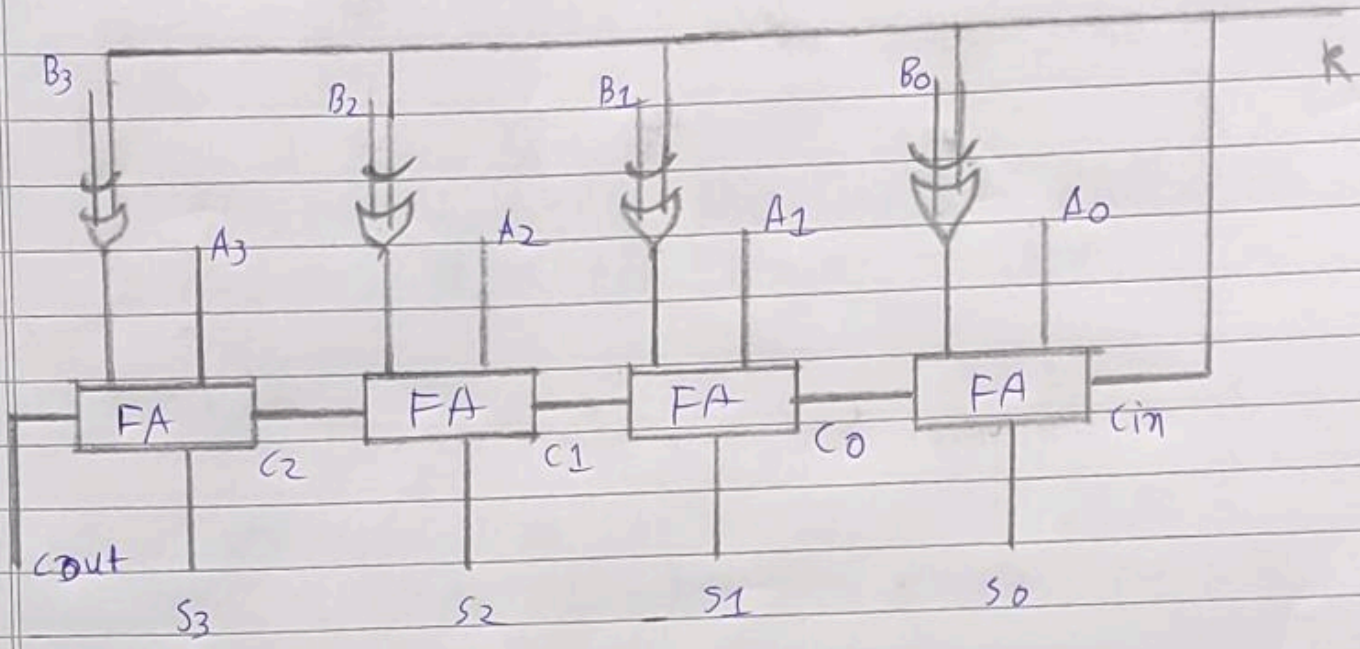
IU2341231551 4CSE-F COA

Assignment-1

1 Draw and Explain working of 4-bit binary adder-subtraction



1 Inputs

→ The circuit takes two 4-bit binary numbers (A and B) as input along with mode control input for addition or subtraction.

2 Addition

→ In the addition mode, A and B are added together using full-adder circuits

3 Subtraction

→ In the subtraction mode, B is complemented and 1 is added to perform two's complement subtraction.

→ The complemented B is added to A using the addition mode of the circuit.

4 Output

→ The result, either the sum or difference, is output as a 4-bit binary number.

5 Overflow

→ Overflow may occure if the result exceeds 4 bits, indicating an error in the operation.

2 List and explain Memory reference instructions in detail.

- Memory reference Instruction

1. AND 3. LOAD 5. BUN 7. ISZ
2. ADD 4. STA 6. BSA

1. AND (Logical AND):

- Syntax: AND operand
- operation: Performs a bitwise AND operation between the value in AC and the value stored in the memory location specified by the operand, and stores the result back into the AC

2 ADD (Addition):

- Syntax: ADD operand
- operation: Adds the value of the operand to AC

3 LDA (Load Accumulator):

- Syntax: LDA operand
- Operation: Loads the value from the memory location specified by the operand into the AC

4 STA (Store Accumulator)

- Syntax : STA operand
- Operation :- Stores the value from the AC into the memory location specified by the operand.

5 BUN (Branch Unconditionally)

- Syntax : BUN operand
- Operation :- Transfers control to the memory location specified by the operand unconditionally.

6 BSA (Branch and save Return Address)

- Syntax : BSA operand
- Operation : Jumps to the memory location specified by the operand and saves the return address (address of the next instruction) in a predetermined location.

7 ISZ (Increment and skip if Zero)

- Syntax : ISZ operand
- Operation : Increments the value stored in the memory location specified by the operand and skips the next instruction if the result is zero.

