11 Processor Instructions

11.1 : Elements of Machine Insruction

Q.1 What do you mean by machine instructions ?

☞ [SPPU : June-22, Marks 3]

Ans. : • The operation of the central processing unit and the computer system is determined by the instructions executed by central processing unit. These instructions are known as **machine instructions** or **computer instructions.**

- Machine instructions are in the form of binary codes.
- A particular sequence of these binary codes used to perform particular task is known as machine language program.

Q.2 What are the elements of machine instructions ?

☞ [SPPU : June-22, Marks 3]

Ans. : Each instruction of the CPU has specific information fields, which are required to execute it. These information fields of machine instructions are called **elements of machine instruction.**

Elements of machine instructions are :

- **Operation code :** Specifies the operation to be performed. The operation is specified by binary code, hence the name operation code or simply opcode.
- **Source / Destination operand :** The source/destination operand field directly specifies the source/destination operand for the instruction.
- **Source operand address :** The operation specified by the instruction may require one or more source operands.
- **Destination operand address :** The operation executed by the CPU may produce result. Usually, the result is stored in the destination operand.

- **Next instruction address** : The next instruction address tells the CPU from where to fetch the next instruction after completion of execution of current instruction.

11.2 : Instruction Representation

Q.3 Define opcode and mnemonics.

Ans. : • The **opcode** of an instruction is the first byte or two that tells the CPU what type the instruction is (ex. add, sub, move, jump) and the types of operands to expect (register, memory address).

- A **mnemonic** is simply the human readable symbol that represents the opcode.

Q.4 Write a note on assembly language elements.

Ans. : The four elements of assembly language are :

1. **The label field :** The first field, which is optional, is the label field, used to specify a symbolic address. A label is an identifier that is assigned to the address of the first byte of the instruction in which it appears. The presence of a label is optional, but if present, the label provides a symbolic name that can be used in branch instructions to branch to the instruction.
2. **The mnemonic / instruction field :** The second field is mnemonic / instruction which is compulsory. It specifies a machine instruction or a pseudo instruction.
3. **Operand Field :** The presence of the operands depends on the instruction. Some instructions have no operands, some have one and some have two. If there are two operands, they are separated by a comma.
4. **The comment field :** The last field is a comment field. It may be empty or it may include a comment. It begins with a delimiter such as the semicolon and continues to the end of the line. The comments are for our benefit. They tell us what the program is trying to accomplish. They are useful for explaining the program and are helpful in understanding the step-by-step procedure taken by the program. They are used only for explanation and are neglected while translating program to binary.

11.3 : Instruction Formats (0-1-2-3 Address Formats)

Q.5 What are the common fields in the instruction format ?

Ans. : The most common fields in instruction formats are :

1. **Opcode field :** It specifies operation to be performed.
2. **Address field :** It designates a memory address or a processor register.
3. **Mode field :** It specifies the way the operand or the effective address is determined.
4. **Addressing mode field :** It specifies the addressing mode to be used to access operand.

Q.6 If an instruction code has 4 bit opcode and 12 bit address field then

i) How many operations this code can perform.

ii) How many memory locations can be addressed.

Ans. :

i) Number of operations can be performed $= 2^4 = 16$

ii) Number of memory locations can be addressed $= 2^{12} = 4096$

Q.7 A computer uses a memory unit with 256 K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts : An indirect address, an operation code, the register code part to specify one of 64 registers and an address part : 10

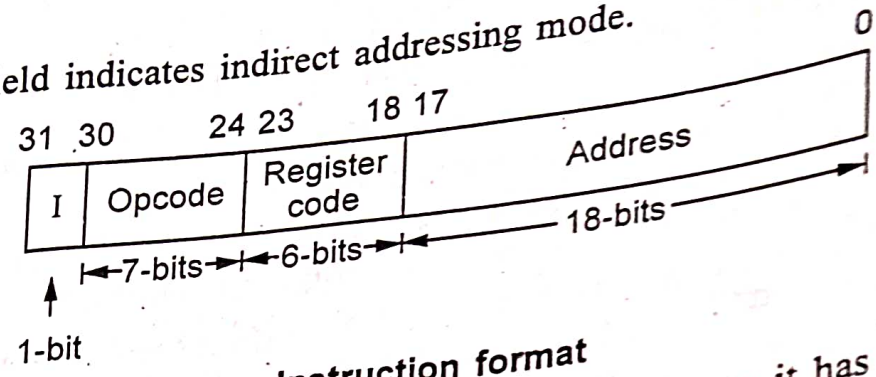
i) How many bits are there in the operation code, the register code part and the address part ? ii) Draw the instruction word format and indicate the number of bits in each part.

iii) How many bits are there in the data and address inputs of the memory ?

