE

	Answer any FIVE FULL QUESTIONS	MARKS	CO	RBT
1	a. Minimize the following boolean function using K-Map- F (A, B, C, D) = Σm(0, 1, 2, 5, 7, 8, 9, 10, 13, 15) b. Minimize the following boolean function F (A, B, C, D) = Σm(0, 2, 8, 10, 14) + Σ d(5, 15)	5+5	CO1	L3
2	a. Minimize the following boolean function using K-Map- F (A, B, C, D) = π (3, 5, 7, 8, 10,11,12, 13) b. Minimize the following boolean function F (A, B, C, D) = π(0, 3,6,7)	5+5	CO1	L3
3	Implement the following boolean function using 8: 1 multiplexer, F (P,Q,R,S) = Σm (0, 1, 3, 4, 8, 9, 15).	10	CO2	L3
4	a. Explain SR Flip Flop in detail. b. Explain about Priority Encoder with its truth table.	5+5	CO2	L2
5	a. With a neat diagram, Explain the basic operational concepts of a computer. Give operating steps. b. Describe Big Endian and Little Endian methods of byte addressing?	5+5	CO3	L2
6	Explain different types of addressing Modes in detail	10	CO3	L2

Scheme and Solutions

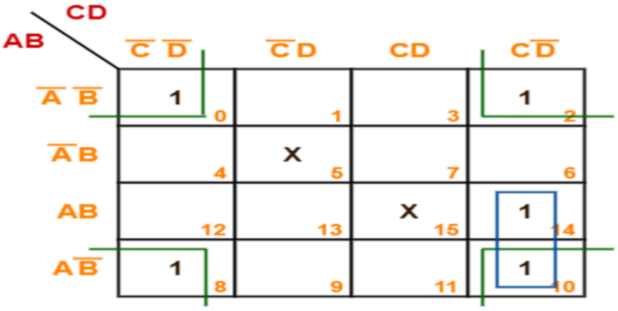
Sub:	**Digital Design and Computer Organization**	Sub Code:	BCS302	Branch:	ISE

	IAT-1	MARKS	CO	RBT
1	a.Minimize the following boolean function using K-Map- F (A, B, C, D) = Σm(0, 1, 2, 5, 7, 8, 9, 10, 13, 15) **Scheme:** K-Map-3 Marks Expression-2 Marks **Solution:**  Thus, minimized boolean expression is- **F(A, B, C, D) = BD + C'D + B'D'** b.Minimize the following boolean function F (A, B, C, D) = Σm(0, 2, 8, 10, 14) + Σd(5, 15) **Scheme:** K-Map-3 Marks	5+5	CO1	L3

<table>
<tr>
<td></td>
<td>

Expression-2 Marks

Solution:



Thus, minimized boolean expression is

F(A, B, C, D) = ACD' + B'D'

</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>

a.Minimize the following boolean function using K-Map-

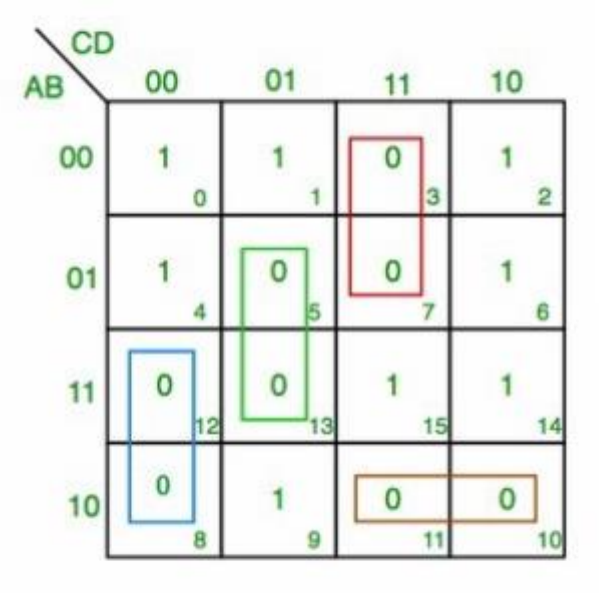
F (A, B, C, D) = π (3, 5, 7, 8, 10,11,12, 13)