3 Define : microinstruction; Identify different types of 16 bits instruction formats for basic computer.

→ A Microinstruction is a low level instruction used by the control unit of a CPU to direct the internal operations of the processor. It defines tasks such as fetching instructions, decoding, executing operations, and managing data flow within the CPU.

→ Types of 16-bit Instruction

a. Single operand Format

→ 4 bits for opcode and 12 bits for the operand.

→ OPCODE (4-bits) | OPERAND (12 bits)

→ Used for instructions involving a single operand, such as load and store operations.

b immediate operand Format

→ 4 bits for opcode, 4 bits for operation specifier, and 8 bits for immediate operand.

→ OPCODE (4 bits) | OPERATION (4-bits) | IMMEDIATE operand (8 bits)

Suitable for instructions that require immediate data for operations, like immediate arithmetic or logical operations.

c Register - Register Format

4 bits for opcode, 4 bits for source register, and 4 bits for destination register.

OPCODE(4-bits) | SRC register (4-bits) | DEST REGISTER (4 bits)

Utilized for operations between two registers such as register-to-register data transfer or arithmetic/logical operations.

d Branch Format

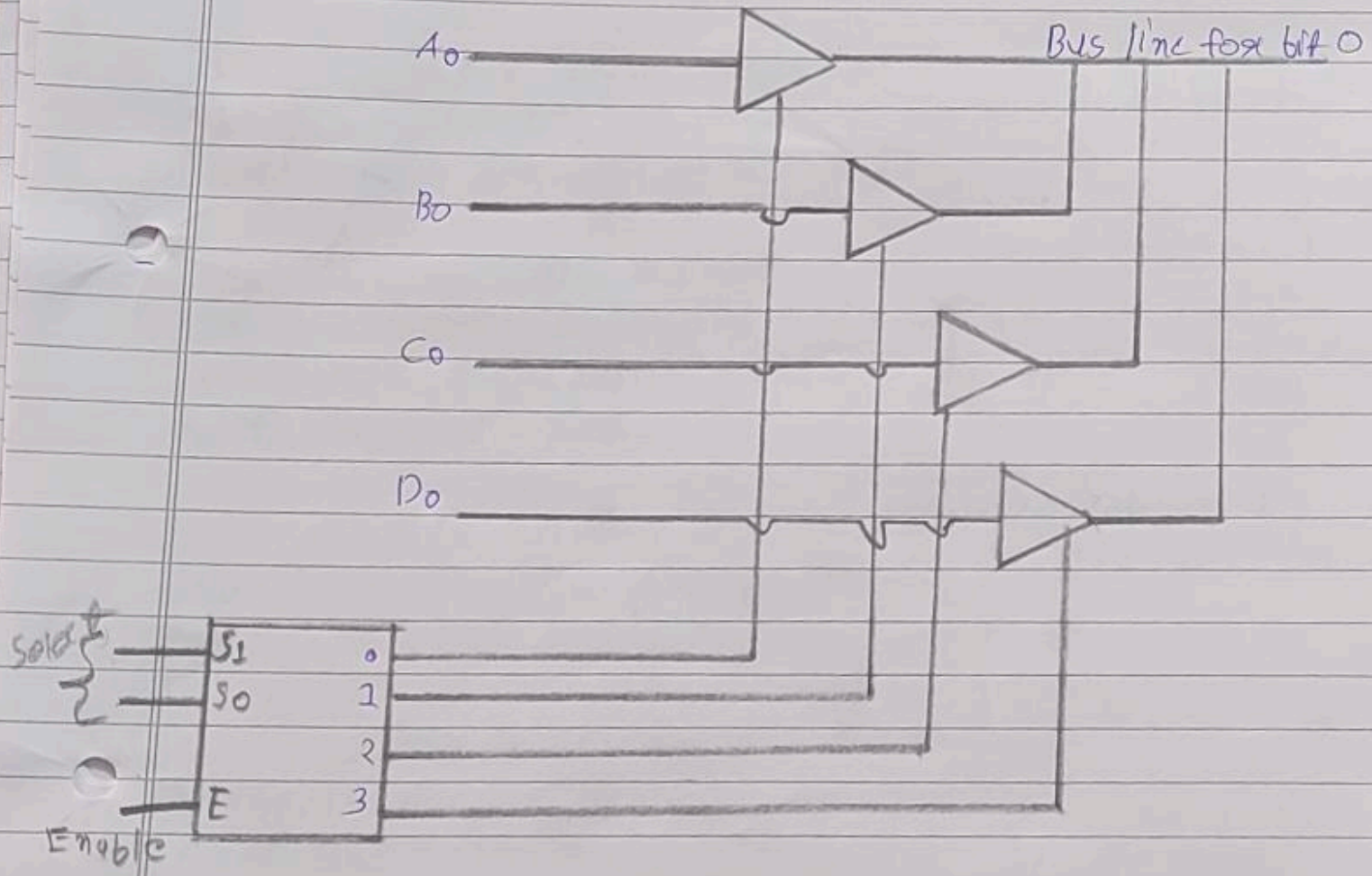
4 bits for opcode and 12 bits for the target address.