Ans. : i) The total bits specified in the instruction are 32. For address part we need 18 bits ($2^{18} = 256$ K) , for register code part we need 6 bits ($2^6 = 64$) and for I field we need 1 bit. The remaining bits ($32 - 18 - 1 - 6 - 1) = 7$) are used for opcode.

ii) The instruction format for the given specification is as shown in the below

The I field indicates indirect addressing mode.



Instruction format

iii) The memory has 256 K memory locations and hence it has 18 address inputs. Since it stores 32-bits words, its data inputs are 32.

Q.8 The memory unit of a computer has 256 k works of 32 bit each. The computer has an instruction format with 4 fields. An operation field a mode field to specify one of seven addressing modes, a register address a field to specify one of 60 processor registers and a memory address. Specify the instruction format and the number of bits in each field if the instruction is in one memory word.

Ans. : To address 256 K words we need 18 (2^{18} = 256 K) address lines.

To specify seven addressing modes we need 3 bits.

To specify 60 processor registers we need 6 bits

Number of opcode bits = 32 – 18 – 3 – 6 = 5

Instruction format

Opcode	Mode	Regiser	Address
5	3	6	18

= 32

Q.9 Write a short note on instruction types.

☞ [SPPU : Dec.-08, Marks 4, June-22, Marks 3]

OR Explain different types of instruction format according to address references.

Ans. : According to address references there are four types of instruction formats.

- Three address,
- Two address,
- One address and
- Zero address reference instructions.
- **Three address instructions :** The three address instruction can be represented symbolically as ADD C, A, B where A, B, C are the variables. (C ← A + B) These variable names are assigned to distinct locations in the memory. In this instruction operands A and B are called source operands and operand C is called destination operand and ADD is the operation to be performed on the operands.
- **Two address instructions** : The two address instruction can be represented symbolically as ADD A, B. This instruction adds the contents of variables A and B and stores the sum in variable A destroying its previous contents (A ← A + B). Here, operand B is source operand; however operand A serves as both source and destination operand.
- **One address instruction** : The one address instruction can be represented symbolically as ADD B. This instruction adds the contents of variable B into the processor register called **accumulator** (AC) and stores the sum back into the accumulator destroying the previous contents of the accumulator (AC ← AC+B).
- **Zero address instructions** : In these instructions, the locations of all operands are defined implicitly. For example, CMA instruction complements the contents of accumulator (AC) and stores complemented contents back into the accumulator (AC ← $\overline{AC}$). Such instructions are also found in machines that store operands in a structure called a **pushdown stack.**

Q.10 Write a program to evaluate the arithmetic statement $Y = (A+B)*(C+D)$ using three-address, two-address, one-address and zero-address instructions.

Ans. : Using three address instructions

```
ADD  R1, A, B  ;  R1 ← M[A] + M[B]
ADD  R2, C, D  ;  R2 ← M[C] + M[D]
```

```
MUL  Y, R1, R2 ;   M[Y] ← R1 * R2
```

Using two address instructions

```
MOV  R1,   A     ;   R1 ← M[A]
ADD  R1,   B;        R1 ← R1 + M[B]
MOV  R2,   C     ;   R2 ← M[C]
ADD  R2,   D     ;   R2 ← R2 + M[D]
MUL  R1,   R2    ;   R1 ← R1 * R2
MOV  Y,    R1    ;   M[Y] ← R1
```

Using one address instruction

```
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  Y ; M[Y] ← AC
```

Using zero address instructions

```
PUSH   A ; TOS ← A
PUSH   B ; TOS ← B
ADD      ; TOS ← (A + B) - zero address instruction
PUSH   C ; TOS ← C
PUSH   D ; TOS ← D
ADD      ; TOS ← (C + D) - zero address instruction
MUL      ; TOS ← (C + D) * (A + B) ← zero address instruction.
POP    Y ; M[Y] ← TOS
```

Q.11 Write a program to evaluate the following arithmetic statement

$X = [A*(B+C)-D]/(E+F-G)$

i) Using a general register computer with three–address instructions,

ii) Using an accumulator type computer with one address instructions,

iii) Using a stack organized computer with zero–address operation instructions.