Scheme:

K-Map-3 Marks

Expression-2 Marks

Solution:



Thus, minimized boolean expression is-

(C+D'+B').(C'+D'+A).(A'+C+D).(A'+B+C')

b.Minimize the following boolean function

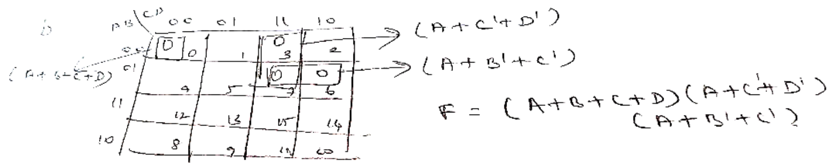
F (A, B, C, D) = π(0, 3,6,7)

Scheme:

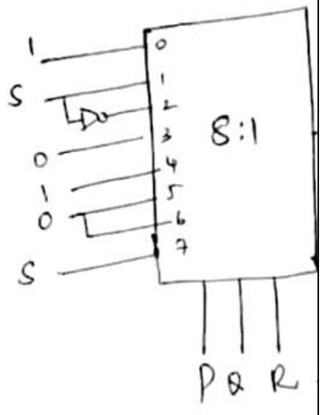
K-Map-3 Marks

Expression-2 Marks

</td>
<td>5+5</td>
<td>CO1</td>
<td>L3</td>
</tr>
</table>

	Solution:  AB\CD 00 01 11 10 (A+B+C+D) → (A+C'+D') → (A+B'+C') F = (A+B+C+D)(A+C'+D')(A+B'+C')			
3	a. Implement the following boolean function using 8: 1 multiplexer, F (P,Q,R,S) = Σm (0, 1, 3, 4, 8, 9, 15). **Scheme:** Explanation: 2 Marks Truthtable: 5 marks Diagram : 3 marks **Solution:** **Variables, n= 4 (P, Q, R, S)** **Select lines= n-1 = 3 (S2, S1, S0)** **2n-1 to MUX i.e., 23 to 1 = 8 to 1 MUX** **Input lines= 2n-1 = 23 = 8 (D0, D1, D2, D3, D4, D5, D6, D7)** Implementation table: Apply variables A and B to the select lines. The procedures for implementing the function are: i. List the input of the multiplexer ii. List under them all the minterms in two rows as shown below. The first half of the minterms is associated with A‘ and the second half with A. The given function is implemented by circling the minterms of the function and applying the following rules to find the values for the inputs of the multiplexer.	10	CO2	L3

	✓	✓	✓			
	P	Q	R	S	F	
0	0	0	0	0	1	F = 1
1	0	0	0	1	1	
2	0	0	1	0	0	F = S
3	0	0	1	1	1	
4	0	1	0	0	1	F = S'
5	0	1	0	1	0	
6	0	1	1	0	0	F = 0
7	0	1	1	1	0	
8	1	0	0	0	1	F = 1
9	1	0	0	1	1	
10	1	0	1	0	0	F = 0
11	1	0	1	1	0	
12	1	1	0	0	0	F = 0
13	1	1	0	1	0	
14	1	1	1	0	0.	F = S.
15	1	1	1	1	1	



4	Design SR and Priority Encoder in detail	5+5	CO2	L2

Scheme:

Explanation: 2 Marks

Truth table : 1 Marks

Circuit Diagram : 1 Marks

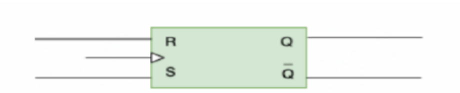
Excitation Table: 1Marks

Solution:

SR FLipflop:

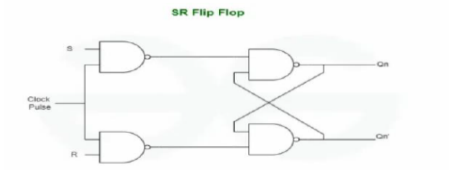
It is a Flip Flop with two inputs, one is S and the other is R. S here stands for Set and R here stands for Reset. Set basically indicates set the flip flop which means output 1 and reset indicates resetting the flip flop which means output 0. Here, a clock pulse is supplied to operate this flip-flop, hence it is a clocked flip-flop.

