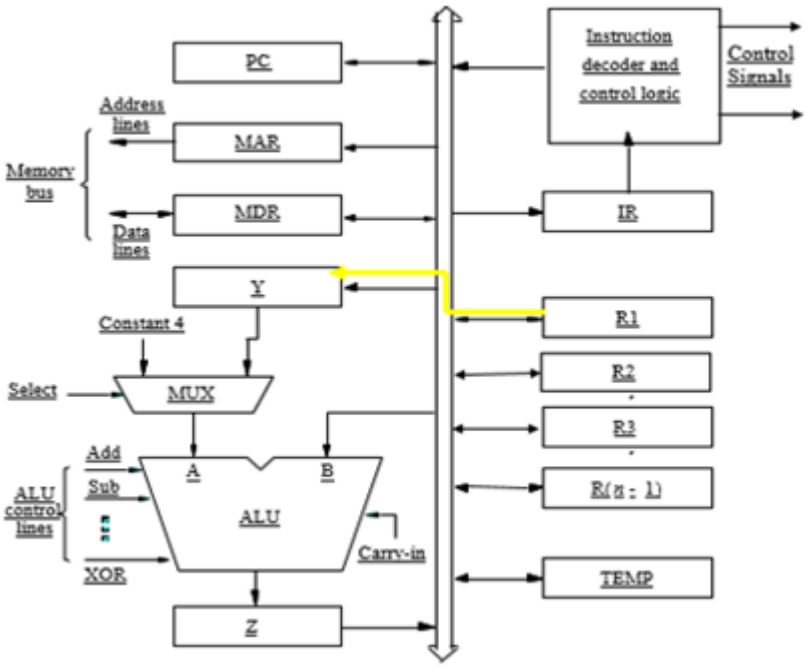
| CMR INSTITUTE OF TECHNOLOGY | |
|---|---|

| Sub: | COMPUTER ORGANIZATION | | |
|---|---|---|---|
| Scheme and solution | | Sem: | III |

| | |
|---|---|
| 1(a) | Draw and explain a single bus organization of the data path inside a processor.<br>Explanation:2.5<br>Diagram:2.5<br><br>- Single bus organization contains MAR, MDR registers. MDR has two inputs and two outputs. Data may be loaded into MDR wither from the memory bus or from the internal processor bus. Data stored in MDR may be place on either bus.<br>- The input of MAR is connected to the internal bus, and its output is connected to the external bus.<br>- The control lines of the memory bus are connected to the instruction decoder and control logic block.<br>- Three registers: Y, Z and Temp are used in this design.<br>- ALU must have only one input connection from the bus. The other input must be stored in a holding register called Y register.<br>- A multiplexer selects among register Y and 4 depending upon select line.<br>One operand of a two-operand instruction must be placed into the Y register before the other operand must be placed onto the bus. Identical reasoning tells us that there must be an output register Z which collects the output of the ALU at the end of each cycle.<br>- This way, there can be one operand in the Y register, one operand on the bus and the result stored in the Z register<br>- The register, ALU and the interconnecting bus are collectively referred as *datapath.*<br>- The following sequence are considered for instruction execution<br>- Transfer a word of data from one processor to another or to the ALU<br>- Perform an arithmetic or logic operation and store the result<br>- Fetch the contents of a given memory location and load them into processor register<br>- Store a word of data from a processor register into a given memory location. |

(b)

