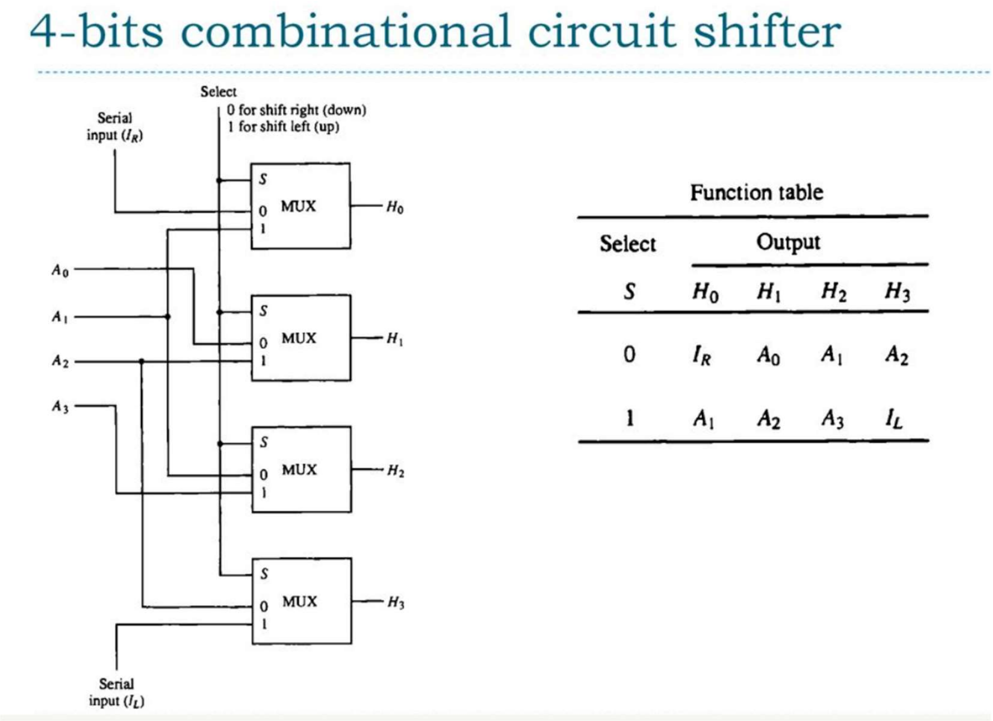
COA-W-2022

Q-1

A. Draw the block diagram of 4-bit combinational circuit shifter.

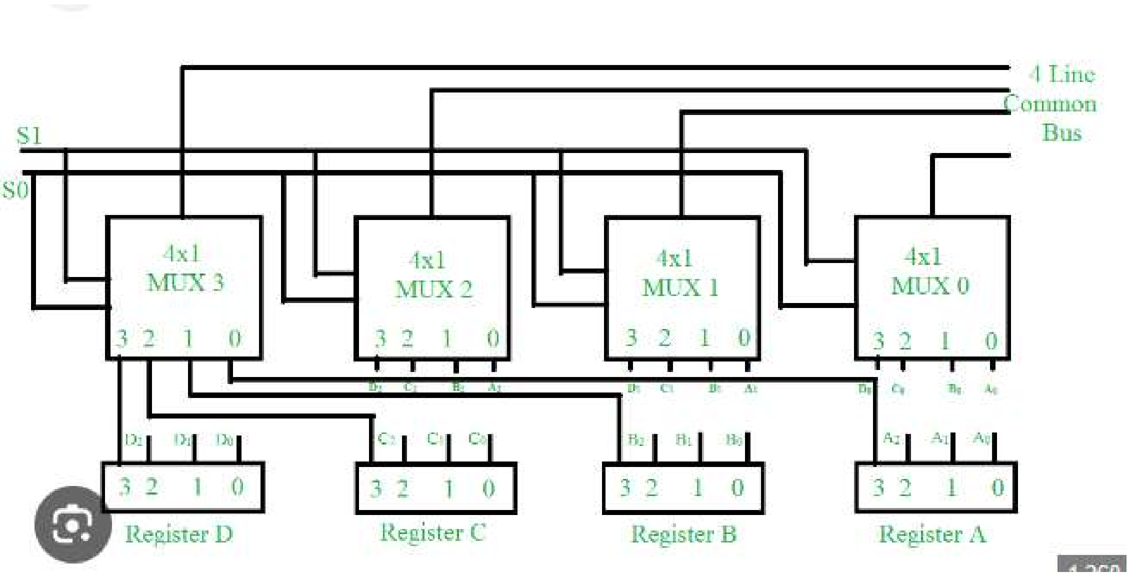


4-bits combinational circuit shifter

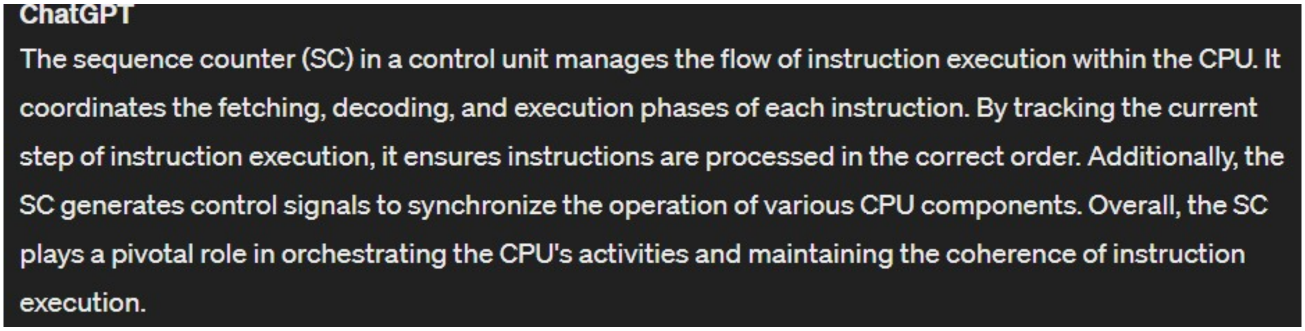
Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

B. Construct diagram of common bus system of four 4-bits registers with diagram.

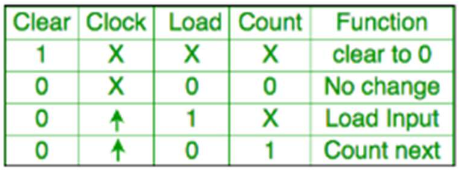


C. What is the role of sequence counter(SC) in control unit? Interpret its concept with the help of its three inputs using diagram.