Block Diagram:

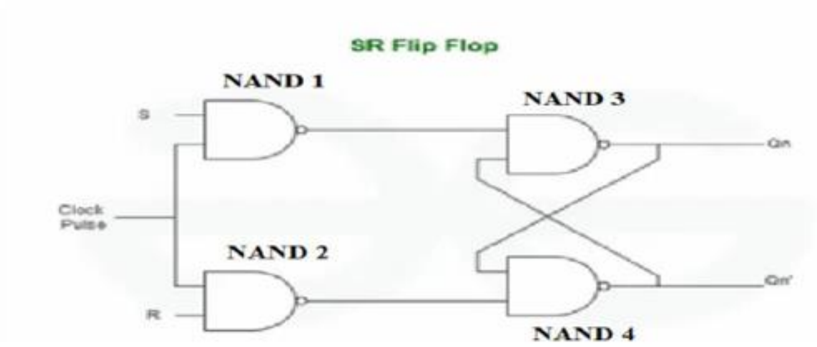


SR Flip Flop basic Block diagram

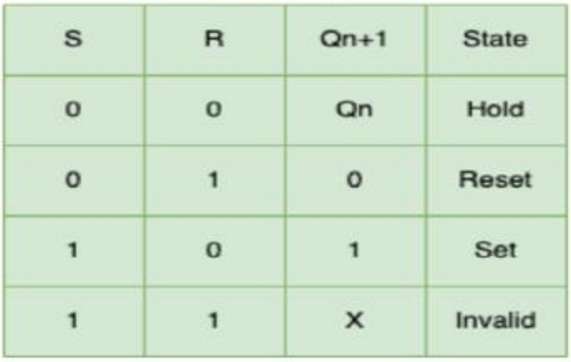
Logic Diagram:



Working of SR Flip Flop



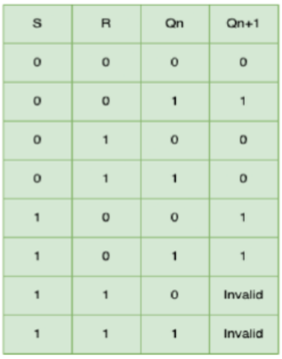
Truth table



S	R	Qn+1	State
0	0	Qn	Hold
0	1	0	Reset
1	0	1	Set
1	1	X	Invalid

Characteristic Table

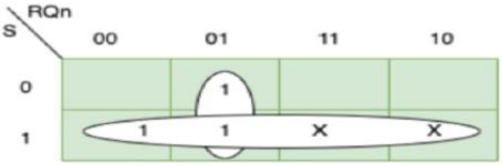
Here, S is the Set input, R is the reset input, Q_n is the previous output and Q_{n+1} is the present output.



S	R	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Invalid
1	1	1	Invalid

Characteristic Equation:

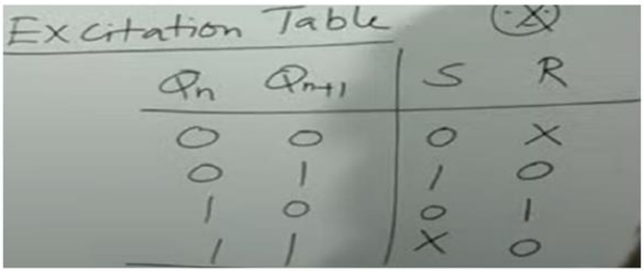
The characteristic equation tells us about what will be the next state of flip flop in terms of present state.