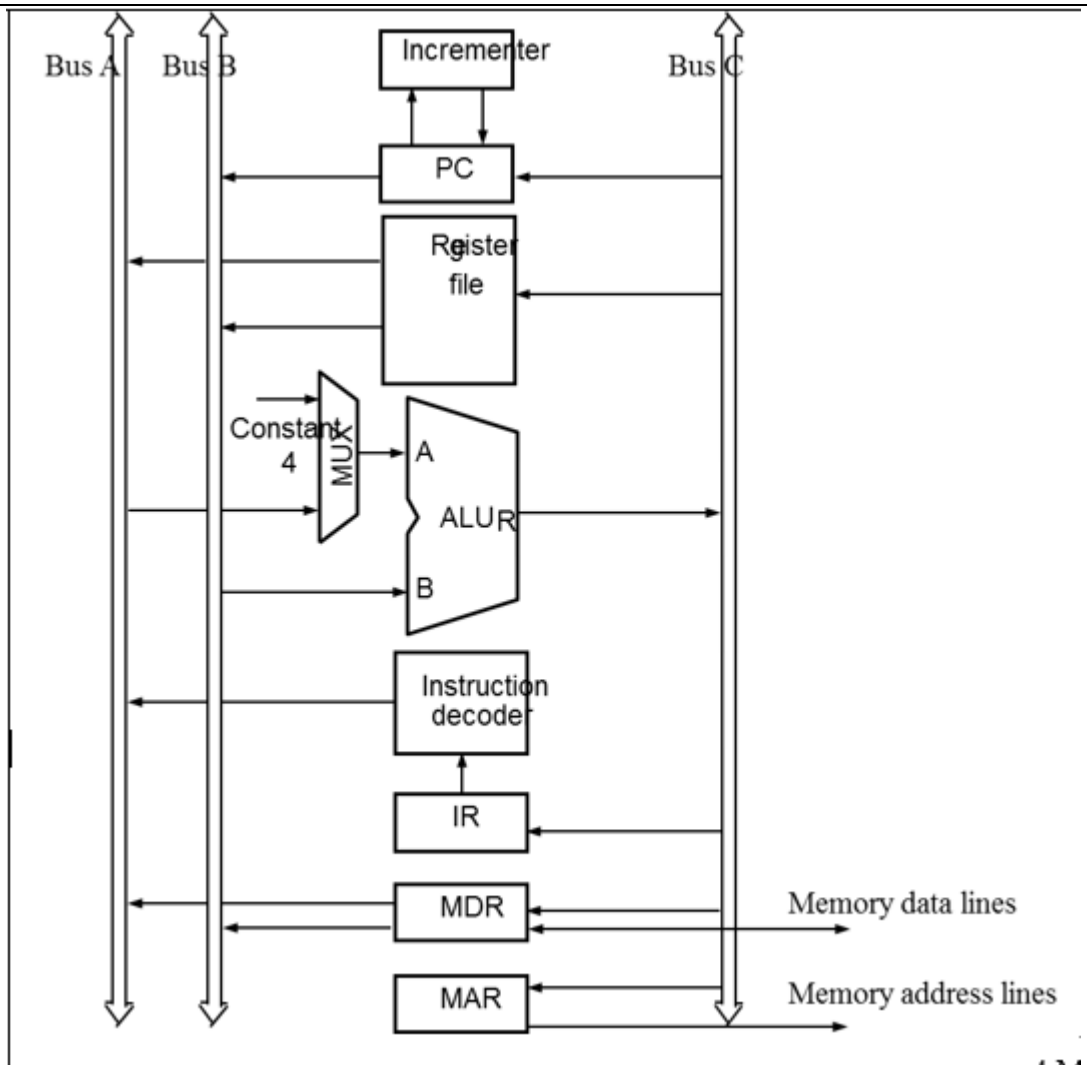



With a suitable diagram, explain about Three-bus organization data path inside the processor. Provide control sequence for ADD R4, R5, R6.

Diagram 2.5

Control sequence:2.5

| | |
|---|---|
| | - In simple single-bus structure, the results in long control sequences, because only one data item can be transferred over the bus in a clock cycle.<br>- Most commercial processors provide multiple internal paths to enable several transfers to take place in parallel.<br>- Three-bus organization to connect the registers and the ALU of a processor. All general-purpose registers are combined into a single block called register file.<br>- Register file has three ports. Two outputs ports connected to buses A and B, allowing the contents of two different registers to be accessed simultaneously, and placed on buses A and B. One input port allows the data on bus C to be loaded into a third register during the same clock cycle.<br>- Inputs to the ALU and outputs from the ALU - Buses A and B are used to transfer the source operands to the A and B inputs of the ALU. Result is transferred to the destination over bus C. |




Example: Add R4, R5, R6

| 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 |

- Control signals for such an operation are *R=A* or *R=B*. Three bus arrangement obviates the need for Registers *Y* and *Z* in the single bus organization.
- Incremental unit - Used to increment the PC by 4. Source for the constant 4 at the ALU multiplexer can be used to increment other addresses such as the memory addresses in multiple load/store instructions.