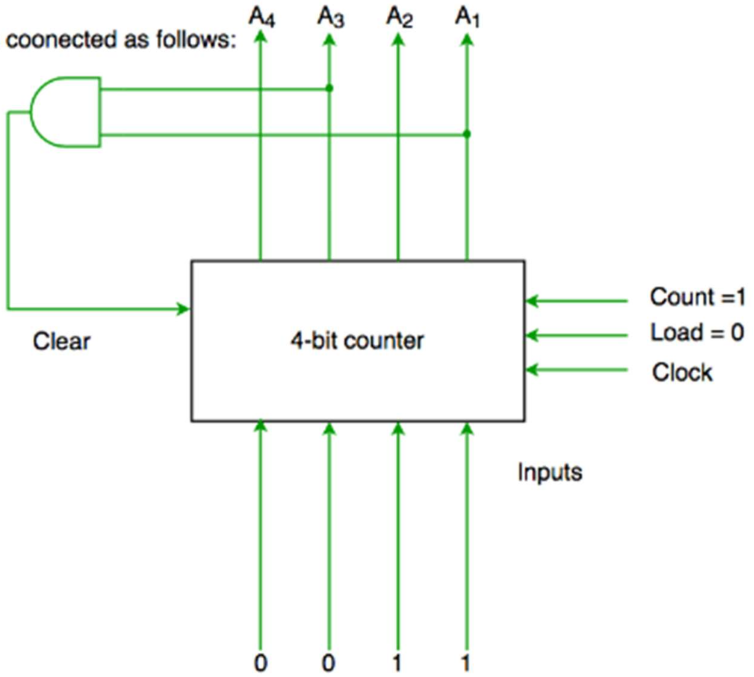
ChatGPT

The sequence counter (SC) in a control unit manages the flow of instruction execution within the CPU. It coordinates the fetching, decoding, and execution phases of each instruction. By tracking the current step of instruction execution, it ensures instructions are processed in the correct order. Additionally, the SC generates control signals to synchronize the operation of various CPU components. Overall, the SC plays a pivotal role in orchestrating the CPU's activities and maintaining the coherence of instruction execution.



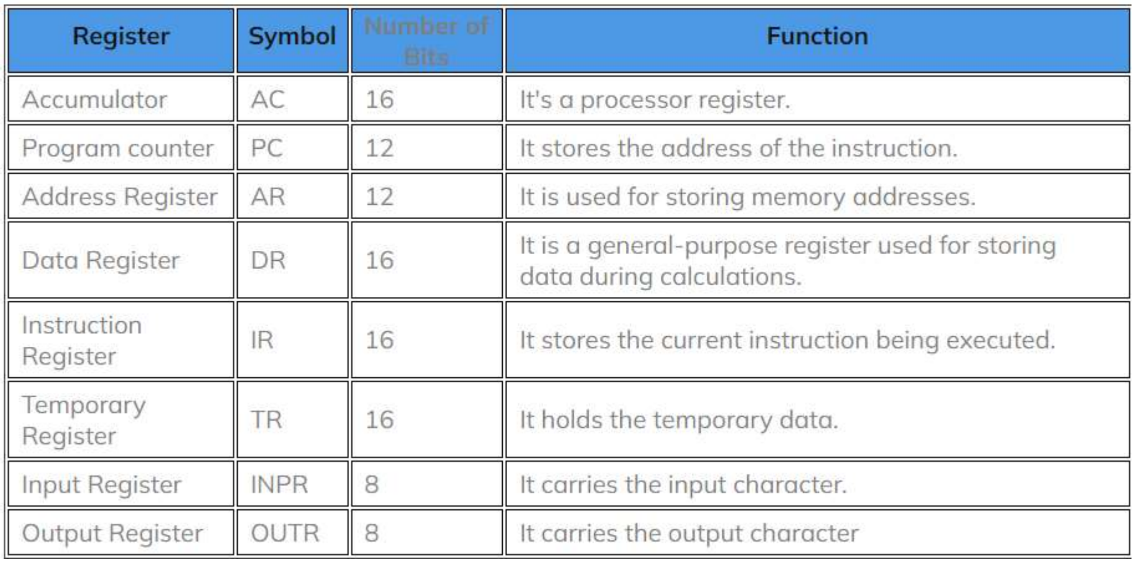
Clear	Clock	Load	Count	Function
1	X	X	X	clear to 0
0	X	0	0	No change
0	↑	1	X	Load Input
0	↑	0	1	Count next

The counter is coonected as follows:



Q-2

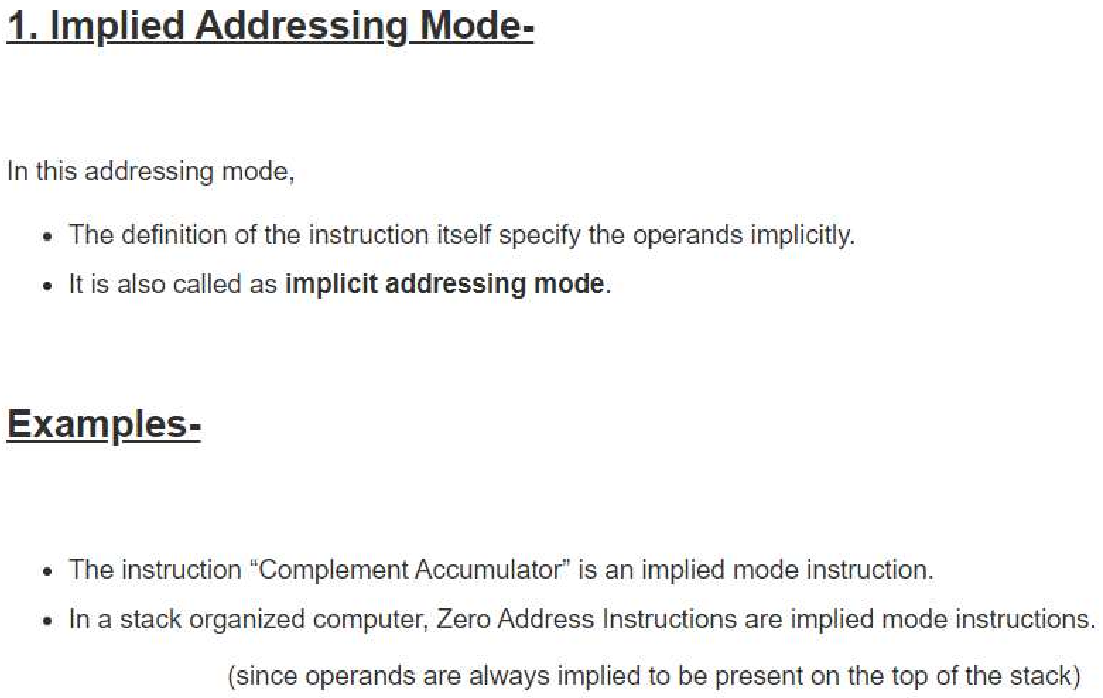
A. List out names of eight main registers of basic computer with their symbolic name and purpose.