S \ RQn	00	01	11	10
0		1		
1	1	1	X	X

The characteristic equation for SR Flipflop will be

$Q_{n+1} = S + Q_nR'$



Excitation Table

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Priority Encoder

Scheme:

Explanation: 2 Marks

Truth table : 2 Marks

Circuit Diagram : 1 marks

Solution:

a. A priority encoder is an encoder circuit that includes the priority function.

b. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

c. The truth table of a four-input priority encoder is given in Table 4.8

ii. List under them all the minterms in two rows as shown below.

The first half of the minterms is associated with A‘ and the second half with A. The given function is implemented by circling the minterms of the function and applying the following rules to find the values for the inputs of the multiplexer.

1. If both the minterms in the column are not circled, apply 0 to the corresponding input.
2. If both the minterms in the column are circled, apply 1 to the corresponding input.
3. If the bottom minterm is circled and the top is not circled, apply C to the input.
4. top minterm is circled and the bottom is not circled, apply C‘ to the input.

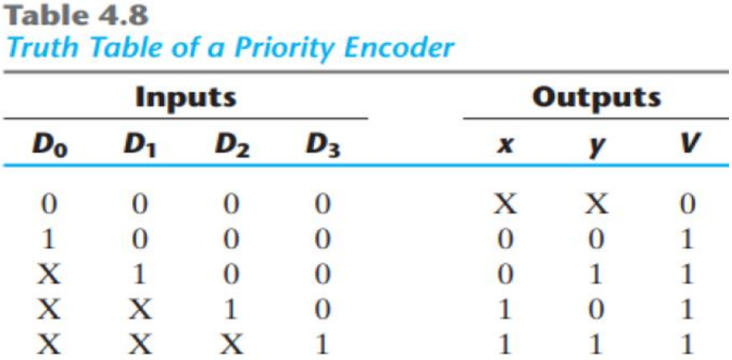


Table 4.8
Truth Table of a Priority Encoder

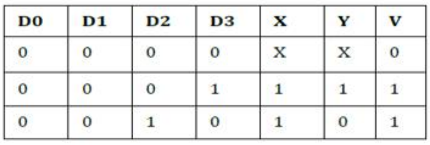
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

In addition to the two outputs x and y, the circuit has a third output designated by V ; this is a valid bit indicator that is set to 1 when one or more inputs are equal to 1.

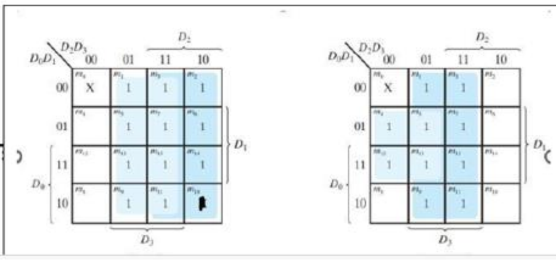
• . If all inputs are 0, there is no valid input and V is equal to 0. The other two outputs are not inspected when V equals 0 and are specified as don't-care conditions.

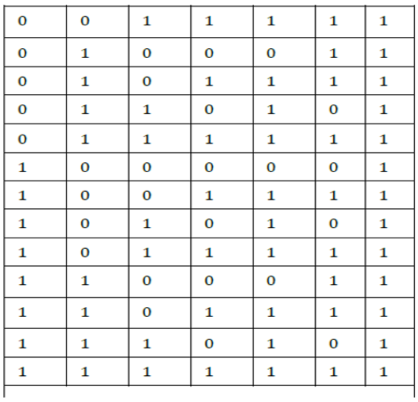
• Input D3 has the highest priority, so, regardless of the values of the other inputs, when this input is 1, the output for xy is 11 (binary 3).

• D2 has the next priority level. The output is 10 if D2 = 1, provided that D3 = 0, regardless of the values of the other two lower priority inputs. The output for D1 is generated only if higher priority inputs are 0.

