TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Course Title: Computer Architecture
Course No: CSC 208
Nature of the Course: Theory + Lab
Semester: III

Full Marks: 60
Pass Marks: 24
Time: 3

TU QUESTIONS-ANSWERS 2077

Section A

Long Questions

Attempt any TWO Questions: (2×10=20)

1. What do you mean by pipeline? Explain with space time diagram for a six segmented pipeline showing the time it takes to process eight tasks.

Ans: 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. We can consider the pipelining concept as a collection of several segments of data processing programs which will be processing the data and sending the results to the next segment until the end of the processing is reached.

Space time diagram for a six segmented pipeline

1. A typical instruction cycle can be split into many sub cycles like Fetch instruction, Decode instruction, Execute and Store. The instruction cycle and the corresponding sub cycles are performed for each instruction. These sub cycles for different instructions can thus be interleaved or in other words these sub cycles of many instructions can be carried out simultaneously, resulting in reduced overall execution time. This is called instruction pipelining.
2. The more are the stages in the pipeline, the more the throughput is of the CPU.
3. If the instruction processing is split into six phases, the pipelined CPU will have six different stages for the execution of the sub phases.
4. The six stages are as follows:
   - Fetch instruction (FI):
   - Decode instruction ((DI):
   - Calculate operand (CO):
   - Fetch operands (FO):
   - Execute Instruction (EI):
   - Write operand (WO):

Fetch instruction: Instructions are fetched from the memory into a temporary buffer before it gets executed.

Decode instruction: The instruction is decoded by the CPU so that the necessary op codes and operands can be determined.

Calculate operand: Based on the addressing scheme used, either operands are directly provided in the instruction or the effective address has to be calculated.

Fetch Operand: Once the address is calculated, the operands need to be fetched from the address that was calculated. This is done in this phase.

Execute Instruction: The instruction can now be executed.
Write operand: Once the instruction is executed, the result from the execution needs to be stored or written back in the memory.

5. The timing diagram of a six stage instruction pipeline is shown in Figure below:

Clock	1	2	3	4	5	6	7	8
I_1	FI	DI	CO	FO	EI	WO		
I_2		FI	DI	CO	FO	EI	WO	
I_3			FI	DI	CO	FO	EI	WO

6. Assuming that the sub cycles of the instruction cycle take exactly the same time to complete i.e. one clock cycle in this case.
7. In case the time required by each of the sub phase is not same appropriate delays need to be introduced. From this timing diagram it is clear that the total execution time of 3 instructions in this 6 stages pipeline is 8-time units. The first instruction gets completed after 6 time unit, and thereafter in each time unit it completes one instruction. Without pipeline, the total time required to complete 3 instructions would have been 18 (6 3) time units. Therefore, there is a speed up in pipeline processing and the speed up is related to the number of stages.

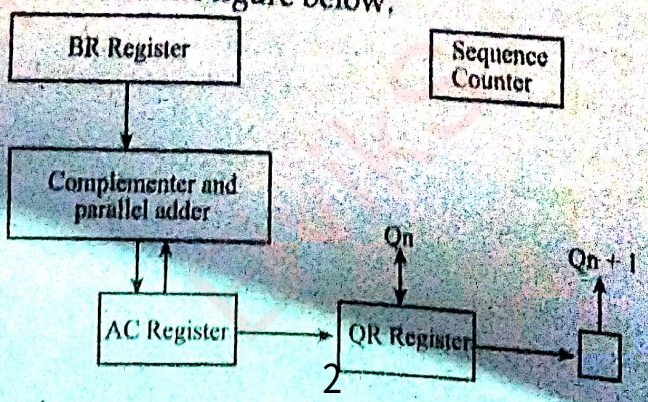
2. Explain Booth multiplication algorithm with hardware implementation diagram. Multiply (-5) x (-7) using Booth multiplication algorithm.

Ans: Booth algorithm gives a procedure for **multiplying binary integers** in signed 2's complement representation **in efficient way**, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{(k+1)}$ to 2^m. As in all multiplication schemes, booth algorithm requires examination **of the multiplier bits** and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Hardware Implementation of Booths Algorithm

The hardware implementation of the booth algorithm requires the register configuration shown in the figure below.



Example – A numerical example of booth's algorithm is shown below for n = 4. It shows the step by step multiplication of -5 and -7.

BR = -5 = 1011,

BR' = 0100, ← 1's Complement (change the values 0 to 1 and 1 to 0)

BR'+1 = 0101 ← 2's Complement (add 1 to the Binary value obtained after 1's compliment)

QR = -7 = 1001 ← 2's Complement of 0111 (7 = 0111 in Binary)

The explanation of first step is as follows: Qn+1

AC = 0000, QR = 1001, Qn+1 = 0, SC = 4

Qn Qn+1 = 10

So, we do AC + (BR)'+1, which gives AC = 0101

On right shifting AC and QR, we get

AC = 0010, QR = 1100 and Qn+1 = 1

OPERATION	AC	QR	Qn+1	SC
	0000	1001	0	4
AC + BR' + 1	0101	1001	0	
ASHR	0010	1100	1	3
AC + BR	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + BR' + 1	0010	0011	1	0

Product is calculated as follows:

Product = AC QR

Product = 0010 0011 = 35

3. **Define I/O interface. Comparison between programmed I/O, interrupt driven I/O and direct memory access (DMA).**

Ans: It provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices, resolves the differences between the computer and peripheral devices

- Peripherals - Electromechanical Devices, CPU or Memory - Electronic Device
- Data Transfer Rate
- Peripherals - Usually slower
- CPU or Memory - Usually faster than peripherals
- Some kinds of Synchronization mechanism may be needed
- Unit of Information
- Peripherals – Byte, Block, …
- CPU or Memory – Word
- Data representations may differ

Data transfer between the CPU and I/O devices can be done in variety of modes. These are three possible modes:

- Programmed I/O
- Interrupt initiated I/O
- Direct Memory Access (DMA)

1. **Programmed I/O**

In this mode the data transfer is initiated by the instructions written in a computer program. An input instruction is required to store the data from the device to the CPU and a store instruction is required to transfer the data from the CPU to the device. Data transfer through this mode requires

constant monitoring of the peripheral device by the CPU and also monitor the possibility of new transfer once the transfer has been initiated. Thus CPU stays in a loop until the I/O device indicates that it is ready for data transfer. Thus programmed I/O is a time consuming process that keeps the processor busy needlessly and leads to wastage of the CPU cycles. This can be overcome by the use of an interrupt facility. This forms the basis for the Interrupt Initiated I/O.

2. **Interrupt Initiated I/O**

This mode uses an interrupt facility and special commands to inform the interface to issue the interrupt command when data becomes available and interface is ready for the data transfer. In the meantime CPU keeps on executing other tasks and need not check for the flag. When the flag is set, the interface is informed and an interrupt is initiated. This interrupt causes the CPU to deviate from what it is doing to respond to the I/O transfer. The CPU responds to the signal by storing the return address from the program counter (PC) into the memory stack and then branches to service that processes the I/O request. After the transfer is complete, CPU returns to the previous task it was executing. The branch address of the service can be chosen in two ways known as vectored and non-vectored interrupt. In vectored interrupt, the source that interrupts, supplies the branch information to the CPU while in case of non-vectored interrupt the branch address is assigned to a fixed location in memory.