Register	Symbol	Number of Bits	Function
Accumulator	AC	16	It's a processor register.
Program counter	PC	12	It stores the address of the instruction.
Address Register	AR	12	It is used for storing memory addresses.
Data Register	DR	16	It is a general-purpose register used for storing data during calculations.
Instruction Register	IR	16	It stores the current instruction being executed.
Temporary Register	TR	16	It holds the temporary data.
Input Register	INPR	8	It carries the input character.
Output Register	OUTR	8	It carries the output character

B. Summarize following addressing modes with example.

1) Implied mode



1. Implied Addressing Mode-

In this addressing mode,

- The definition of the instruction itself specify the operands implicitly.
- It is also called as **implicit addressing mode**.

Examples-

- The instruction “Complement Accumulator” is an implied mode instruction.
- In a stack organized computer, Zero Address Instructions are implied mode instructions.

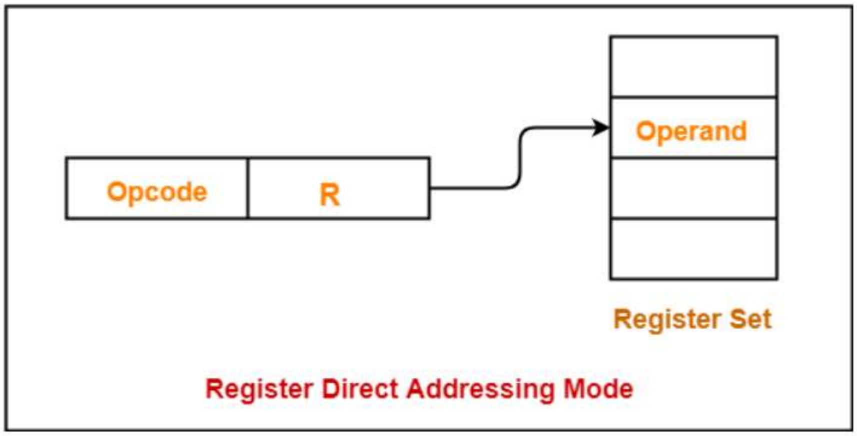
(since operands are always implied to be present on the top of the stack)

2) Register mode

6. Register Direct Addressing Mode-

In this addressing mode,

- The operand is contained in a register set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No reference to memory is required to fetch the operand.



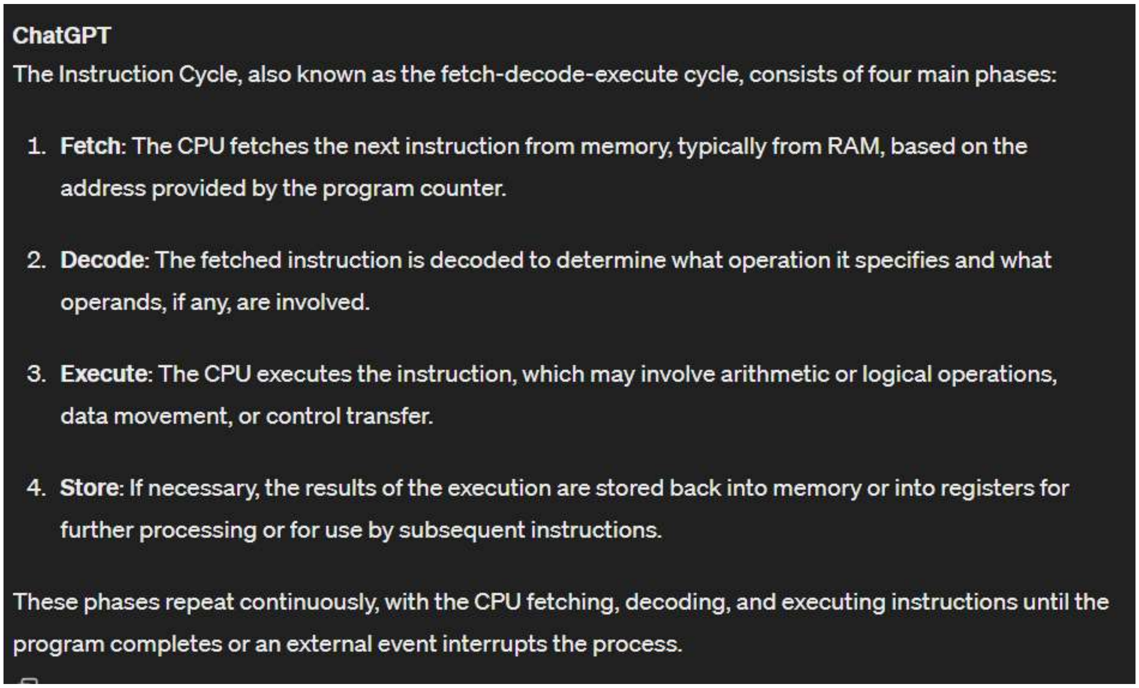
Register Direct Addressing Mode

Example-

- ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

C. Which are the different phases of Instruction Cycle? Describe Register transfer for fetch phase with its diagram.



ChatGPT

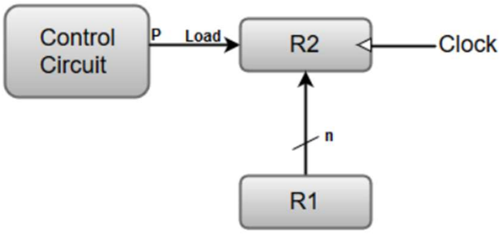
The Instruction Cycle, also known as the fetch-decode-execute cycle, consists of four main phases:

1. **Fetch**: The CPU fetches the next instruction from memory, typically from RAM, based on the address provided by the program counter.
2. **Decode**: The fetched instruction is decoded to determine what operation it specifies and what operands, if any, are involved.
3. **Execute**: The CPU executes the instruction, which may involve arithmetic or logical operations, data movement, or control transfer.
4. **Store**: If necessary, the results of the execution are stored back into memory or into registers for further processing or for use by subsequent instructions.

These phases repeat continuously, with the CPU fetching, decoding, and executing instructions until the program completes or an external event interrupts the process.

The following image shows the block diagram that depicts the transfer of data from R1 to R2.