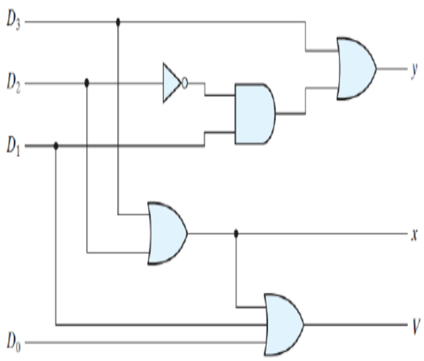


D0	D1	D2	D3	X	Y	V
0	0	0	0	X	X	0
0	0	0	1	1	1	1
0	0	1	0	1	0	1





0	0	1	1	1	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1



5	a. With a neat diagram, Explain the basic operational concepts of a computer. Give operating steps. **Scheme:** Explanation: 3 Marks Diagram: 2 marks	5+5	CO3	L2

Solution:

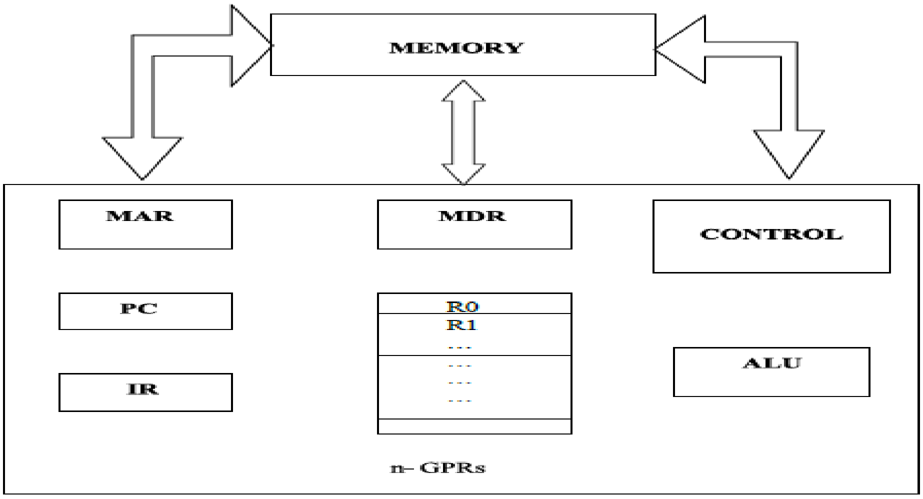


Fig b : Connections between the processor and the memory

The fig shows how memory & the processor can be connected. In addition to the ALU & the controlcircuitry, the processor contains a number of registers used for several different purposes.

The instruction register (IR):- Holds the instructions that is currently being executed. Its output is available for the control circuits which generates the timing signals that control the various processing elements in one execution of instruction.

The program counter PC:-

This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed.

Besides IR and PC, there are n-general purpose registers R0 through R_{n-1}.

The other two registers which facilitate communication with memory are: -

1. **MAR – (Memory Address Register):-** It holds the address of the location to beaccessed.
2. **MDR – (Memory Data Register):-** It contains the data to be written into or readoutof the address location.

Operating steps are

1. Programs reside in the memory & usually get these through the I/P unit.
2. Execution of the program starts when the PC is set to point at the first instructionofthe program.
3. Contents of PC are transferred to MAR and a Read Control Signal is sent to thememory.
4. After the time required to access the memory elapses, the address word is read outof the memory and loaded into the MDR.
5. Now contents of MDR are transferred to the IR & now the instruction is ready tobe decoded and executed.
6. If the instruction involves an operation by the ALU, it is necessary to obtain therequired operands.
7. An operand in the memory is fetched by sending its address to MAR & Initiatingaread cycle.
8. When the operand has been read from the memory to the MDR, it is transferredfrom MDR to the ALU.
9. After one or two such repeated cycles, the ALU can perform the

desiredoperation.

10. If the result of this operation is to be stored in the memory, the result is sent toMDR.
11. Address of location where the result is stored is sent to MAR & a write cycle isinitiated.
12. The contents of PC are incremented so that PC points to the next instruction thatisto be executed.

b. Describe Big Endian and Little Endian methods of byte addressing?