| | |
|---|---|
| | 1. Pass the contents of the *PC* through ALU and load it into *MAR*.<br>Increment *PC*.<br>2. Wait for *MFC*.<br>3. Load the data received into *MDR* and transfer to *IR through ALU*.<br>4. Execution of the instruction is the last step. |
| 2(a) | Explain the different actions required for execution of instruction ADD (R3), R1.Write the control sequence of it.<br>Explanation:2.5<br>Control sequence:2.5<br><br>To execute the instruction we must execute the following tasks:<br>1. Fetch the instruction.<br>2. Fetch the operand (contents of the memory location pointed to by *R3*.)<br>3. Perform the addition.<br>4. Load the result into *R1*.<br><br>**Step — Action**<br>**1** — $PC_{out}$, $MAR_{in}$, Read, Select4, Add, $Z_{in}$<br>**2** — $Z_{out}$, $PC_{in}$, $Y_{in}$, WMFC<br>**3** — $MDR_{out}$, $IR_{in}$<br>**4** — $R3_{out}$, $MAR_{in}$, Read<br>**5** — $R1_{out}$, $Y_{in}$, WMFC<br>**6** — $MDR_{out}$, SelectY, Add, $Z_{in}$<br>**7** — $Z_{out}$, $R1_{in}$, End |

Recall that: *PC* holds the address of the memory location which has the next instruction to be executed. *IR* holds the instruction currently being executed.

Step 1
- Load the contents of *PC* to *MAR*.
- Activate the *Read* control signal.
- Increment the contents of the *PC* by 4.
- $\boldsymbol{PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}}$.

Step 2
- Update the contents of the *PC*.
- Copy the updated *PC* to Register *Y* (useful for Branch instructions).
- Activate the control signal to load data from external bus to *MDR*
- Wait for *MFC* from memory.
- $\boldsymbol{Z_{out}, PC_{in}, Y_{in}, MDR_{inE}, WMFC}$

Step 3
- Place the contents of *MDR* onto the bus.
- Load the *IR* with the contents of the bus.
- $\boldsymbol{MDR_{out}, IR_{in}}$

Step 4: - Place the contents of Register *R3* onto internal processor bus.
- Load the contents of the bus onto *MAR*.
- Activate the *Read* control signal.
- $\boldsymbol{R3_{out}, MAR_{in}, Read}$

Step 5: - Place the contents of *R1* onto the bus.
- Load the contents of the bus into Register *Y (*Recall one operand in *Y)*
- Wait for *MFC*.
- $\boldsymbol{R1_{out}, Y_{in}, MDR_{inE}, WMFC}$

Step 6: - Load the contents of *MDR* onto the internal processor bus.
- Select *Y,* and perform the addition.
- Place the result in *Z*.
- $\boldsymbol{MDR_{out}, SelectY, Add, Z_{in}}$.

Step 7: - Place the contents of Register *Z* onto the internal processor bus.
- Place the contents of the bus into Register *R1*.
- $\boldsymbol{Z_{out}, R1_{in}}$

(b) What is instruction pipelining? What are the different types of hazards involved in it?

Definition+diagram:2.5

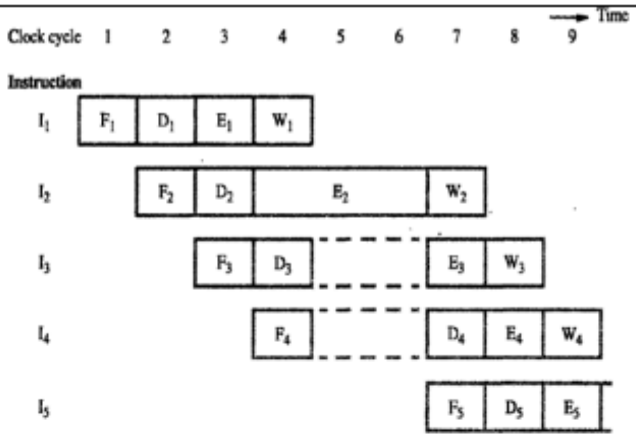
Hazards:2.5

Pipeline is a particularly effective way of organizing concurrent activity in a computer systems. Simple in design with assembly line operation.