OPCODE(4-bits) | TARGET ADDRESS (12 bits)

~~Usage Allows cond~~

Enables branching or jumping to a specified memory address based on condition or control flow requirements

4 Explain hardware implementation of common bus system using three state buffers.



⇒ Three-state Buffers allows devices to connect to a common bus system and controls whether they actively drive the bus lines or disconnect from them.

The common bus system includes data, address, and control lines, facilitating communication between device.

Each device connects to the bus through three-state buffers, enabling them to transmit data onto the bus when needed.

Devices activate their output buffers to transmit data onto the bus. Other devices deactivate their output buffers to avoid interfering with data transfer

Bus arbitration mechanisms determine which device has priority access to the bus in multi-master systems.

Receiving devices activate their input buffers to capture data from the bus, while others keep their input buffers inactive to prevent data corruption.

5 Discuss application of Logical microops

1 Selative - set

→ sets specific bits to 1 while leaving others unchanged.

→ suppose we have register R with an initial value of 1010. we want to set the second and fourth bits to 1

.. original value 1010
0101 (bit positions to set)
1111

2 selative clear

→ clears specific bits to 0 while preserving others.

→ Consider the same register R with an initial value of 1010, we want to clear the first and third bits

original value 1010
1000 (bit position to clear)
0000

3 Selative Complement

⇒ Complements the value of chosen bits in a word.

⇒ Using the same reg R with an intial value of 1010, we want to complement the first and fourth bits.

⇒ Original value 1010
1011 (bit position to complement)
0100

4 Insert

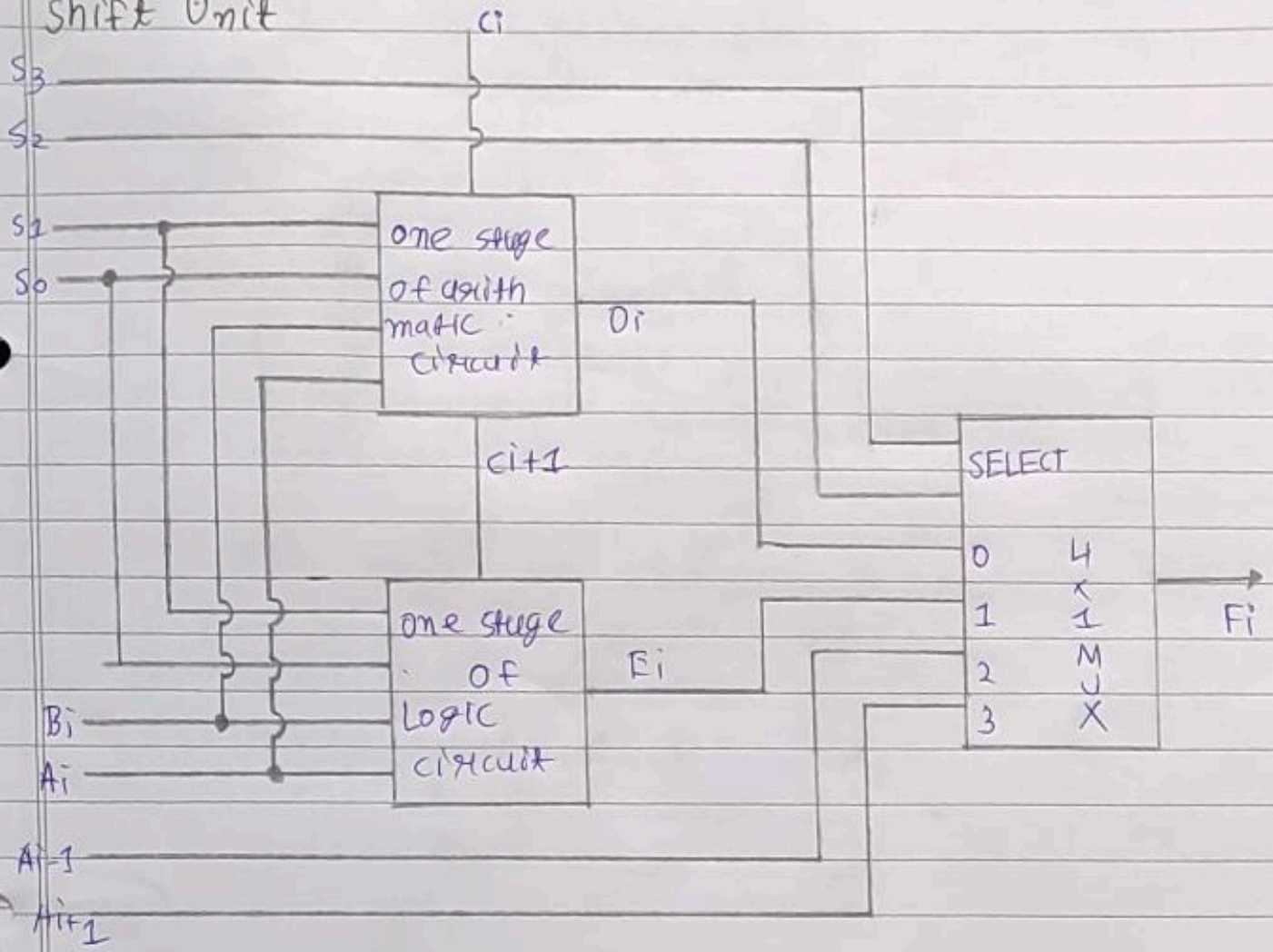
⇒ Insertion is used to insert specific data into selected bits of a word, while keeping other bits unchanged.