Transfer from R1 to R2 when P = 1:

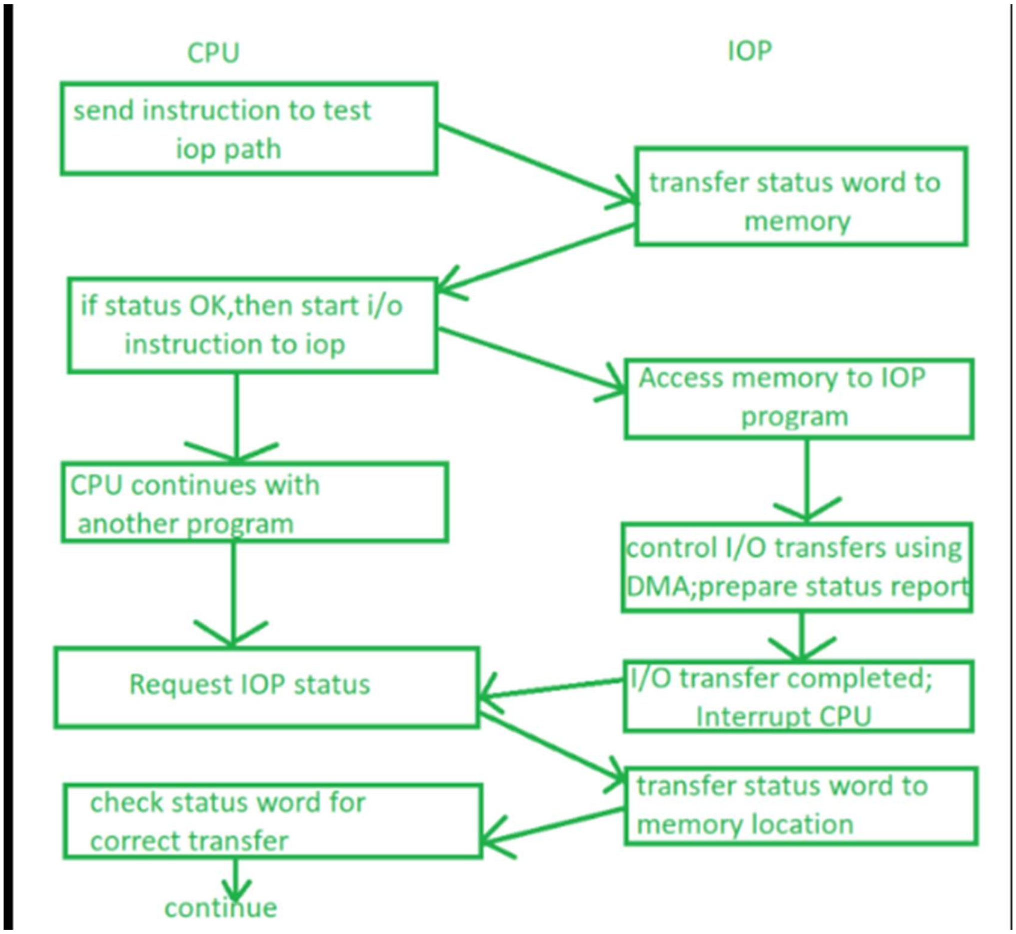


Here, the letter 'n' indicates the number of bits for the register. The 'n' outputs of the register R1 are connected to the 'n' inputs of register R2.

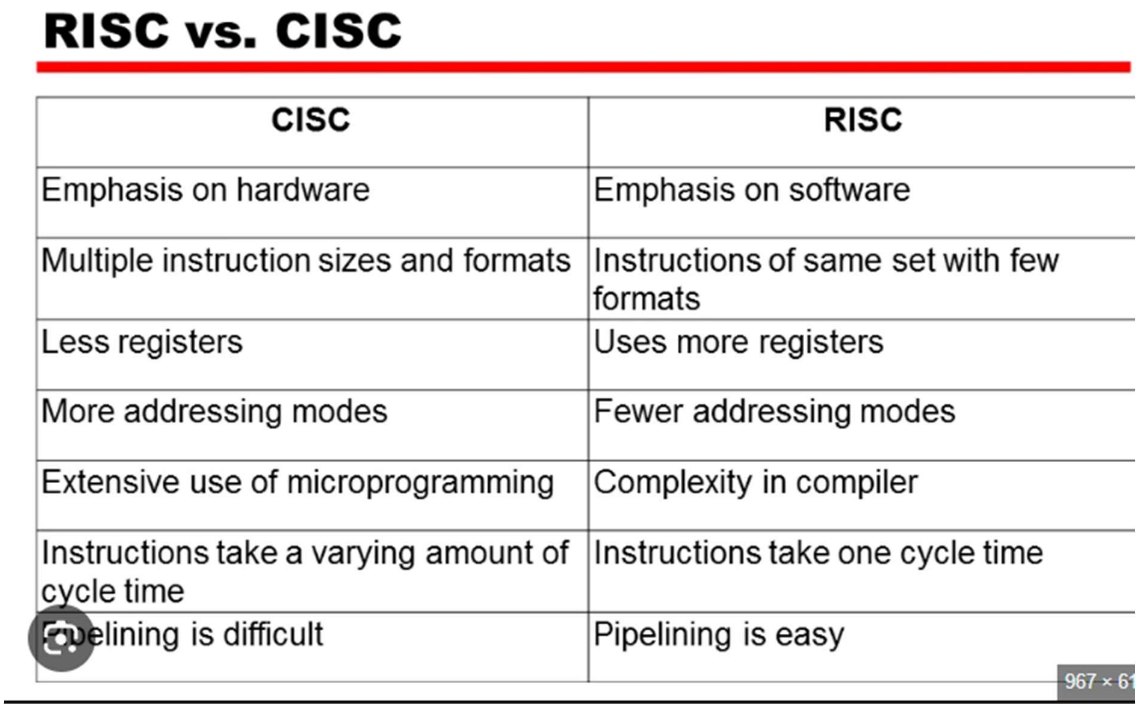
A load input is activated by the control variable 'P' which is transferred to the register R2.