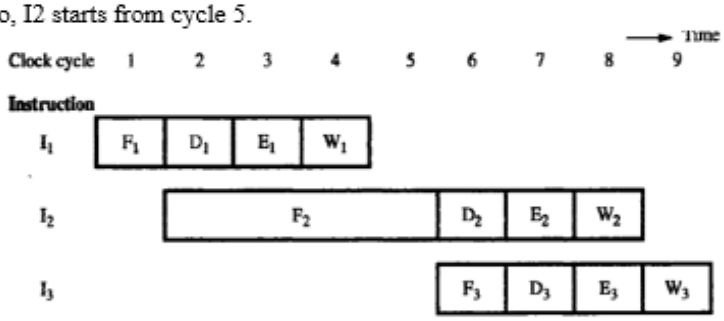
- *Pipeline Performance:*

i) Sometimes pipeline stages may not be able to complete its processing task for a given instruction in the allotted time.

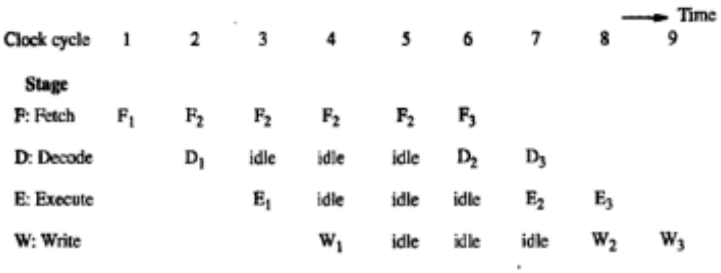
- As per the following example, I2 requires three cycle to complete, from cycle 4 through cycle 6. Thus, in cycle 5 and 6, the write stage must be told to do nothing, because it has no data to work with. This stalled clock cycle known as *hazard.*




ii) Cache miss on pipeline is given as, I1 is fetched from the cache in cycle 1, and its execution proceeds normally. But in I2, the result data missed in cache. Thus instruction fetch unit must now suspend any further fetch request and wait for I2 to arrive. So, I2 starts from cycle 5.