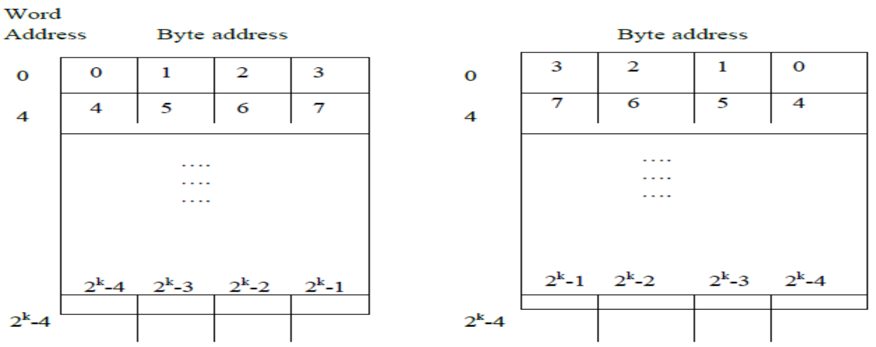
Scheme:

Explanation 2 Marks

Diagram: 3 marks

Solution:

There are two ways that byte addresses can be assigned across words, as shown in fig



(a) Big-endian assignment (b) Little-endian assignment

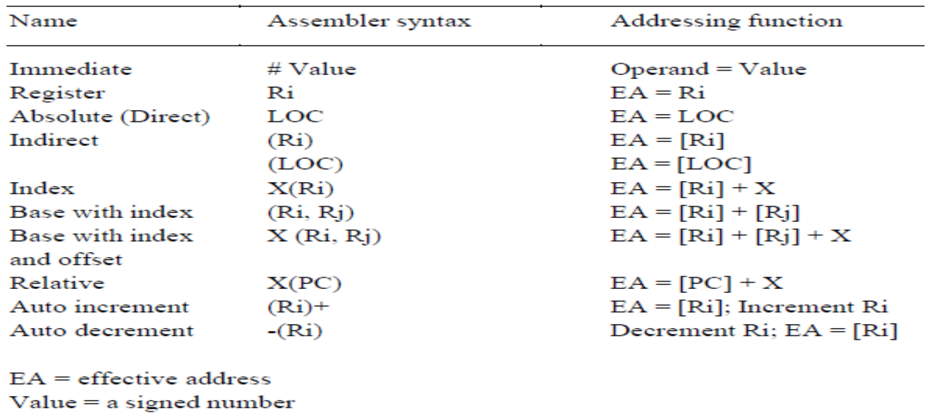
The name big-endian is used when lower byte addresses are used for the more significant bytes (the leftmostbytes) of the word. The name little-endian is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word. In addition to specifying the address ordering of bytes within a word, it is also necessary to specify the labeling of bits within a byte or a word. The same ordering is also used for labeling bits within a byte, that is, b_7, b_6, …., b_0, from left to right.

6 Explain different types of addressing Modes in detail 10 CO3 L2

Scheme:

Explanation with diagram for each modes: 1 Marks

Solution:



Name	Assembler syntax	Addressing function
Immediate	# Value	Operand = Value
Register	Ri	EA = Ri
Absolute (Direct)	LOC	EA = LOC
Indirect	(Ri)	EA = [Ri]
	(LOC)	EA = [LOC]
Index	X(Ri)	EA = [Ri] + X
Base with index	(Ri, Rj)	EA = [Ri] + [Rj]
Base with index and offset	X (Ri, Rj)	EA = [Ri] + [Rj] + X
Relative	X(PC)	EA = [PC] + X
Auto increment	(Ri)+	EA = [Ri]; Increment Ri
Auto decrement	-(Ri)	Decrement Ri; EA = [Ri]

EA = effective address

Value = a signed number

In general, a program operates on data that reside in the computer's memory. These

1.Immediate mode:

The operand is given explicitly in the instruction.