Q-3

A. Prepare flowchart of CPU-IOP communication.


CPU
IOP
send instruction to test iop path
transfer status word to memory
if status OK,then start i/o instruction to iop
Access memory to IOP program
CPU continues with another program
control I/O transfers using DMA;prepare status report
Request IOP status
I/O transfer completed; Interrupt CPU
transfer status word to memory location
check status word for correct transfer
continue

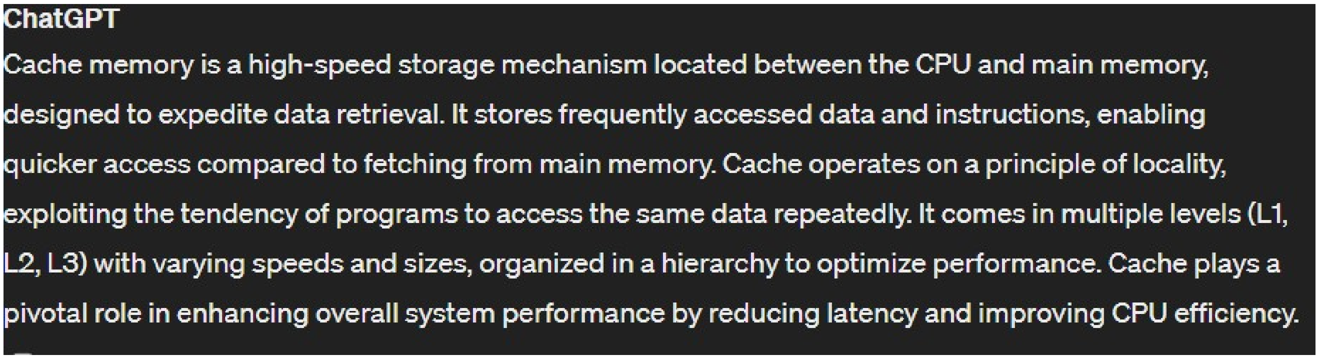
B. Differentiate RISC and CISC architecture.



RISC vs. CISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

C. What is cache memory? Interpret direct addressing mapping with diagram.

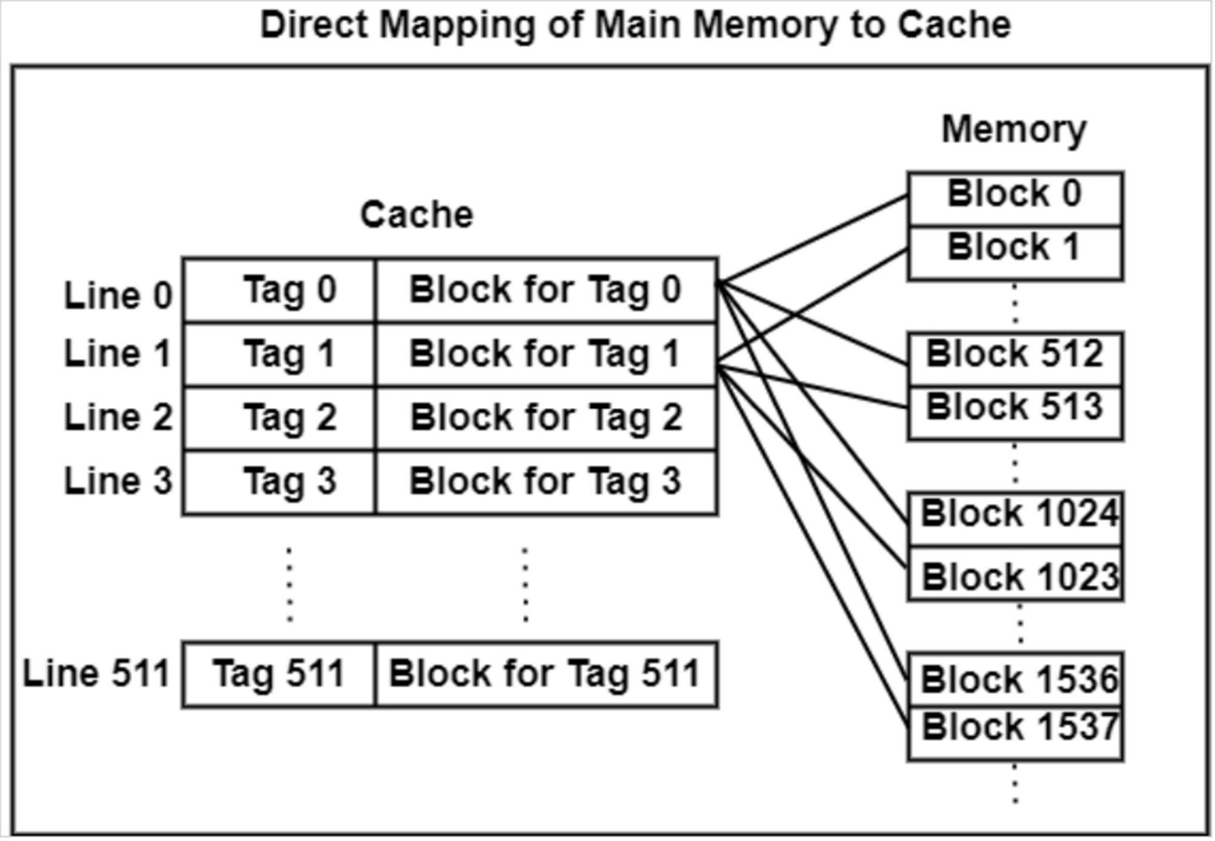


ChatGPT

Cache memory is a high-speed storage mechanism located between the CPU and main memory, designed to expedite data retrieval. It stores frequently accessed data and instructions, enabling quicker access compared to fetching from main memory. Cache operates on a principle of locality, exploiting the tendency of programs to access the same data repeatedly. It comes in multiple levels (L1, L2, L3) with varying speeds and sizes, organized in a hierarchy to optimize performance. Cache plays a pivotal role in enhancing overall system performance by reducing latency and improving CPU efficiency.

Direct mapping is a procedure used to assign each memory block in the **main memory** to a particular line in the cache. If a line is already filled with a memory block and a new block needs to be loaded, then the old block is discarded from the **cache**.

The figure shows how multiple blocks from the example are mapped to each line in the cache.