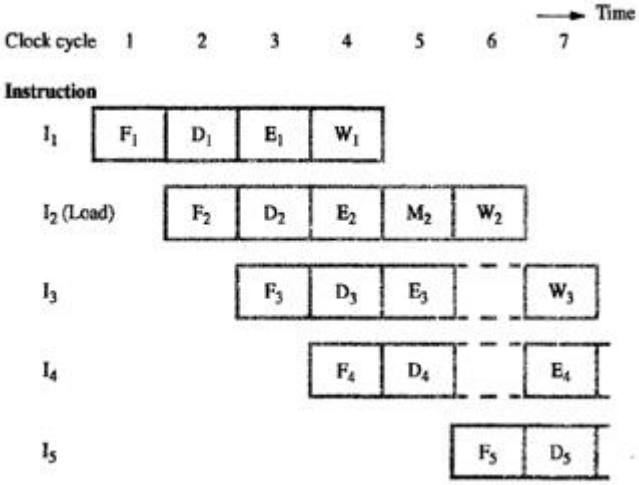

(a) Instruction execution steps in successive clock cycles



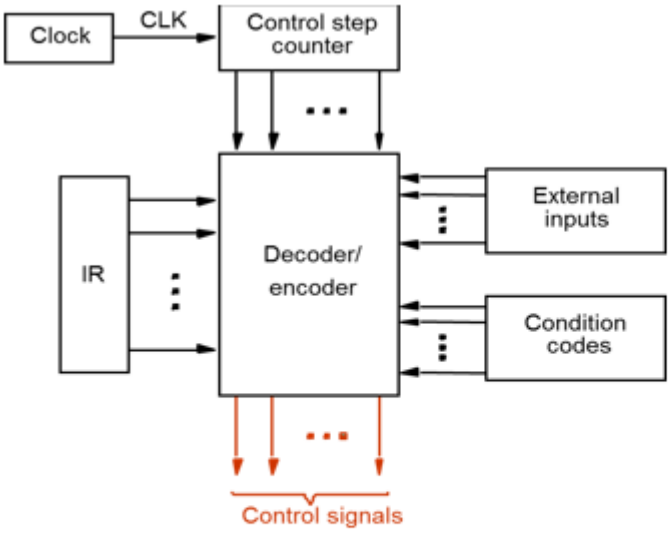

(b) Function performed by each processor stage in successive clock cycles

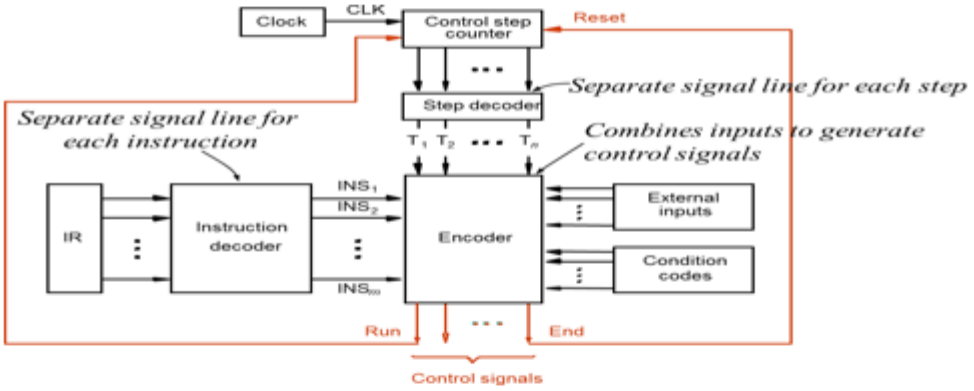
iii) Structural hazard – This situation will occur when two instruction require the use of a given hardware resource at the same time.
Example: LOAD X (R1), R2
- Four stage pipeline is used for this example. The memory address, X + [R1], is computed in step E2 in cycle 4, then memory access (M) takes place in cycle 5. The

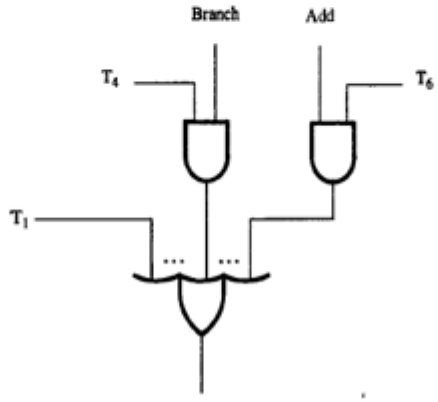
operand read from memory is written into R2 in cycle 6.



| | |
|---|---|
| 3 | Describe the function of Hardwired control unit with block diagram, decoding / encoding diagram and circuit diagram for control signal Zin.<br><br>- Required control signals are determined by the following information:<br>i) Contents of the control step counter. Determines which step in the sequence.<br>ii) Contents of the instruction register. Determines the actual instruction<br>iii) Contents of the condition code flags. Used for example in a BRANCH instruction.<br>iv) External input signals such as MFC.<br><br>- Control unit consists of a decoder/encoder block to accept the following inputs:<br>- Control step counter.<br>- Instruction Register. IR<br>- Condition codes<br>- External inputs. |

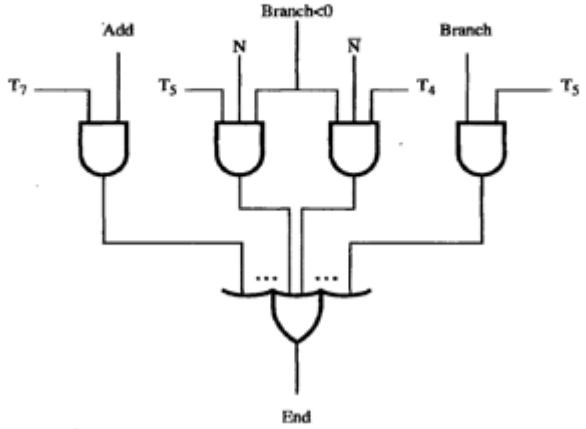



$$Z_{in} = T_1 + T_6 \cdot ADD + T_4 \cdot BR + \cdots$$

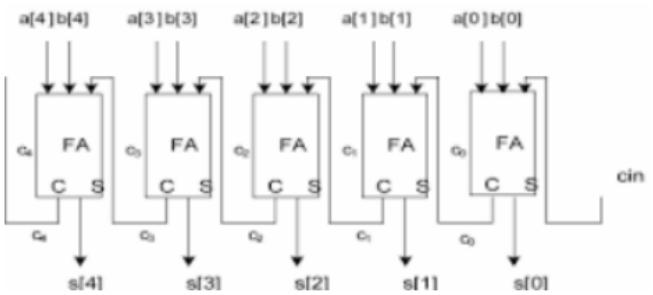
- Suppose if $Z_{in}$ is asserted:
  - During $T_1$ for all instructions.
  - During $T_6$ for *ADD* instruction.
  - During $T_4$ for unconditional *BRANCH* instruction