Difference between Programmed and Interrupt Initiated I/O

Programmed I/O	Interrupt Initiated I/O
Data transfer is initiated by the means of instructions stored in the computer program. Whenever there is a request for I/O transfer the instructions are executed from the program.	The I/O transfer is initiated by the interrupt command issued to the CPU.
The CPU stays in the loop to know if the device is ready for transfer and has to continuously monitor the peripheral device.	There is no need for the CPU to stay in the loop as the interrupt command interrupts the CPU when the device is ready for data transfer.
This leads to the wastage of CPU cycles as CPU remains busy needlessly and thus the efficiency of system gets reduced.	The CPU cycles are not wasted as CPU continues with other work during this time and hence this method is more efficient.
CPU cannot do any work until the transfer is complete as it has to stay in the loop to continuously monitor the peripheral device.	CPU can do any other work until it is interrupted by the command indicating the readiness of device for data transfer
Its module is treated as a slow module.	Its module is faster than programmed I/O module.
It is quite easy to program and understand.	It can be tricky and complicated to understand if one uses low level language.
The performance of the system is severely degraded.	The performance of the system is enhanced to some extent.

Direct Memory Access(DMA)

DMA provides this capability to carry out memory specific operations with minimal CPU intervention. When any I/O device needs a memory access. It sends a DMA request(in form of interrupt) to CPU. CPU initiates the transfer by providing appropriate grant signals to the data bus. And passes the control to the DMA controller which controls the rest of the data transfer and transfers the data directly to I/O device. During this time, CPU continues with other instructions. Once the Read/Write operation in completed (or any exception is occurred)the DMA controller initiates an interrupt and notifies the processor about the status of read/write operation.

In this way the read/write operation is also carried out and CPU also executes some other instruction during that time. However, initialization of DMA still requires CPU intervention. And so the overall performance is maximized.

Section B

Short Questions

Attempt any TEN Questions: **(6×10=60)**

4. Draw an instruction cycle state diagram with interrupt and explain it.

Ans:

(i) Instruction address calculation (iac): Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.

For example, if each instruction is 16 bits long and memory is organized into 16-bit words, then add 1 to the previous address.

(ii) Instruction fetch (if): Read instruction from its memory location into the processor.

(iii) Instruction operation decoding (iod): Analyze instruction to determine type of operation to be performed and operand to be used.

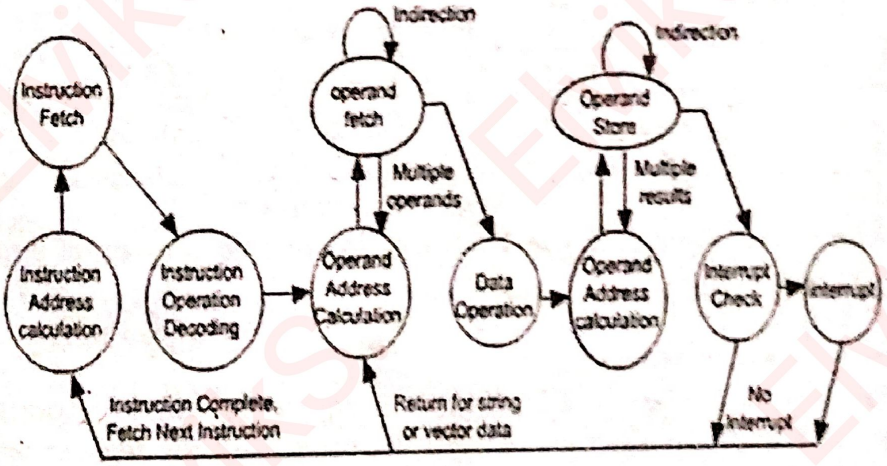


Figure: Instruction cycle state diagram with interrups

(iv) Operand address calculation (oac): If the operation involves reference to an operation in memory or available via I/O, then determine the address of the operand.

(v) Operand fetch (of): Fetch the operand from memory or read it in from I/O.

(vi) Data operation (dt): Perform the operation indicated in the instruction.

(vii) Operand store(os): Write the result into memory or out to I/O.

(viii) Fetch: Read the next instruction from memory into the processor.

(ix) Execute: Interpret the op-code and perform the indicated operation.

(x) Interrupt: If interrupts are enabled and an interrupt has occurred, save the current process state and service the interrupt.
The main line of activity consists of alternating instruction fetch and instruction execution activities. After an instruction is fetched, it is examined to determine if any indirect addressing is involved. If so, the required operands are fetched using indirect addressing following execution, an interrupt may be processed before the next instruction fetch.

5. **Explain register transfer language with example.**

Ans: The term register transfer means the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register.
The word language is borrowed from programmers who apply this term to programming languages. This programming language is a procedure for writing symbols to specify a given computational process.
Information transferred from one register to another is designated in symbolic form by means of replacement operator.
R2 ← R1
It denotes the transfer of the data from register R1 into R2.

Basic symbols for Register Transfer Language

Symbol	Description	Examples
Letters & numerals	Denotes a register	MAR, R2
Parenthesis ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two micro operations	R2 ← R1, R1 ← R2

Example:

Symbolic Designation	Description
R3 ← R1 + R2	Contents of R1+R2 transferred to R3.
R3 ← R1 - R2	Contents of R1-R2 transferred to R3.
R2 ← (R2)'	Compliment the contents of R2.
R2 ← (R2)' + 1	2's compliment the contents of R2.
R3 ← R1 + (R2)' + 1	R1 + the 2's compliment of R2 (subtraction).
R1 ← R1 + 1	Increment the contents of R1 by 1.
R1 ← R1 - 1	Decrement the contents of R1 by 1.

6. **Write codes using 3, 2 and 1 address instruction formats to perform the given operations.**
X=(A+B)*(C+D)

Ans: One Address Instructions
This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.
Expression: X = (A+B)*(C+D)
AC is accumulator
M[] is any memory location
M[T] is temporary location
LOAD A AC = M[A]
ADD B AC = AC + M[B]
STORE T M[T] = AC
LOAD C AC = M[C]

ADD D AC = AC + M[D]
MUL T AC = AC * M[T]
STORE X M[X] = AC

Two Address Instructions

This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address.

Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

Three Address Instructions

This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

7. Explain the various addressing modes with example.

Ans: The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

- To give the programming versatility to the user.
- To reduce the number of bits in addressing field of instruction.

Types of Addressing Modes

Below we have discussed different types of addressing modes one by one:

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.

Example: The instruction "Complement Accumulator" is an implied mode instruction.

In a stack organized computer, Zero Address Instructions are implied mode instructions.

Stack Addressing Mode

In this addressing mode,

- The operand is contained at the top of the stack.

Example

ADD

This instruction simply pops out two symbols contained at the top of the stack.

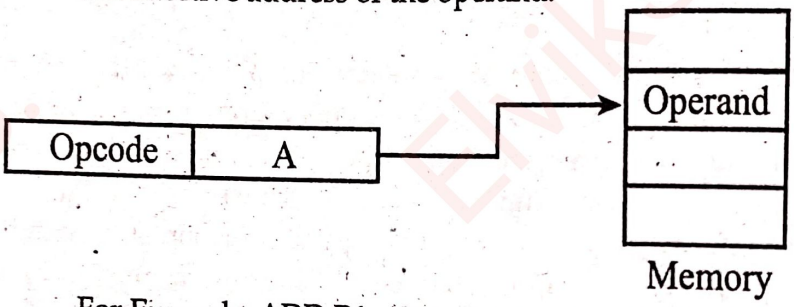
Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself. Single memory reference to access data. No additional calculations to find the effective address of the operand.