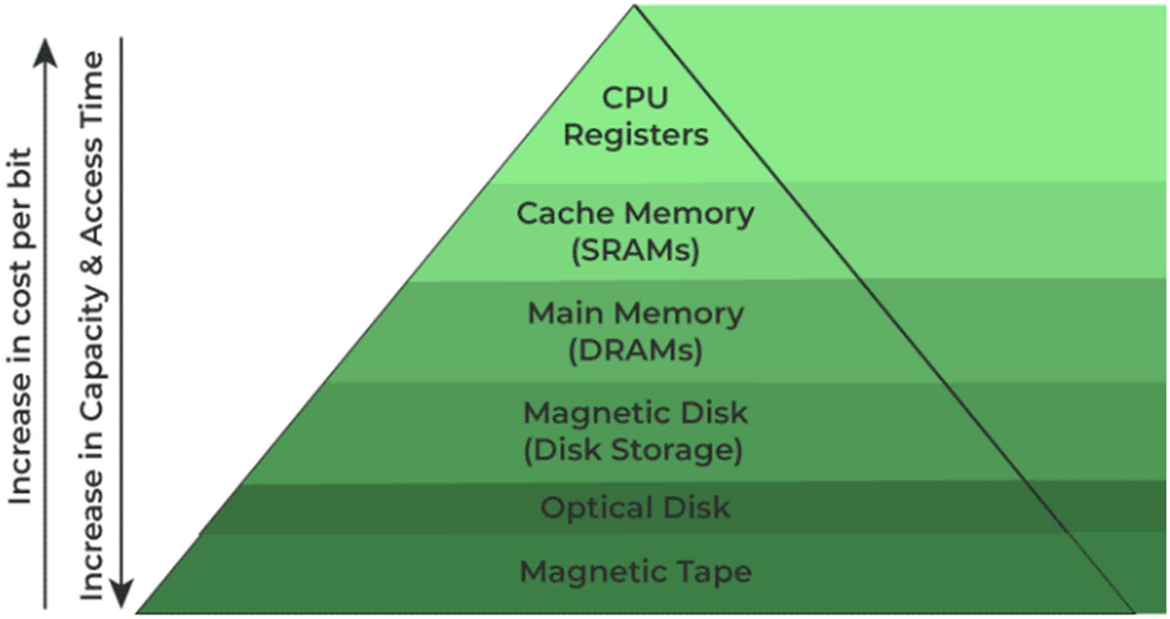
Direct Mapping of Main Memory to Cache

Just like locating a word within a block, bits are taken from the main memory address to uniquely describe the line in the cache where a block can be stored.

Example − Consider a cache with = 512 lines, then a line would need 9 bits to be uniquely identified.