Syntax: #Value

Example : Move #200, R0

This instruction places the value 200 in register R0. Clearly, the Immediate mode is only used to specify the value of a source operand.

2.Register mode:

The operand is the contents of a processor register; the name of the register is given in the instruction.

Syntax: Ri

Example : Move R2, R1

This instruction moves the content of Register R2 to Register R1

3.Absolute or Direct mode:

The operand is in a memory location; the address of this location is given explicitly in the instruction.

Syntax: LOC

Example: Move LOC, R2

This instruction moves the content in memory location LOC to Register R2

4.Indirect mode:

The effective address of the operand is the contents of a register that is specified in the instruction.

Syntax: (Ri)

Example: Add (R1),R0

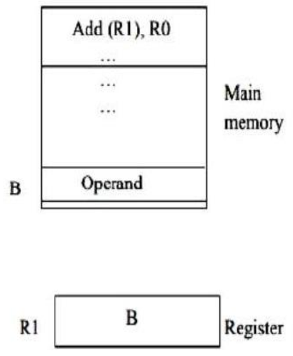


Figure 3.13: Indirect addressing

To execute the Add instruction in Figure 3.13, the processor uses the value which is in **register R1, as the effective address** of the operand.

It requests a read operation from the memory to **read the contents of location B.** the value read is the **desired operand,** which the processor adds to the contents of register R0 and the resultant sum is stored in R0.

5.Index mode:

The effective address of the operand is generated by adding a constant value to the contents of a register.

Syntax: X(Ri)

Where X denotes the constant value contained in the instruction and Ri is the name of the register involved. The **effective address** of the operand is given by **EA = X + [Ri].**

Example: Add 20(R1),R2

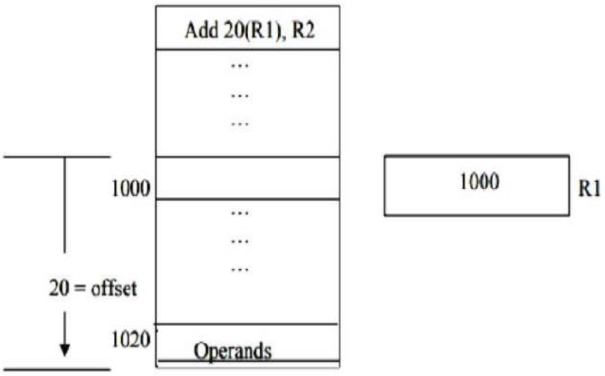


Figure 3.14 Indexed addressing

Figure 3.14 Indexed addressing

In Figure 3.14, the index register, R1, contains the address of a memory location, and the value 20 defines an offset (also called a displacement) from this address to the location where the operand is found. This operand is added to the content in Register R2 and the resultant sum is placed inside Register R2.