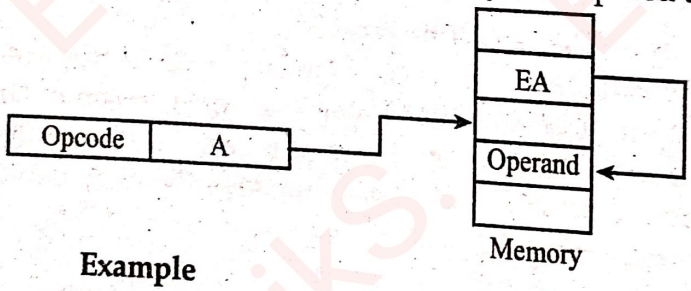


For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

Indirect Addressing Mode

In this addressing mode,

- The address field of the instruction specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Example

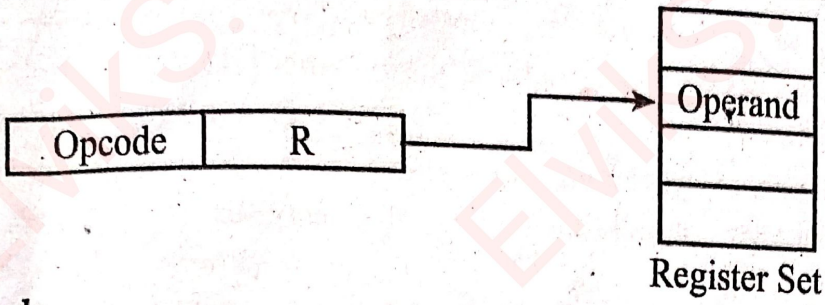
ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

AC ← AC + [[X]]

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.



Example

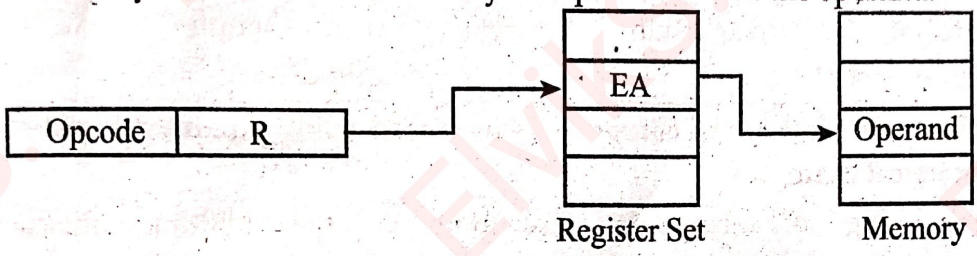
ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

Register Indirect Addressing Mode

In this addressing mode

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.



Example

ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

AC ← AC + [[R]]

8. **Differentiate between hardwired control unit and micro-programmed control unit.**

Ans: To execute an instruction, there are two types of control units Hardwired Control unit and Micro-programmed control unit.

- Hardwired control units are generally faster than micro programmed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA circuit.
- A micro programmed control unit is a relatively simple logic circuit that is capable of (1) sequencing through microinstructions and (2) generating control signals to execute each microinstruction.

Hardwired Control Unit	Micro programmed Control Unit
1. Hardwired control unit generates the control signals needed for the processor using logic circuits.	1. Micro-programmed control unit generates the control signals with the help of micro instructions stored in control memory.
2. Hardwired control unit is faster when compared to micro programmed control unit as the required control signals are generated with the help of hardware.	2. This is slower than the other as micro instructions are used for generating signals here.

3. Difficult to modify as the control signals that need to be generated are hard wired	3. Easy to modify as the modification need to be done only at the instruction level
4. More costlier as everything has to be realized in terms of logic gates.	4. Less costlier than hardwired control as only micro instructions are used for generating control signals.
5. It cannot handle complex instructions as the circuit design for it becomes complex.	5. It can handle complex instructions.
6. Only limited number of instructions are used due to the hardware implementation	6. Control signals for many instructions can be generated.
7. Used in computer that makes use of Reduced Instruction Set Computers(RISC).	7. Used in computer that makes use of Complex Instruction Set Computers(CISC).

9. How performance of computer is increased using pipeline? Explain with practical example.

Ans: Pipelining organizes the execution of the multiple instructions simultaneously. Pipelining improves the throughput of the system. In pipelining the instruction is divided into the subtasks. Each subtask performs the dedicated task.

The instruction is divided into 5 subtasks: instruction fetch, instruction decode, operand fetch, instruction execution and operand store. The instruction fetch subtask will only perform the instruction fetching operation, instruction decode subtask will only be decoding the fetched instruction and so on the other subtasks will do.

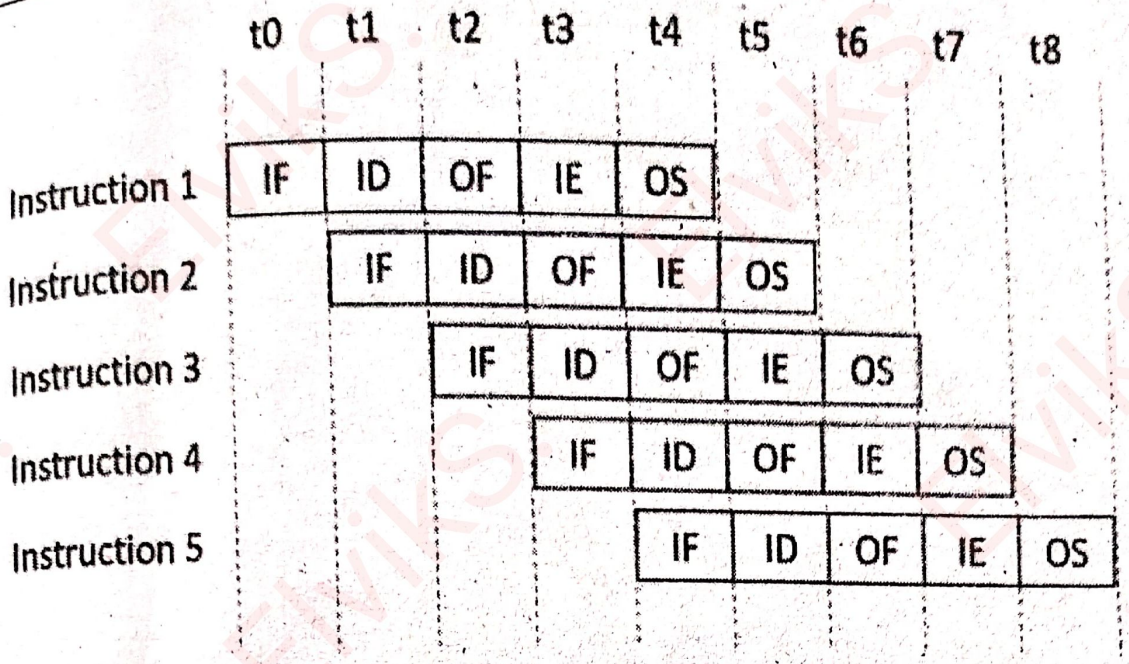
The output of the first pipeline becomes the input for the next pipeline. It is like a set of data processing unit connected in series to utilize processor up to its maximum.

An instruction in a process is divided into 5 subtasks likely,

- In the first subtask, the instruction is fetched.
- The fetched instruction is decoded in the second stage.
- In the third stage, the operands of the instruction are fetched.
- In the fourth, arithmetic and logical operation are performed on the operands to execute the instruction.
- In the fifth stage, the result is stored in memory.