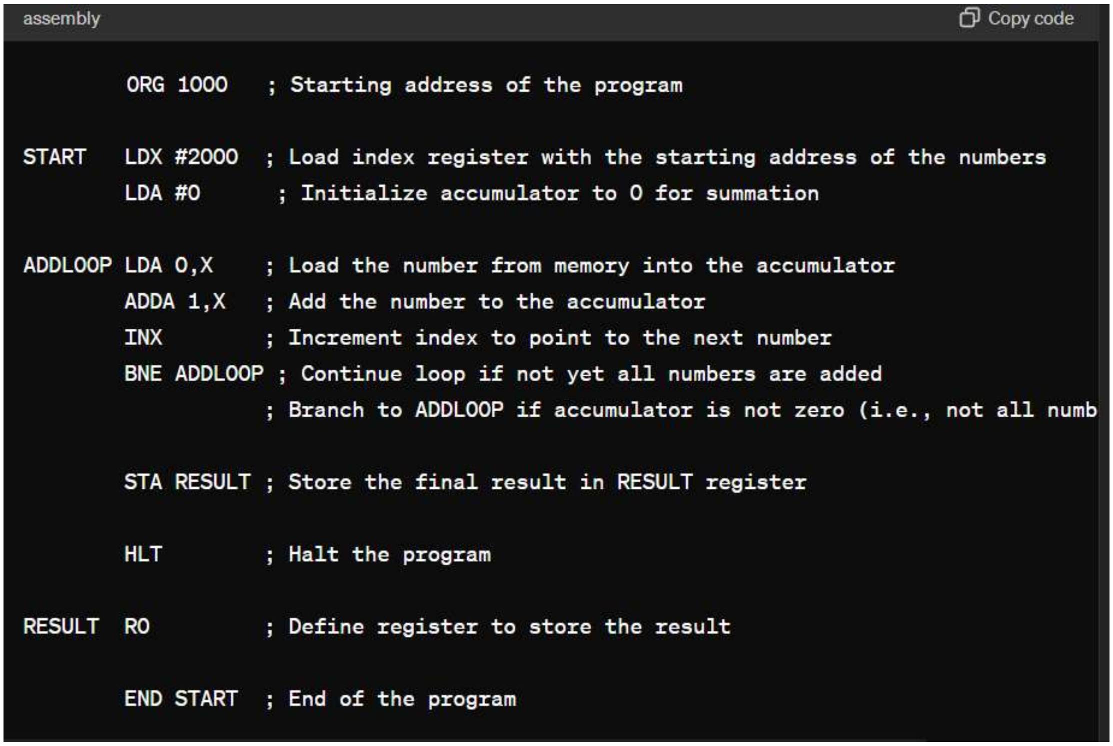
Q-4

A. Draw and criticize memory hierarchy in a computer system.



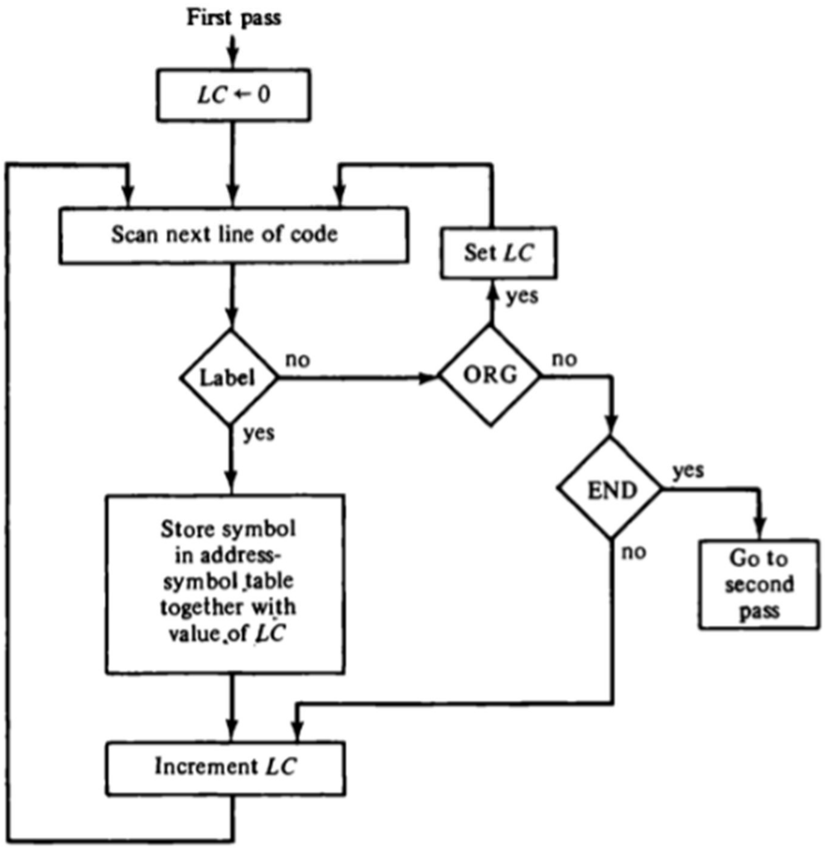
Memory Hierarchy Design

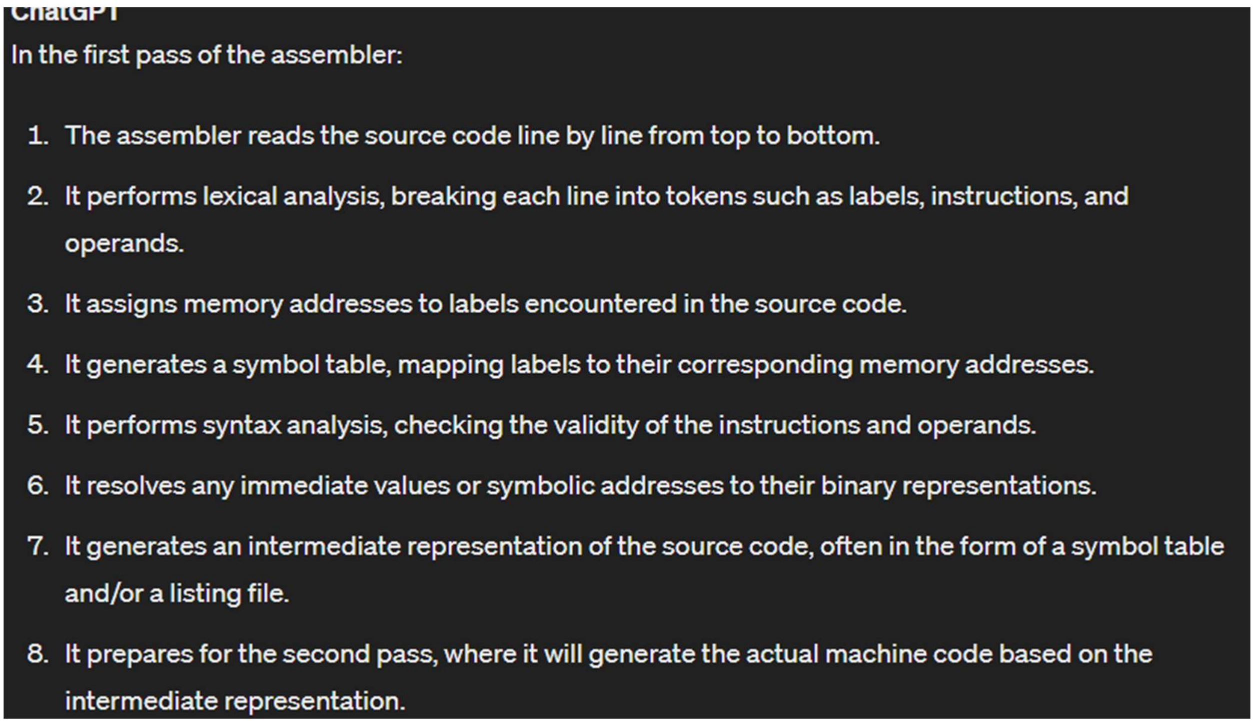
B. Write an Assembly level program for addition of 50 numbers.



```
        ORG 1000   ; Starting address of the program

START   LDX #2000  ; Load index register with the starting address of the numbers
        LDA #0      ; Initialize accumulator to 0 for summation

ADDLOOP LDA 0,X    ; Load the number from memory into the accumulator
        ADDA 1,X   ; Add the number to the accumulator
        INX        ; Increment index to point to the next number
        BNE ADDLOOP ; Continue loop if not yet all numbers are added
                   ; Branch to ADDLOOP if accumulator is not zero (i.e., not all numb

        STA RESULT ; Store the final result in RESULT register

        HLT        ; Halt the program

RESULT  R0         ; Define register to store the result

        END START  ; End of the program
```

C. Draw the flowchart of first pass of the assembler and explain working of the same.



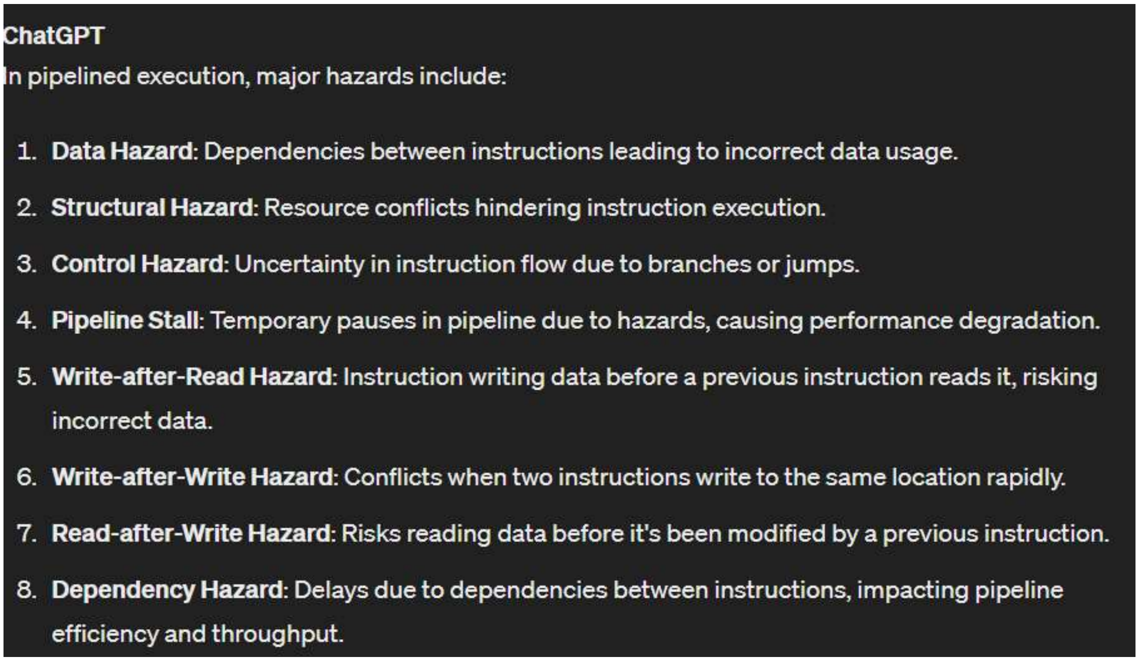

ChatGPT

In the first pass of the assembler:

1. The assembler reads the source code line by line from top to bottom.
2. It performs lexical analysis, breaking each line into tokens such as labels, instructions, and operands.
3. It assigns memory addresses to labels encountered in the source code.
4. It generates a symbol table, mapping labels to their corresponding memory addresses.
5. It performs syntax analysis, checking the validity of the instructions and operands.
6. It resolves any immediate values or symbolic addresses to their binary representations.
7. It generates an intermediate representation of the source code, often in the form of a symbol table and/or a listing file.
8. It prepares for the second pass, where it will generate the actual machine code based on the intermediate representation.