⇒ Consider a reg R with an initial value of 1010. We want insert the value 11 into the second and third bits.

⇒ Original value: 1010
Value to insert: 11

Mask: 0110
Result: 1110

6 Draw and Explain one stage of Arithmetic Logic shift Unit



→ The ALU receives two operands, Operand A and operand B. These operands, typically stored in registers, represent the data on which arithmetic Logic, or shift operations will be performed.

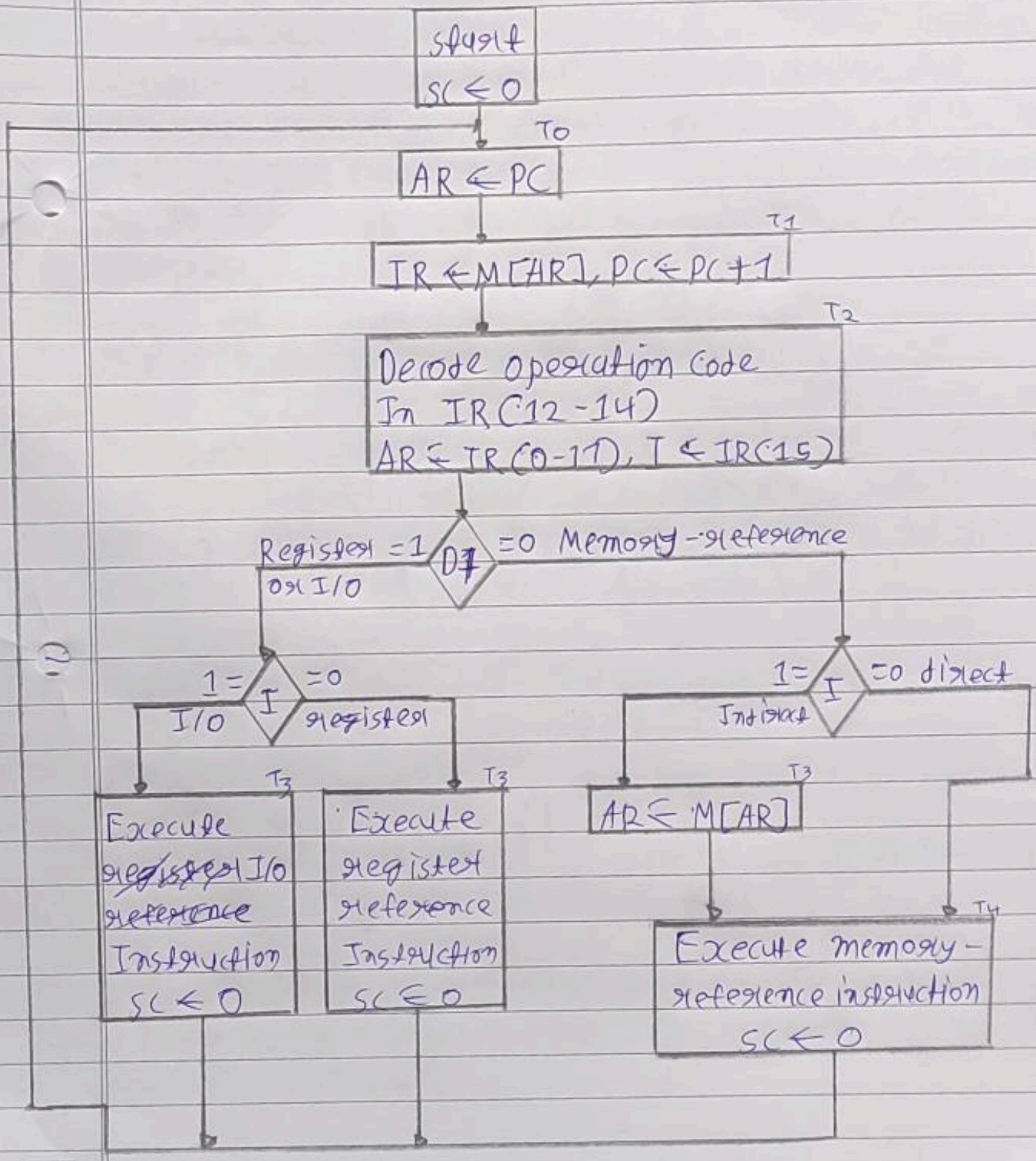
→ Within this stage, the ALU executes various arithmetic and logic operations on the input operands. These operations include addition, subtraction, AND, OR, XOR, and comparison operations like equality and inequality checks.