Now, understanding the division of the instruction into subtasks. Let us understand, how the n number of instructions in a process, are pipelined.

Look at the figure below the 5 instructions are pipelined. The first instruction gets completed in 5 clock cycle. After the completion of first instruction, in every new clock cycle, a new instruction completes its execution.



Observe that when the Instruction fetch operation of the first instruction is completed in the next clock cycle the instruction fetch of second instruction gets started. This way the hardware never sits idle it is always busy in performing some or other operation. But, no two instructions can execute their same stage at the same clock cycle.

10. **Differentiate between restoring division and non-restoring division.**

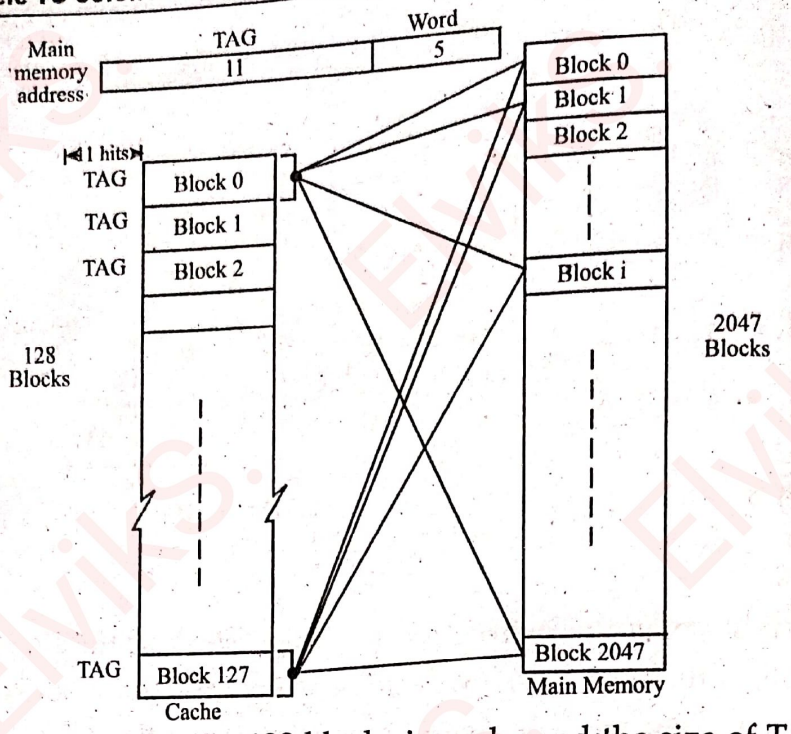
Ans: When we divide two numbers using a division algorithm, we get a quotient and a remainder. Slow algorithms and fast algorithms are the two most common types. Slow division algorithms include restoring, non-restoring, non-performing restoring, and the SRT algorithm, while Newton-Raphson and Goldschmidt fall within the quick division category.

Non-restoring division is less complex than restoring division since it involves basic operations such as addition and subtraction, as well as the addition and subtraction of numbers, as well as the addition and subtraction of numbers. Rely on the sign bit of the register, which is initially set to zero and is designated as A, in the technique.

11. **Given the appropriate reasons why replacement algorithm is required in associative mapping?**

Ans : In the associative mapping, any block of main memory can go to any block of cache, so it has got the complete flexibility and we have to use proper replacement policy to replace a block from cache if the current accessed block of main memory is not present in cache.

It might not be practical to use this complete flexibility of associative mapping technique due to searching overhead, because the TAG fiel of main memory address has to be compared with the TAG field of all the cache block.



In this example, there are 128 blocks in cache and the size of TAG is 11 bits. The whole arrangement of Associative mapping technique is shown in the figure above.

12. Differentiate between isolated versus memory mapped I/O.

Ans: Memory Mapped I/O and Isolated I/O are two methods of performing input-output operations between CPU and installed peripherals in the system. Memory mapped I/O uses the same address bus to connect both primary memory and memory of hardware devices. Thus the instruction to address a section or portion or segment of RAM can also be used to address a memory location of a hardware device.

On the other hand, isolated I/O uses separate instruction classes to access primary memory and device memory. In this case, I/O devices have separate address space either by separate I/O pin on CPU or by entire separate bus. As it separates general memory addresses with I/O devices, it is called isolated I/O.

Differences Between Isolated I/O and Memory Mapped I/O

Isolated I/O	Memory Mapped I/O
1. Isolated I/O uses separate memory space.	1. Memory mapped I/O uses memory from the main memory.
2. Limited instructions can be used. Those are IN, OUT, INS, OUTS.	2. Any instruction which references to memory can be used.
3. The addresses for Isolated I/O devices are called ports.	3. Memory mapped I/O devices are treated as memory locations on the memory map.
4. IORC & IOWC signals expands the circuitry.	4. IORC & IOWC signals has no functions in this case which reduces the circuitry.
5. Efficient I/O operations due to using separate bus	5. Inefficient I/O operations due to using single bus for data and addressing
6. Comparatively larger in size	6. Smaller in size
7. Uses complex internal logic	7. Common internal logic for memory and I/O devices
8. Slower operations	8. Faster operations

TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Course Title: Computer Architecture
Course No: CSC201
Nature of the Course: Theory + Lab
Semester: III

Full Marks: 80
Pass Marks: 32
Credit Hrs.: 3

Computer Architecture

TU QUESTIONS-ANSWERS 2078

Section A

Long Questions

Attempt any TWO Questions: (2×10=20)

1. **Differentiate between hardwired and micro-programmed control unit. Describe with example of micro-program sequencer used in micro-programmed control unit.**

Ans: To execute an instruction, there are two types of control units Hardwired Control unit and Micro-programmed control unit.

Hardwired control units are generally faster than micro-programmed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA circuit.

A micro-programmed control unit is a relatively simple logic circuit that is capable of:

1. sequencing through microinstructions and
2. generating control signals to execute each microinstruction.

Hardwired Control Unit	Micro-programmed Control Unit
1. Hardwired control unit generates the control signals needed for the processor using logic circuits.	1. Micro-programmed control unit generates the control signals with the help of micro instructions stored in control memory.
2. Hardwired control unit is faster when compared to micro-programmed control unit as the required control signals are generated with the help of hardware.	2. This is slower than the other as micro instructions are used for generating signals here.
3. Difficult to modify as the control signals that need to be generated are hard wired.	3. Easy to modify as the modification need to be done only at the instruction level.
4. More costlier as everything has to be realized in terms of logic gates.	4. Less costlier than hardwired control as only micro instructions are used for generating control signals.
5. It cannot handle complex instructions as the circuit design for it becomes complex.	5. It can handle complex instructions.
6. Only limited number of instructions are used due to the hardware implementation.	6. Control signals for many instructions can be generated.