$$End = T_7 \cdot ADD + T_5 \cdot BR + (T_5 \cdot N + T_4 \cdot \overline{N}) \cdot BRN + \cdots$$

- In the above diagram, End signal starts a new instruction fetch cycle by resettling the control step counter to its starting value. It contains another signal called RUN. If it is active, the counter is incremented by one at the end of every clock cycle.
- When RUN = 0, the counter stops counting. This is needed whenever the WMFC signal is issued, to cause the processor to wait for the reply from the memory.
- Control hardware can be viewed as a state machine. Changes state every clock cycle depending on the contents of the instruction register, condition codes, and external inputs.
- Outputs of the state machine are control signals. Sequence of control signals generated by the machine is determined by wiring of logic elements, hence the name "hardwired control".
- Speed of operation is one of the advantages of hardwired control is its speed of operation.
- Disadvantages include: Little flexibility. Limited complexity of the instruction set it can implement

| | |
|---|---|
| 4(a) | Construct and explain a 5-bit carry lookahead adder along with proper equations and block diagram. |
| (b) | Prove the following using Binary Arithmetic<br>i) (-7) – (-5) ii) (+2) – (-3) |




3 Marks

Explanation

2 Marks

Calculate the following using binary subtraction

i) (-7) – (-5) ii) (+2) – (-3)

i) -7 = 1001 (1001)

-5 = 1011 (0101)

---

-2 = 1110

1.5 Marks

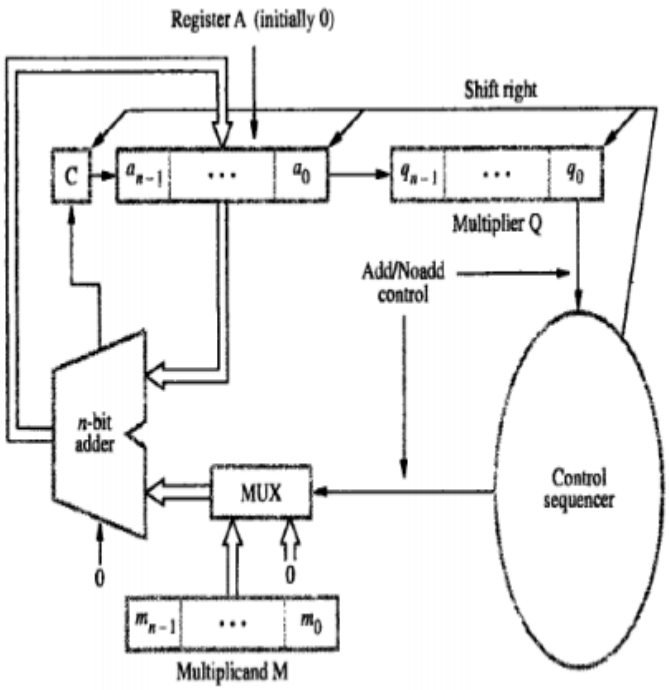
ii) +2 = 0010 (0010)

-3 = 1101 ( 0011)