→ Additionally, the ALU handles shift operations which involves shifting the bits of an operand left or right. These shifts can be arithmetic or logical

→ The control logic dictates which operation the ALU performs based on the instruction being executed. It generates control signals to select the appropriate operation and manages other components of the ALU.

→ The result of the selected operation is the output of this stage. This result may be stored in a register for later use or directly forward to subsequent stages of the CPU pipline.

7 Which are the different phase of Instruction cycle? Describe Register transfer for fetch phase with its diagram.



- Fetch phase

→ During Fetch phase, the CPU retrieves the next instruction from Memory. This phase involves transferring the instruction's address from the program counter to memory Address Register (MAR) fetching the instruction from memory, and transferring it to the instruction Register (IR) for further processing

- Register Transfer for Fetch Phase

1. PC to MAR Transfer

→ The current value stored in the PC, which holds the address of the next instruction to be executed, is transferred to the AR.

→ Register Transfer: AR ← PC

2 Memory Read Operation

→ The CPU initiates a memory read operation using the address stored in the AR

→ Control signals are sent to the memory unit to retrieve the instruction stored at the address specified by the AR

3 Memory Data to IR Transfer

→ Once the instruction is fetched from memory, it is transferred to the IR for decoding and execution.

→ The data retrieved from memory is transferred from the memory data bus to the instruction register.

→ IR ← m[AR]

8 State differences between hardwired control unit and microprogrammed control unit.

Hard Wired Control Unit	Microprogrammed control unit
Implemented using combinational logic circuits	Implemented using control memory and a sequencer
Limited flexibility; changes require hardware modifications.	Highly flexible; changes can be made by modifying microprograms
Less complex design	More complex design due to the need for control memory
Difficult to accommodate changes in the instruction set.	Easily accommodates changes in the instruction set.
~~Generally~~ Faster operation as control signals are generated directly	Slightly slower operation due to additional memory accessess.

9 Explain register stack and Memory stack.

• Register stack

→ A register stack is a stack data structure implemented using a set of registers in computer's CPU.

→ The register stack typically consists of a fixed number of registers arranged in a stack-like fashion

→ The top of the stack is represented by the register that is currently accessible for read and write operations

→ Push Operation

When a value needs to be added to the stack it is stored in the top register, and the stack pointer is decremented to the point to the next available register.

→ Pop Operation

→ When a value is removed from the stack, the value in the top register is read, and the stack pointer is incremented to the next register.

Memory stack

→ A memory stack, often referred to simply as a stack, is a data structure used in computer science for managing data storage and retrieval.

→ The stack memory is contiguous block of memory divided into fixed-size slots, each capable of storing a data element.

→ The stack operates on LIFO principle, meaning that the last item pushed onto the stack is the first one to be popped off.