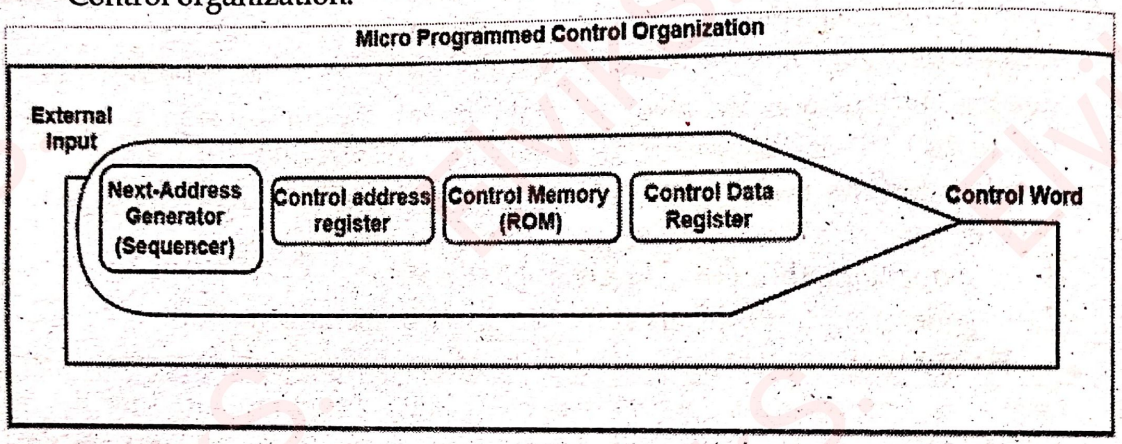
7. Used in computer that makes use of Reduced Instruction Set Computers (RISC).	7. Used in computer that makes use of Complex Instruction Set Computers (CISC).

A control unit whose binary control values are saved as words in memory is called a micro-programmed control unit.

A controller results in the instructions to be implemented by constructing a definite collection of signals at each system clock beat. Each of these output signals generates one micro-operation including register transfer. Thus, the sets of control signals are generated definite micro-operations that can be saved in the memory.

Each bit that forms the microinstruction is linked to one control signal. When the bit is set, the control signal is active. When it is cleared the control signal turns inactive. These microinstructions in a sequence can be saved in the internal 'control' memory. The control unit of a micro-program-controlled computer is a computer inside a computer.

The following image shows the block diagram of a Micro-programmed Control organization.



There are the following steps followed by the micro-programmed control are:

- It can execute any instruction. The CPU should divide it down into a set of sequential operations. This set of operations are called microinstruction. The sequential micro-operations need the control signals to execute.
- Control signals saved in the ROM are created to execute the instructions on the data direction. These control signals can control the micro-operations concerned with a microinstruction that is to be performed at any time step.
- The address of the microinstruction is executed next is generated.
- The previous 2 steps are copied until all the microinstructions associated with the instruction in the set are executed.

2. Explain the various types of addressing modes and compare them algorithm, advantage and disadvantage.

Ans: The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

- To give the programming versatility to the user.
- To reduce the number of bits in addressing field of instruction.

Types of Addressing Modes

Below we have discussed different types of addressing modes one by one:

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.

Example: The instruction "Complement Accumulator" is an implied mode instruction.

In a stack organized computer, Zero Address Instructions are implied mode instructions.

Stack Addressing Mode

In this addressing mode,

- The operand is contained at the top of the stack.

Example

ADD

This instruction simply pops out two symbols contained at the top of the stack.

Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

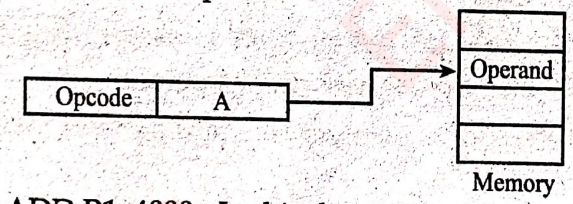
For example: ADD 7; which says Add 7 to contents of accumulator. 7 is the operand here.

Advantage of immediate addressing mode: No memory reference other than the instruction fetch is required to obtain the operand.

Disadvantage of immediate addressing mode: Size of the number is restricted to the size of the address field.

Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself. Single memory reference to access data. No additional calculations to find the effective address of the operand.



For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

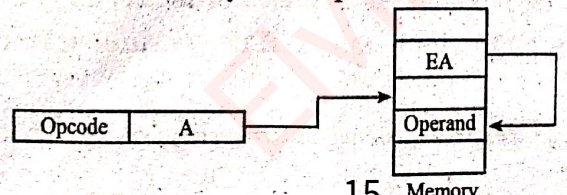
Advantage of direct addressing mode: It requires only one memory reference and no special calculation.

Disadvantages of direct addressing mode: It provides only a limited address space.

Indirect Addressing Mode

In this addressing mode,

- The address field of the instruction specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Example

ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

AC ← AC + [[X]]

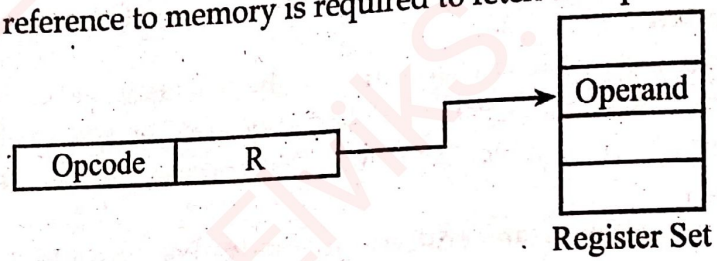
Advantage of indirect addressing mode: Address range is not limited by address field.

Disadvantage of indirect memory address: Instruction execution requires two memory references to fetch the operand: one to get its address and a second to get its value.

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.



Example

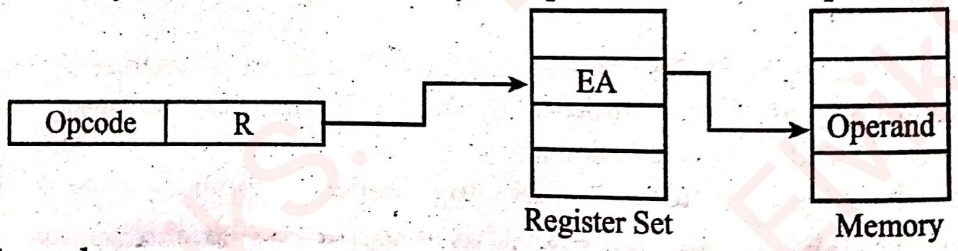
ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

Register Indirect Addressing Mode

In this addressing mode

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.



Example

ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

AC ← AC + [[R]]

Advantages of register direct addressing mode: Only a small address field is needed in the instruction, and no time consuming memory references are required.

Disadvantage of register direct addressing mode: Address space is very limited.

Difference between Register Direct Addressing Mode and Register Indirect Addressing Mode:

Register Direct Addressing Mode	Register Indirect Addressing Mode
1. Operand is placed in register.	1. Operand is placed in Main memory.
2. Registers are used to store operands.	2. Registers are used to store address of operand in main memory.
3. Address field contains effective address of operand.	3. Address field contains reference to the effective address of operand.
4. One register reference for operand.	4. Two register reference for operand.
5. Fast	5. Slow
6. Easier to access data	6. Complex
7. Limitation of address field to store operand address.	7. There is no such limit.

Difference between Direct Addressing Mode and Indirect Addressing Mode:

Direct Addressing Mode	Indirect Addressing Mode
1. Address field contains effective address of operand.	1. Address field contains memory location where effective address is present
2. One memory reference to access operand	2. Two memory reference to access operand
3. Fast	3. Slow
4. Address size of operand is limited to size of address field.	4. No such limitation. Because operand address is stored in main memory.
5. Less calculation to access operand	5. More calculation to access operand.