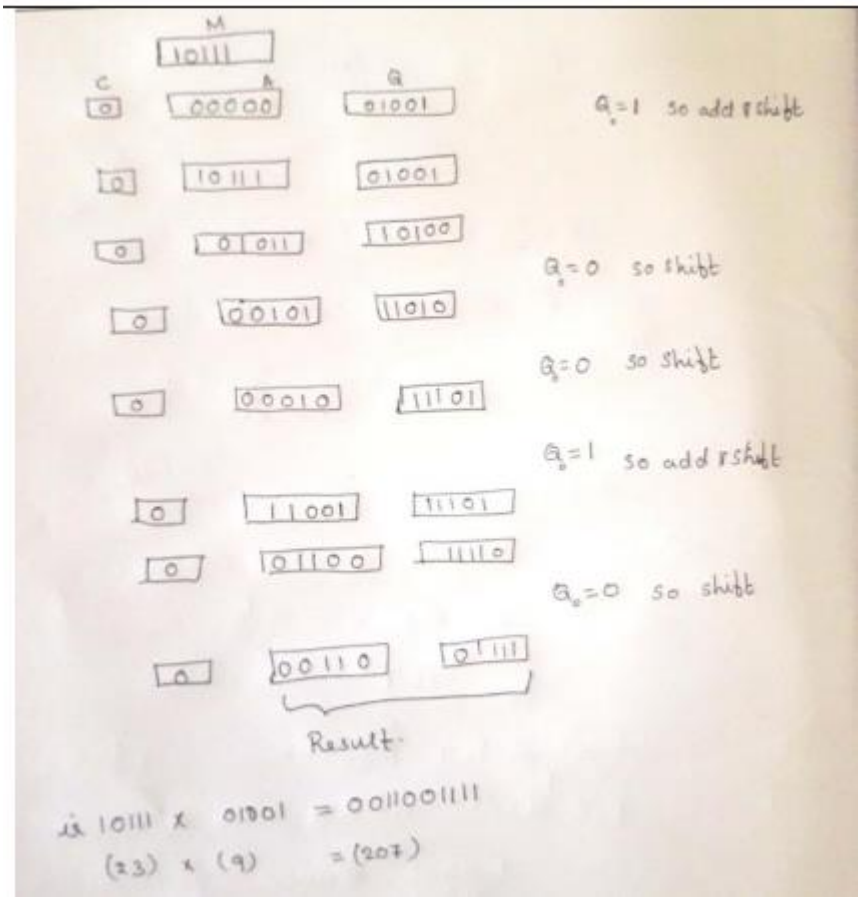
---

-5 = 0101

| | |
|---|---|
| 5(a) | Design a 5-bit register block diagram for sequential circuit binary multiplier. Prove the same with M=10111, Q=01001, C=0 and A=00000<br><br>Block diagram:2. Problem:3 |