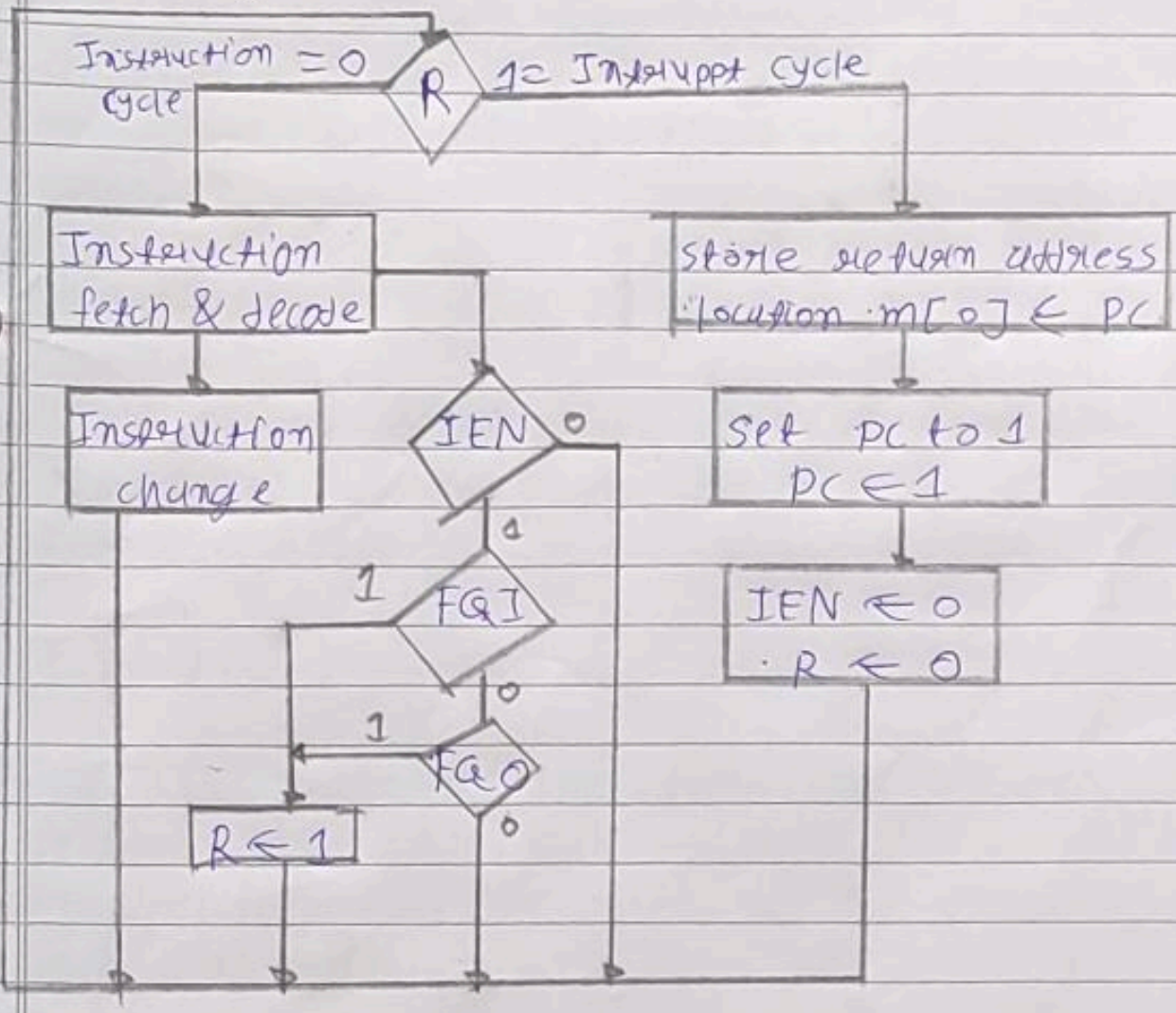
→ Push operation

When a value needs to be added to the stack it is placed at the top of stack, and the stack pointer is incremented to point to the next available memory slot.

→ POP operation

When a value is removed from the stack, the value at the top of stack is read, and the stack pointer is decremented to point to the read memory slot.

10 What is interrupt? Describe interrupt cycle with neat Diagram



⇒ An interrupt is a singnal sent from device or software to the the CPU to request its attension

⇒ Interrupts are used in computer systems to handle events or conditions that require immediate processing, such as I/O operations, hardware errors or timer events.

1 Fetch

→ CPU retrieves instruction from memory at the address specified by the program counter, ensuring the next instruction is available for processing.

2 Decode

→ CPU interprets the fetched instruction, identifying its opcode and any associated operands, preparing for execution

3 Execute

→ CPU performs the operation specified by the decoded instruction, such as logic, or, control flow, manipulating data or altering the program's execution path.

4 Fetch operands

→ In some CPU architectures, operands required for the instruction are fetched from memory or registers before the execution, phase begins, optimizing processing efficiency.