Difference between Immediate Addressing Mode and Direct Addressing Mode:

Immediate Addressing Mode	Direct Addressing Mode
1. Address field contains operand	1. Address field contains effective address of operand
2. Operand is in address field	2. Operand is present in main memory
3. No memory reference	3. One memory reference
4. Fast	4. Slow
5. Operand size depends on size of address field	5. No limitation on operand size

Difference between Immediate Addressing Mode and Indirect Addressing Mode:

Immediate Addressing Mode	Indirect Addressing Mode
1. Address field contains operand	1. Address field contains memory reference of operand
2. There is no memory reference	2. There is two memory reference
3. Operand size depends on size of address field in instruction	3. There is no such limit on operand size
4. Fast	4. Slow
5. Operand present in address field	5. Operand is present in main memory

3. Explain the non-restoring division algorithm with flow chart and hardware implementation diagram. Divide 11/3 using restoring division.

Ans: A division algorithm provides a quotient and a remainder when we divide two number. They are generally of two type slow algorithm and fast algorithm. Slow division algorithm are restoring, non-restoring, non-performing restoring, SRT algorithm and under fast comes Newton-Raphson and Goldschmidt.

The non-restoring flowchart algorithm is as given below:

Algorithm

Step 1: If the sign of A is 0, shift A and Q left one bit position and subtract divisor from A; otherwise, shift A and Q left and add divisor to A.

Step 2: Repeat steps 1 and 2 for n times.

Step 3: If the sign of A is 1, add divisor to A.

Note: Step 3 is required to leave the proper positive remainder in A at the end of n cycles.

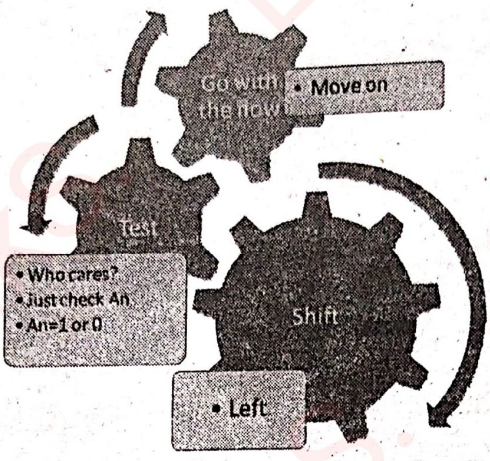
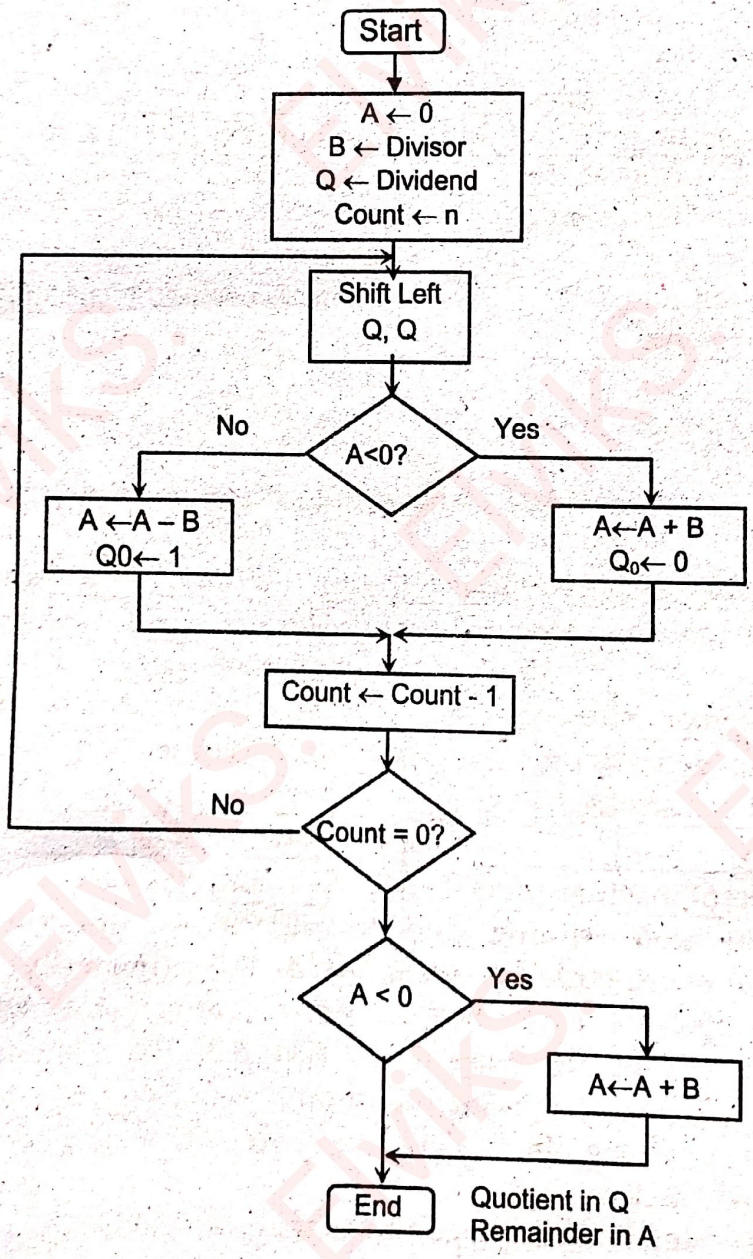


Figure Non-restoring division operation steps



Example: Perform Division Restoring Algorithm
Dividend = 11
Divisor = 3

n	M	A	Q	Operation
4	00011	00000	1011	initialize
	00011	00001	011_	shift left AQ
	00011	11110	011_	A=A-M
	00011	00001	0110	Q[0]=0 And restore A
3	00011	00010	110_	shift left AQ
	00011	11111	110_	A=A-M
	00011	00010	1100	Q[0]=0
2	00011	00101	100_	shift left AQ
	00011	00010	100_	A=A-M
	00011	00010	1001	Q[0]=1
1	00011	00101	001_	shift left AQ
	00011	00010	001_	A=A-M
	00011	00010	0011	Q[0]=1

Remember to restore the value of A most significant bit of A is 1. As that register Q contain the quotient, i.e. 3 and register A contain remainder 2.

Section B

Short Questions

Attempt any TEN Questions: **(6×10=60)**

4. Explain the bus interconnection scheme with diagram.

Ans: A bus is a communication pathway connecting two or more devices. A key characteristic of a bus is that it is a shared transmission medium. Multiple devices connect to the bus, and a signal transmitted by any one device is available for reception by all other devices attached to the bus. If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit.

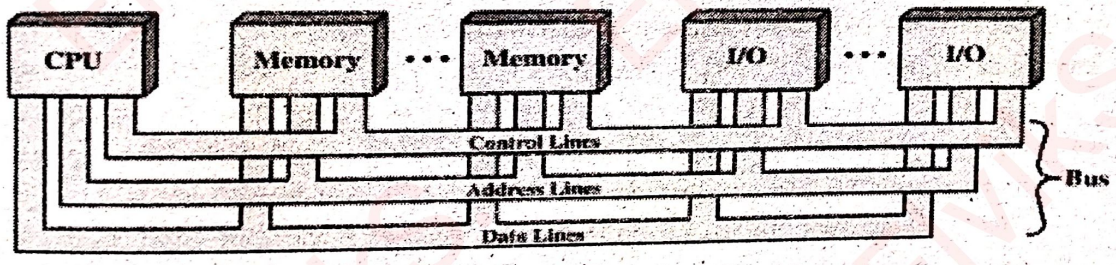


Figure: A bus interconnection scheme

Typically, a bus consists of multiple communication pathways, or lines. Each line is capable of transmitting signals representing binary q and binary 0. An 8-bit unit of data can be transmitted over eight bus lines. A bus that connects major computer components is called a system bus.