Q-5

A. Summarize major hazards in pipelined execution.



ChatGPT

In pipelined execution, major hazards include:

1. **Data Hazard**: Dependencies between instructions leading to incorrect data usage.
2. **Structural Hazard**: Resource conflicts hindering instruction execution.
3. **Control Hazard**: Uncertainty in instruction flow due to branches or jumps.
4. **Pipeline Stall**: Temporary pauses in pipeline due to hazards, causing performance degradation.
5. **Write-after-Read Hazard**: Instruction writing data before a previous instruction reads it, risking incorrect data.
6. **Write-after-Write Hazard**: Conflicts when two instructions write to the same location rapidly.
7. **Read-after-Write Hazard**: Risks reading data before it's been modified by a previous instruction.
8. **Dependency Hazard**: Delays due to dependencies between instructions, impacting pipeline efficiency and throughput.

B. What is a data dependency conflict in instruction pipeline?

→ Recommend solutions for data dependency conflicts.

A data dependency conflict in an instruction pipeline occurs when the execution of one instruction depends on the result of a previous instruction, causing a hazard that can lead to incorrect execution or stalls in the pipeline. There are three types of data dependency conflicts:

1. **Read-after-Write (RAW)**: An instruction attempts to read data before a previous instruction writes to it.

2. **Write-after-Read (WAR)**: An instruction writes data before a previous instruction reads from it.

3. **Write-after-Write (WAW)**: Two instructions attempt to write to the same data location in rapid succession.

To mitigate data dependency conflicts in instruction pipelines, several techniques can be employed:

1. **Forwarding (or Data Bypassing)**: Allows data to be passed directly from the output of one pipeline stage to the input of another, bypassing intermediate storage.
2. **Hardware Interlocking (or Stalling)**: Introduces stall cycles into the pipeline, delaying the execution of dependent instructions until the required data is available.

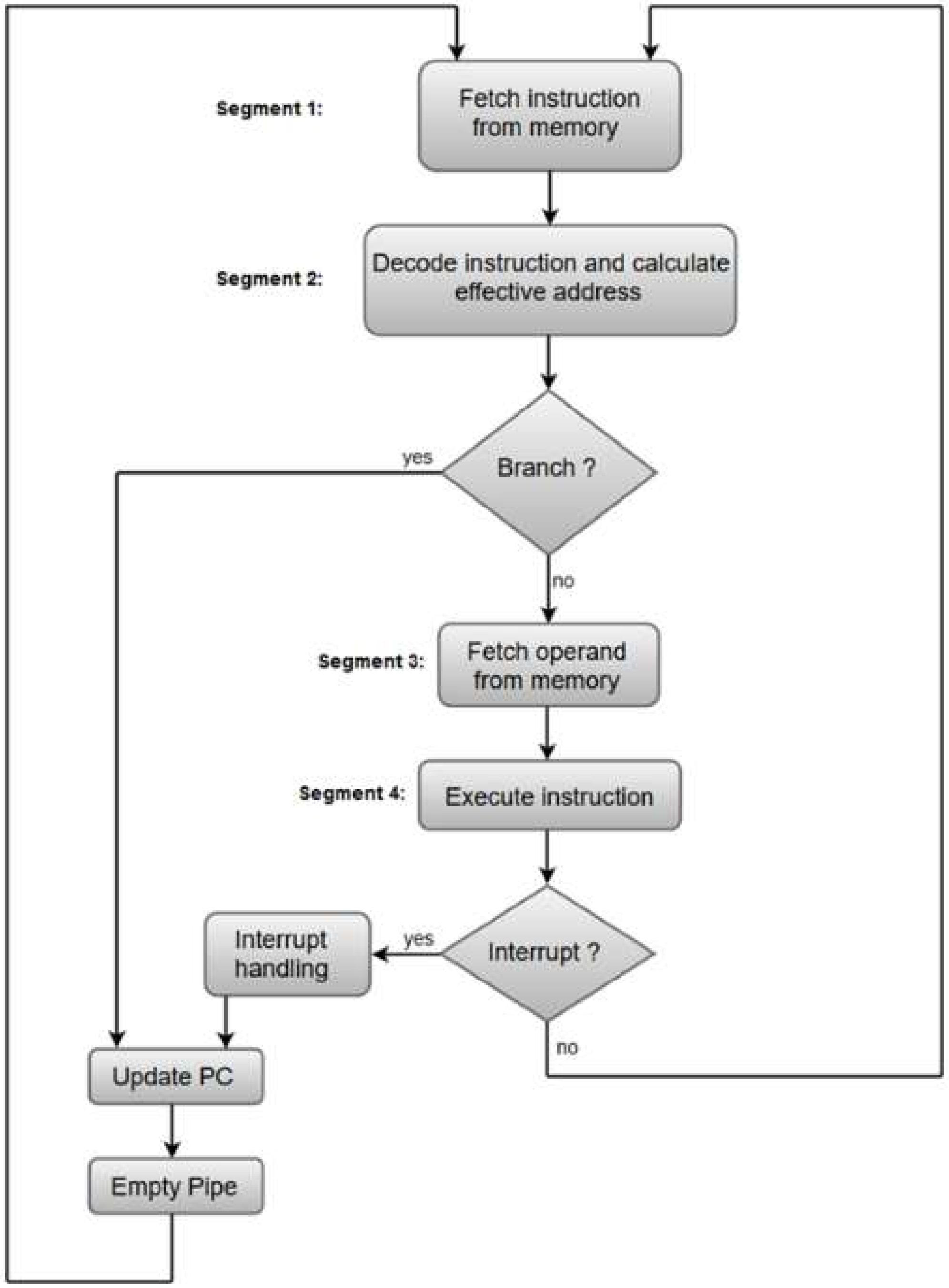
3. **Software Scheduling (or Compiler Optimization)**: Reorders instructions to reduce dependencies and maximize instruction-level parallelism.
4. **Speculation**: Speculative execution involves predicting the outcome of branch instructions and executing instructions based on that prediction, which can help avoid stalls caused by control hazards.
5. **Dependency Checkers (or Scoreboarding)**

By employing a combination of these techniques, processors can effectively manage data dependency conflicts in instruction pipelines, improving performance and efficiency.

C. Demonstrate four-segment instruction pipeline in detail.

→ A four-segment instruction pipeline combines two or more different segments and makes it as a single one. For instance, the decoding of the instruction can be combined with the calculation of the effective address into one segment.

The following block diagram shows a typical example of a four-segment instruction pip



Segment 1:

The instruction fetch segment can be implemented using first in, first out (FIFO) buffer.

Segment 2:

The instruction fetched from memory is decoded in the second segment, and eventually, the effective address is calculated in a separate arithmetic circuit.

Segment 3:

An operand from memory is fetched in the third segment.

Segment 4:

The instructions are finally executed in the last segment of the pipeline organization.