5 Store results

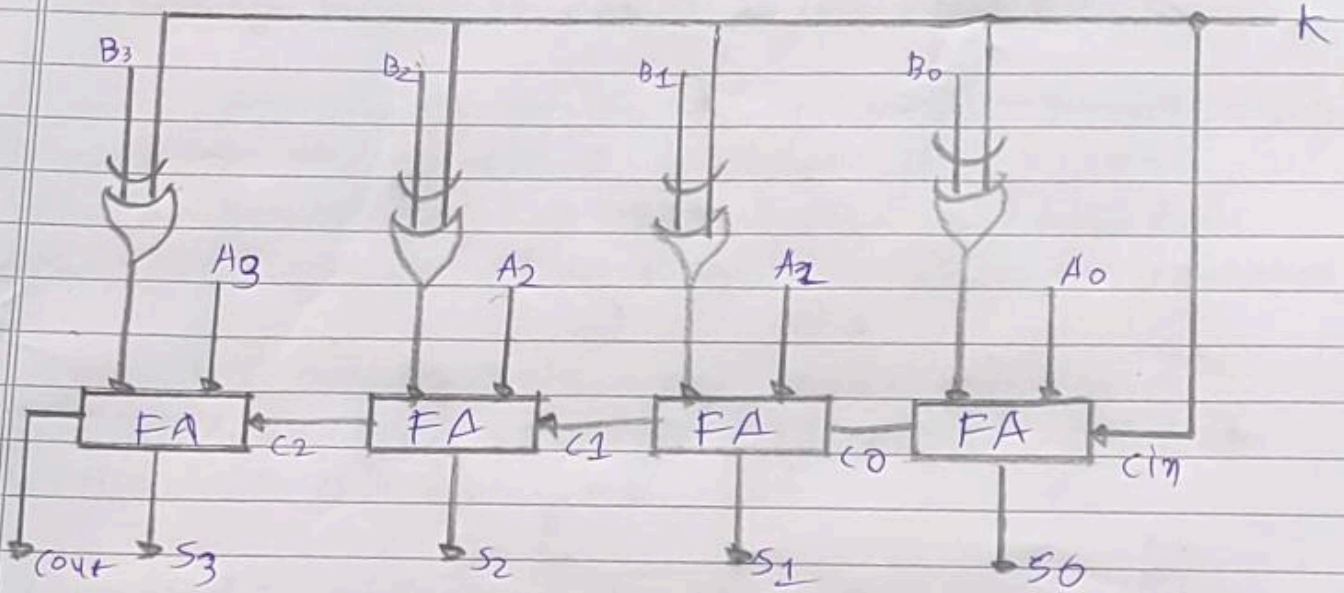
→ After executing the instruction, some CPUs store the results, such as computation outcomes or updated data, back into memory or registers, completing the instruction cycle.

6 Interrupt Handling

→ The CPU is capable of responding to interrupts, signals from external devices or software requesting immediate attention.

→ When interrupt occurs, the CPU suspends its current operation, executing an interrupt handler to service the interrupt and address the external event.

11 Construct a 4-bit adder-subtractor circuit.



12 What addressing mode means ? Explain any three addressing modes in detail with example.

→ Addressing modes in computer architecture refer to the methods used to specify to the operands of an instruction, determining how the CPU accesses data from memory or registers.

→ Different addressing modes provide flexibility in programming and allow efficient utilization of resources.

1 Immediate Addressing Mode

→ In immediate addressing mode, the operand is directly specified within the instruction itself

→ The operand value is constant and embedded directly into the instruction.

→ This mode is useful for operations that involves constants or immediate data.

- For Ex,

```
MOV AX, #5
ADD BX, #10
```

2 Register Addressing Mode

→ In register addressing mode, the operand is specified by referencing a register directly.

→ The instruction operates on data stored in registers, avoiding memory access.

→ This mode is efficient for operations that involve data manipulation within registers.

• For Ex,

```
ADD AX, BX
SUB CX, DX
```

3 Direct addressing Mode

→ In Indirect addressing mode, the operand's memory address is directly specified within the instruction