On any bus the lines can be classified into three functional groups. Data, address and control lines. In addition, there may be power distribution lines that supply power to the attached modules.

The data lines provide a path for moving data among system modules. These lines, collectively are called the data bus.

The address lines are used to designate the source or destination of the data on the bus. For example, on an 8-bit address bus, address 01111111 and

below might reference locations in a memory module with 128 words of memory, and address 10000000 and above refer to devices attached to an I/O module.

The control lines are used to control the access to and the use of the data and address lines. Control signals transmit both command and timing information among system modules.

Timing signals indicate the validity of data and address information. Command signals specify operations to be performed. Typical control lines include

- **Memory Write:** Causes data on the bus to be written into the addressed location
- **Memory read:** Causes data from the addressed location to be placed on the bus.
- **I/O write:** Causes data on the bus to be output to the addressed I/O prot.

5. What do you mean by instruction format? Explain with an example.

Ans: An instruction format defines the different component of an instruction. The main components of an instruction are opcode (which instruction to be executed) and operands (data on which instruction to be executed).

Opcode: The operation code (opcode) represents action that the processor must execute. It tells the processor what basic operations to perform.

Operands: The operand code defines the parameters of the action and depends on the operation. It specifies the locations of the data or the operand on which the operation is to be performed. It can be data or a memory address.

Operation Code	Operand code

Fig: Instruction Format

E.g. ADD B
// Add the contents of register B to the content of the accumulator.
MOV C, A // Copy the content of accumulator in the register C.

6. Explain the data transfer instructions with example.

Ans: Data transfer instructions transfer the data between memory and processor registers, processor registers, and I/O devices, and from one processor register to another. There are eight commonly used data transfer instructions. Each instruction is represented by a mnemonic symbol.

The table shows the eight data transfer instructions and their respective mnemonic symbols.

Data Transfer Instructions

Name	Mnemonic Symbols
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	In
Output	OUT
Push	PUSH
Pop	POP

All these instructions are associated with a variety of addressing modes. Some assembly language instructions use different mnemonic symbols just to differentiate between the different addressing modes.

7. Explain the symbolic microinstruction with example.

Ans: The microinstructions can be determined by symbols. It is interpreted to its binary format with an assembler. The symbols should be represented for each field in the microinstruction. The users should be enabled to represent their symbolic addresses. Each line in an assembly language represents symbolic instruction. These instructions are divided into five fields such as label, micro-operations, CD, BR, and AD.

The fields that specify the following information are as follows:

- The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:).
- The micro-operations field consists of one, two, or three symbols, separated by commas. But each F field includes only a single symbol.
- The CD field has one of the letters U, I, S, or Z.
- The BR field contains one of the four symbols defined.
- The AD field specifies a value for the address field of the microinstruction in one of three possible ways:
  - ✓ With a symbolic address, which must also appear as a label.
  - ✓ With the symbol NEXT to designate the next address in sequence.
  - ✓ When the BR field includes a RET or MAP symbol, the AD field is left null and is transformed to seven zeros by the assembler.

Microinstructions that are situated in addresses 64, 65, and 66 are important for the fetch routine. There are various symbolic language is as follows:

	ORG 64
FETCH:	PCTAR U JMP NEXT
	READ, INCPC U JMP NEXT
	DRTAR U MAP

The table shows the results of binary translation for the assembly language.

Binary Translation for Assembly Language

Binary Address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	101	000	00	11	0000000

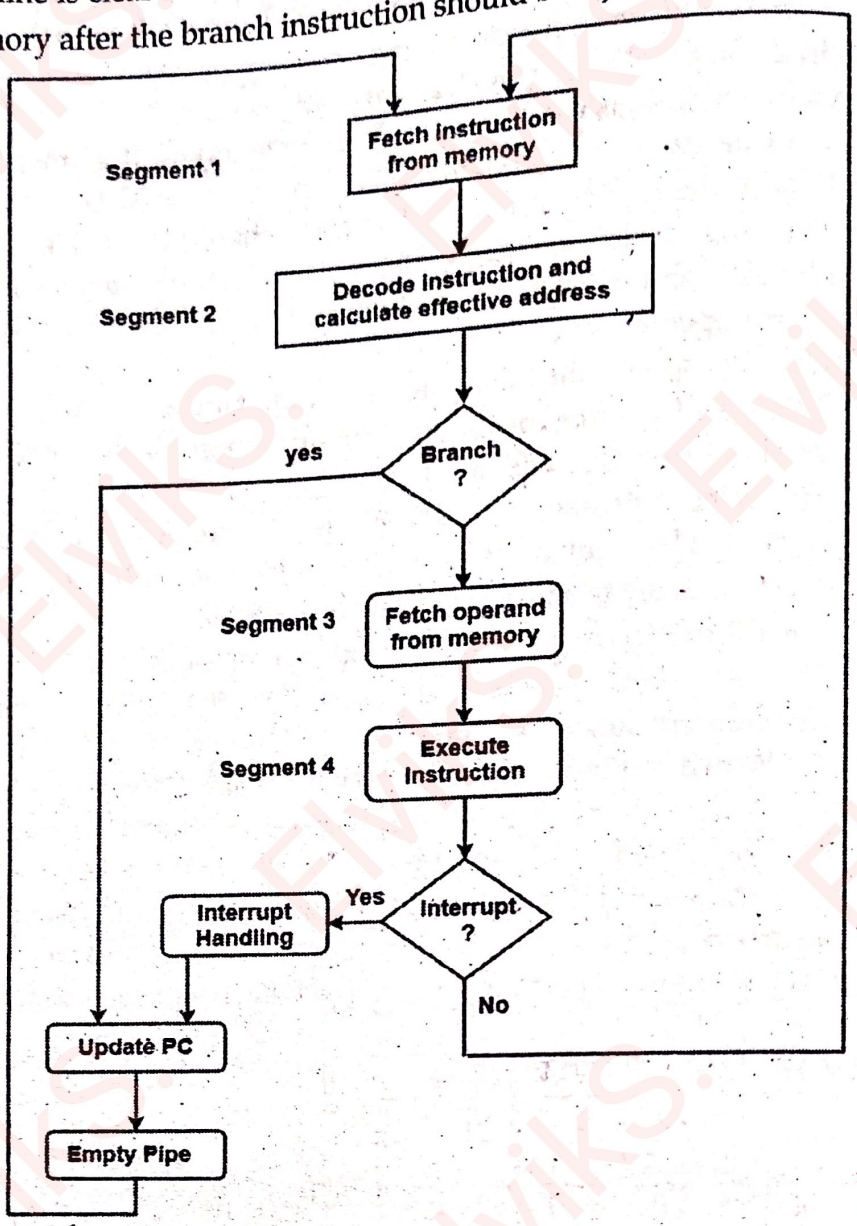
Each microinstruction executes the internal register transfer operation displayed by the register transfer representation. The representation in symbols is important while writing micro-programs in an assembly language format. The actual internal content which is saved in the control memory is in binary representation.

8. Explain an instruction pipeline with an example.

Ans: An instruction pipeline reads consecutive instructions from memory while in the other segments the previous instructions are being implemented. Pipeline processing appears both in the data flow and in the instruction stream. This leads to the overlapping of the fetch and executes the instruction and hence simultaneous operations are performed.