6. Base with Index mode

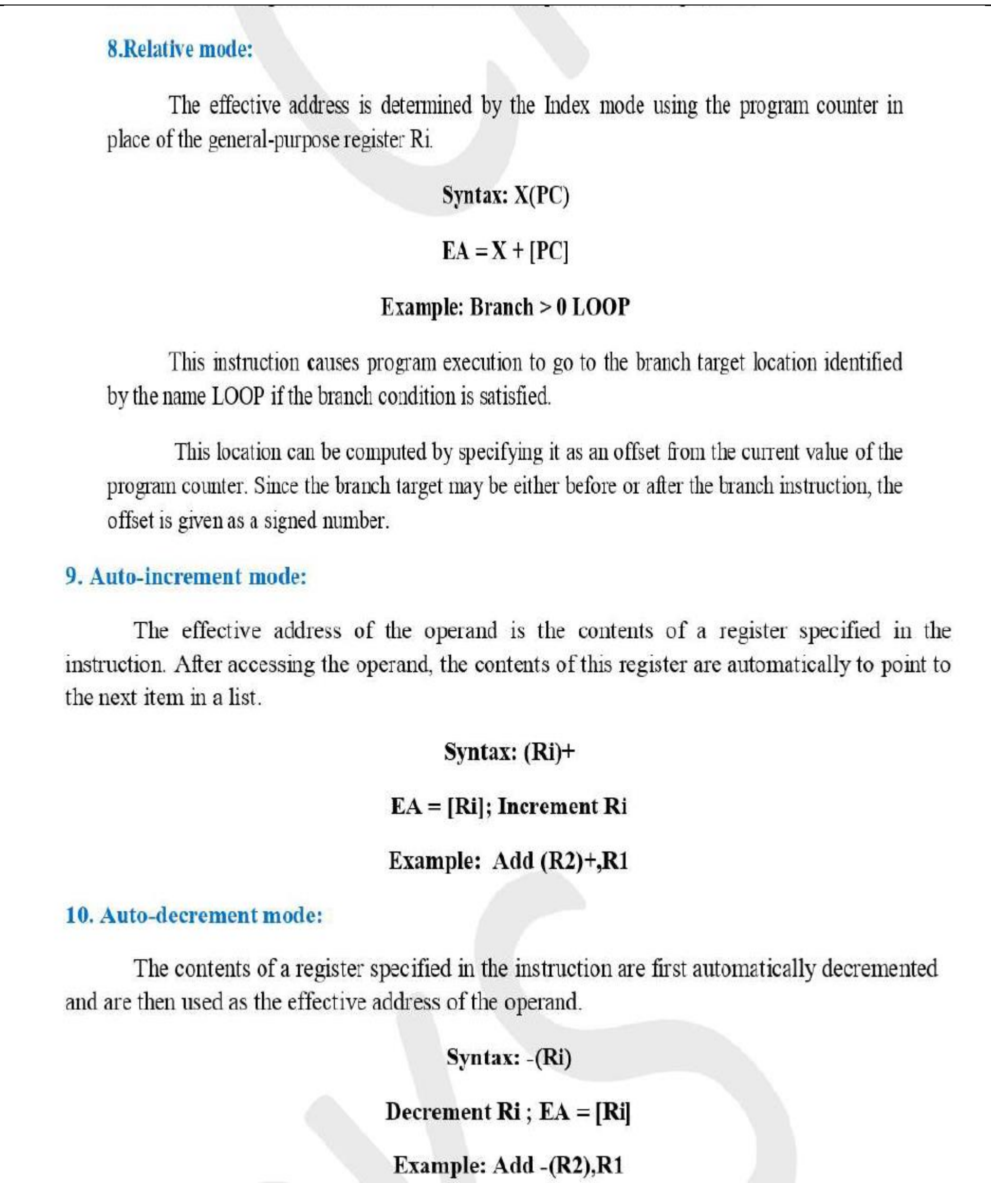
The effective address of the operand is sum of the contents of registers Ri and Rj.

Syntax: (Ri,Rj)

EA = [Ri]+ [Rj]

Example: Add (R1,R2),R3

Here, the index registers, R1 and R2, contains the address of a memory location, which is summed up to give the address to the location where the operand is found. This operand is added to the content in Register R3 and the resultant sum is placed inside Register R3.

	 **8.Relative mode:** The effective address is determined by the Index mode using the program counter in place of the general-purpose register Ri. **Syntax:** X(PC) **EA = X + [PC]** **Example: Branch > 0 LOOP** This instruction causes program execution to go to the branch target location identified by the name LOOP if the branch condition is satisfied. This location can be computed by specifying it as an offset from the current value of the program counter. Since the branch target may be either before or after the branch instruction, the offset is given as a signed number. **9. Auto-increment mode:** The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically to point to the next item in a list. **Syntax: (Ri)+** **EA = [Ri]; Increment Ri** **Example: Add (R2)+,R1** **10. Auto-decrement mode:** The contents of a register specified in the instruction are first automatically decremented and are then used as the effective address of the operand. **Syntax:** -(Ri) **Decrement Ri** ; **EA = [Ri]** **Example: Add -(R2),R1**			

CI CCI HOD