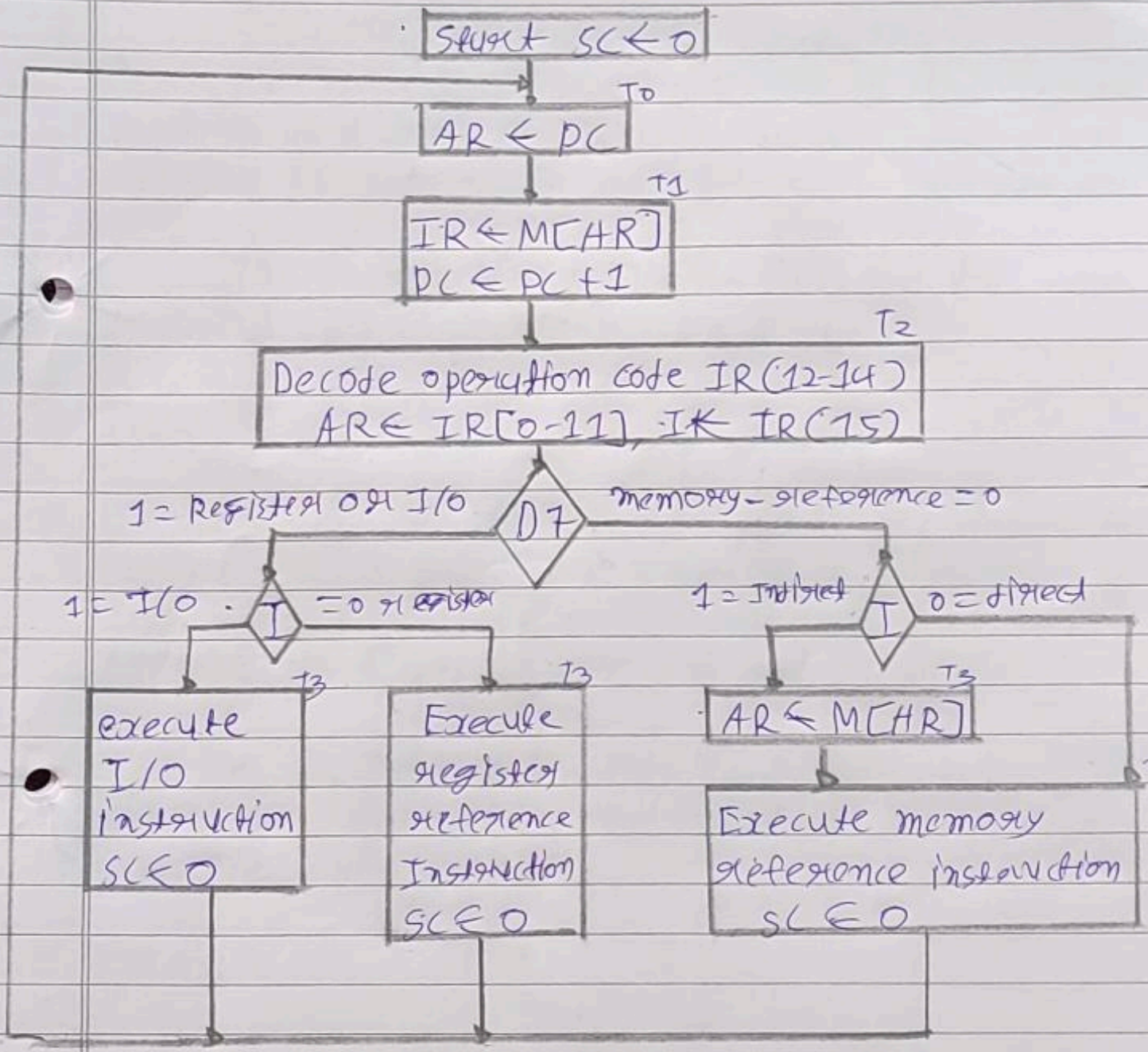
→ The instruction accesses the data stored at the specified memory address.

→ This mode is suitable for accessing variable or data stored in memory.

* For Ex,

```
MOV AX, [5000]
ADD BX, [6000]
```

13 Draw the flowchart for instruction cycle and Explain



→ The instruction cycle, aka the fetch-execute cycle, is the fundamental process by which a CPU executes instructions in computer program.

1 Fetch

→ The CPU retrieves the next instruction from memory. This involves accessing the memory location specified by the PC, which holds the address of the next instruction to be executed.

→ The instruction is loaded into the IR for decoding and execution.

2 Decode

→ The CPU ~~interrupt~~ interprets the instruction fetched during the fetch phase. This involves identifying the opcode of instruction and any operands that are required for the operation.

→ The decoded instruction provides information about the operation that needs to be performed and the data involved.

3 Execute

→ The CPU performs the operation specified by the decoded instruction. This may involve various actions such as arithmetic or logic operations, data manipulation, control flow alterations, or interaction with external devices.

4 Increment PC

→ After executing the current instruction, the PC is updated to point to the address of the next instruction in sequence. This prepares the CPU to fetch the next instruction in the subsequent iteration of the instruction cycle

14 List various types of addressing modes and explain each in detail.

1 Immediate Addressing Mode

→ Operands value is directly specified within the instruction itself

→ For Ex. MOV AX, #5

→ Suitable for operations involving constants or immediate data

2 Register Addressing Mode

→ Operands is specified by referencing a register directly.

→ For Ex, ADD AX, BX

→ provides efficient data manipulation within registers.

3 Direct Addressing Mode

→ Operand's memory address is directly specified within the instruction.

→ For Ex, MOV AX, [5000]

4 Indirect Addressing Mode

→ Operand is accessed by retrieving the address from memory first, then accessing the data stored at that address

→ For Ex, MOV AX, [BX]

→ Often used for implementing pointers or accessing data structures.

5 Indexed Addressing Mode

→ Operands memory address is calculated by adding an offset to base address stored in a register.

→ For Ex, MOV AX, [BX+10]

→ Commonly used for accessing elements of arrays or data structure

6 Relative Addressing MODE

→ Operand's memory address is calculated relative to the current instruction's address

→ For Ex. JMP Label

→ Frequently utilized for branch instruction and jumps.

15 Compare and construct RISC and CISC

RISC	CISC
Reduced and Simplified instruction set	Complex and extensive instruction set
Lower hardware complexity	Higher hardware complexity
~~Pipe~~ often feture deep pipeline for concurrent execution.	Typically has a shorter pipeline due to complex instructions.
Typically has larger number of general-purpose registers.	often has fewer general purpose registed.
Relies more on memory accesses for complex operations.	May perform more operations directly on memory.
Executes instruction quickly and efficiently	May execute complex instructions more slowly