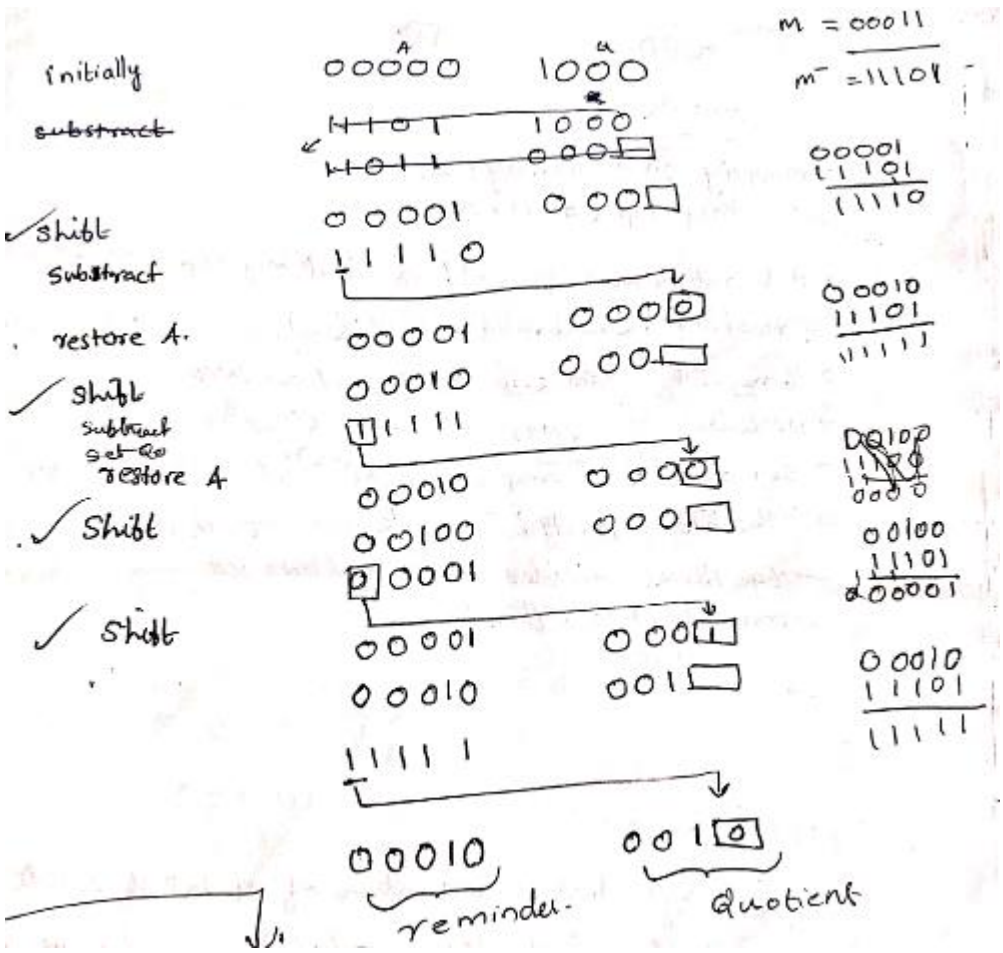



(a) Register configuration




(b) With neat diagram explain the circuit arrangement of binary division.Perform restoring division for 8÷3 by showing all the steps.

Problem:2.5

| | |
|---|---|
| | Circuit:2.5<br>  <br> |
| 6 | Prove the following booth algorithm's functionality<br>i) Basic formula /Encoding scheme table for Booth algorithm<br>ii) Booth recoding for 0100010100<br>iii) Booth multiplication for + 10 and -12 |

i) Basic formula /Encoding scheme table for Booth algorithm

| Multiplier | | Version of multiplicand selected by bit $i$ |
|---|---|---|
| Bit $i$ | Bit $i-1$ | |
| 0 | 0 | $0 \times M$ |
| 0 | 1 | $+1 \times M$ |
| 1 | 0 | $-1 \times M$ |
| 1 | 1 | $0 \times M$ |

2 Marks

ii) Booth recoding for 0100010100

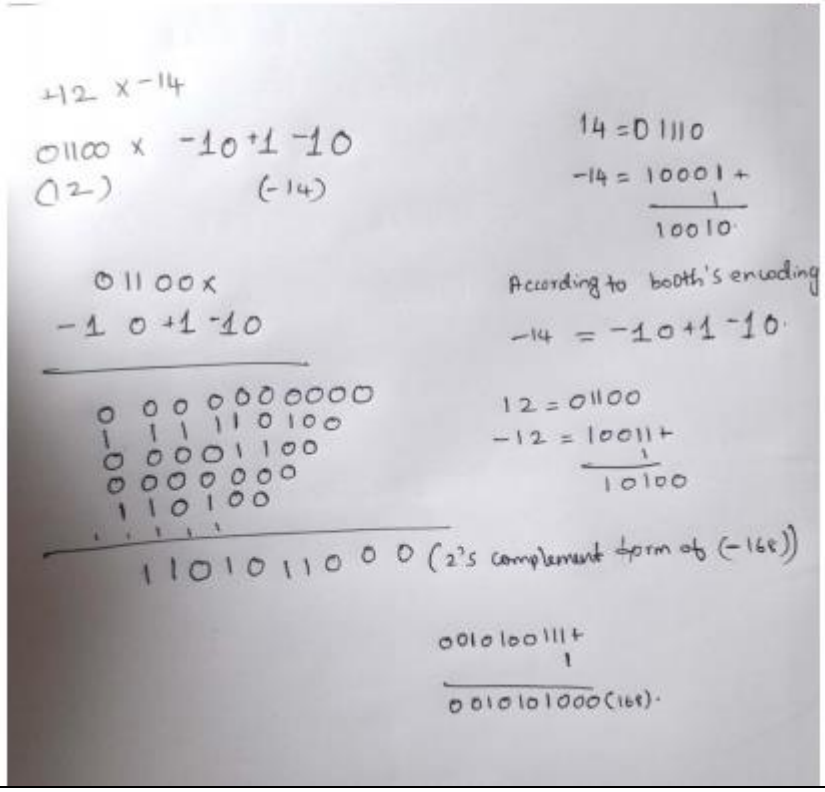
Ans: +1-1000-1+1-100

3 Marks

iii) Booth multiplication for + 12 and -14

Ans:

5 Marks




---