There is one possible more event associated with such a design is that instruction can generate a branch out of a sequence. In this method, the

pipeline is clear and all the instructions that have previously been read from memory after the branch instruction should be rejected.



Segment 1

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

Segment 2

The instruction fetched from memory is decoded in the second segment. The effective address is computed in an independent arithmetic circuit.

Segment 3

An operand from memory is fetched in the third segment.

Segment 4

The instructions are finally implemented in the final segment of the pipeline organization.

9. Explain the Booth multiplication algorithm with example.

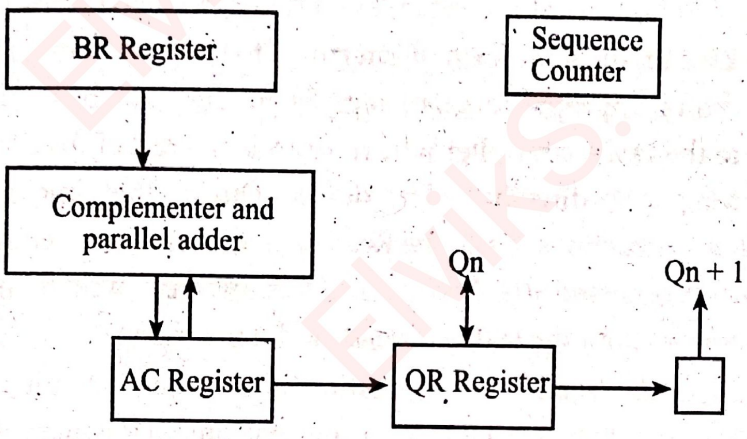
Ans: Booth algorithm gives a procedure for **multiplying binary integers** in signed 2's complement representation **in efficient way**, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{(k+1)}$ to 2^m. As in all multiplication schemes, booth algorithm requires examination of the **multiplier bits** and shifting of the partial product. Prior to the shifting, the

multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Hardware Implementation of Booths Algorithm

The hardware implementation of the booth algorithm requires the register configuration shown in the figure below.



10. What are the advantage and disadvantage of direct mapping and associative mapping between cache and main memory?

Ans: Associative Mapping

In associative mapping both the address and data of the memory word are stored. The associative mapping method used by cache memory is very flexible one as well as very fast. This mapping method is also known as fully **associative cache.**

Advantages of associative mapping

- Associative mapping is fast.
- Associative mapping is easy to implement.

Disadvantages of associative mapping

- Cache Memory implementing associative mapping is expensive as it requires to store address along with the data.

Direct Mapping

In direct mapping cache, instead of storing total address information with data in cache only part of address bits is stored along with data. The new data has to be stored only in a specified cache location as per the mapping rule for direct mapping. So it doesn't need replacement algorithm.

Advantages of direct mapping

- Direct mapping is simplest type of cache memory mapping.

- Here only tag field is required to match while searching word that is why it fastest cache.
- Direct mapping cache is less expensive compared to associative cache mapping.

Disadvantages of direct mapping

- The performance of direct mapping cache is not good as requires replacement for data-tag value.

11. What are the major differentiate between input-output processor (IOP) and direct memory access (DMA).

Ans: Direct Memory Access(DMA):

DMA provides this capability to carry out memory specific operations with minimal CPU intervention. When any I/O device needs a memory access. It sends a DMA request(in form of interrupt) to CPU. CPU initiates the transfer by providing appropriate grant signals to the data bus. And passes the control to the DMA controller which controls the rest of the data transfer and transfers the data directly to I/O device. During this time, CPU continues with other instructions. Once the Read/Write operation in completed (or any exception is occurred)the DMA controller initiates an interrupt and notifies the processor about the status of read/write operation.

In this way the read/write operation is also carried out and CPU also executes some other instruction during that time. However, initialization of DMA still requires CPU intervention. And so the overall performance is maximized.

I/O processor

The I/O processor, generally used in large computer systems, is a coprocessor which is capable of executing the instructions in addition to transfer of data. By the way, the coprocessor instruction system is different from the central processing unit.

CPU can execute the I/O specific program by initializing the basic operations like enabling the data path and setting up the I/O devices participating in operation. And then it transfers the task to I/O processor, which then carry out rest of the tasks and upon completion notifies the processor. The processor meanwhile executes other important instructions.

The I/O processor is essentially a small DMA dedicated processor that can execute limited input and output instructions and can be shared by multiple peripherals.

12. Draw a tree dimensional hypercube and explain with example.

Ans: Hypercube (or Binary n-cube multiprocessor) structure represents a loosely coupled system made up of N=2n processors interconnected in an n-dimensional binary cube. Each processor makes a made of the cube. Each processor makes a node of the cube. Therefore, it is customary to refer to each node as containing a processor, in effect it has not only a CPU but also

local memory and I/O interface. Each processor has direct communication paths to n other neighbor processors. These paths correspond to the cube edges.

There are 2 distinct n-bit binary addresses which can be assigned to the processors. Each processor address differs from that of each of its n neighbors by exactly one bit position.

- Hypercube structure for n= 1, 2 and 3.
- A one cube structure contains n = 1 and 2n = 2.
- It has two processors interconnected by a single path.
- A two-cube structure contains n=2 and 2n=4.
- It has four nodes interconnected as a cube.
- An n-cube structure contains 2n nodes with a processor residing in each node.

Each node is assigned a binary address in such a manner, that the addresses of two neighbors differ in exactly one bit position. For example, the three neighbors of the node with address 100 are 000, 110, and 101 in a three-cube structure. Each of these binary numbers differs from address 100 by one bit value.

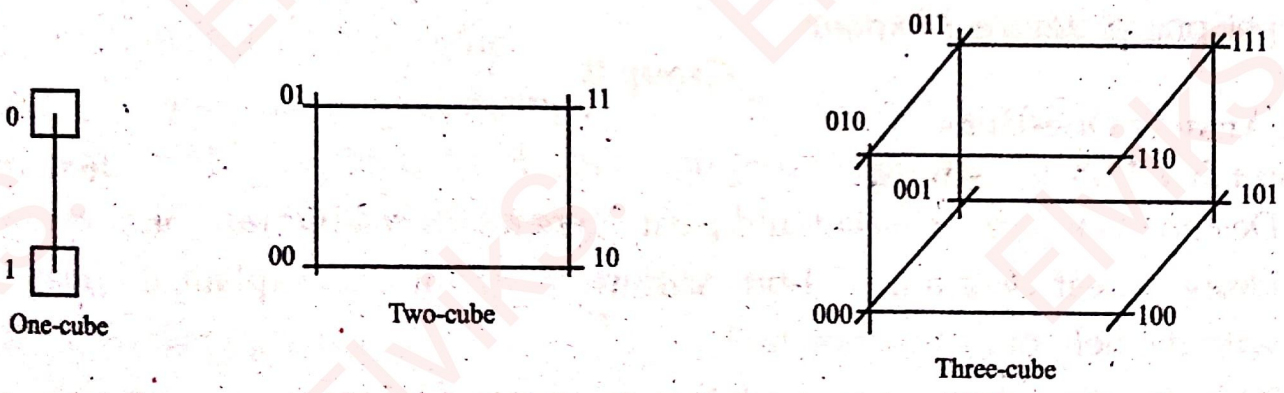


Figure: Hypercube structure for n=1, 2 and 3

❑❑❑