Q.12 Write the program to evaluate the expression

$X = \frac{A[B+C(D+E)]}{F(G+H)}$ **using the zero address instructions and one address instructions.**

Ans. : Program using zero address instructions

```
PUSH   D    ;  TOS ← D
PUSH   E    ;  TOS ← E
ADD         ;  TOS ← (D + E)
PUSH C      ;  TOS ← C
MUL         ;  TOS ← C × (D + E)
PUSH   B    ;  TOS ← B
ADD         ;  TOS ← B + C × (D + E)
PUSH   A    ;  TOS ← A
MUL         ;  TOS ← A [B + C × (D + E)]
PUSH   G    ;  TOS ← G
PUSH   H    ;  TOS ← H
ADD         ;  TOS ← G + H
PUSH   F    ;  TOS ← F
MUL         ;  TOS ← F × (G + H)
DIV         ;  TOS ← A[B + C × (D + E)]/F × (G + H)
POP    X    ;  M [X] ← TOS
```

Program using one address instructions

```
LOAD   H    ;  AC ← M[H]
ADD    G    ;  AC ← AC + M[G]
MUL    F    ;  AC ← AC * M[F]
STORE  T    ;  M[T] ← AC
LOAD   D    ;  AC ← M[D]
ADD    E    ;  AC ← AC + M[E]
MUL    C    ;  AC ← AC * M[C]
ADD    B    ;  AC ← AC + M[B]
MUL    A    ;  AC ← AC * M[A]
DIV    T    ;  AC ← AC/M[T]
STORE  X    ;  M[X] ← AC
```

Q.13 $X = A \times B + C \times C$

Explain how the above expression will be executed in one address, two address and three address processors in an accumulator organization.

Ans. : Using one address instruction

```
LOAD   A ;  AC ← M[A]
MUL    B ;  AC ← AC * M[B]
STORE  T ;  M[T] ← AC
LOAD   C ;  AC ← M[C]
MUL    C ;  AC ← AC * M[C]
ADD    T ;  AC ← AC + M[T]
STORE  X ;  M[X] ← AC
```

Using two address instructions

```
MOV  A, R1   ;  R1 ← M[A]
MUL  B, R1   ;  R1 ← R1 * M[B]
MOV  C, R2   ;  R2 ← M[C]
MUL  C, R2   ;  R2 ← R2 * M[C]
ADD R2, R1   ;  R1 ← R1 + R2
MOV  R1, X   ;  M[T] ← R1
```

Using three address instructions

```
MUL  A, B, R1 ;  R1 ← M[A] * M[B]
MUL  C, C, R2 ;  R2 ← M[C] + M[C]
ADD  R1, R1, X;  M[X] ←R1 * R2
```

11.4 : Types of Operands

Q.14 Write a note on type of operands. ☞ [SPPU : June-22, Marks 9]

Ans. : The operand may appear in different forms; these are -

- **Addresses** : The addresses are in fact a form of data. In many situations, some calculation must be performed on the operand reference in an instruction to determine physical address. In this context, addresses can be considered as unsigned integer operands.

- **Numbers** : All computer supports numeric data types. The common numeric data types are :
  - Integer or fixed point
  - Floating point
  - Decimal.
- **Characters** :
  - For documentation a common form of data is text or character strings.
  - Today, most of the computers use ASCII (American Standard Code for Information Interchange) code for character represented by a unique 7-bit pattern ; thus, 123 different characters can be represented.
  - However, the ASCII encoded characters are always stored and transmitted using 8-bits per character. The eighth bit may be set to 0 or used as a parity bit for error detection.
- **Logical data** :
  - Most of the processors interpret data as a bit, byte, word or double word. These are referred to as units of data.
  - When data is viewed as n 1-bit items of data, each item having the value 0 or 1, it is considered as a **logical data.**
  - The logical data is used to store an array of Boolean or binary data items and with logical data we can manipulate the bits of data items.

11.5 : Types of Operations/Instructions

Q.15 List the various types of operations supported by most of the processors.

Ans. : Generally type of operations supported by most of the machines can be categorized as follows :

- Data transfer operations
- Arithmetic operations

- Logical operations
- Conversion operations
- I/O operations
- System control operations
- Transfer of control operations.

Q.16 State any four data transfer operations.

Ans. : Data transfer operations :

Operation	Description
Move (transfer)	Transfers word or block from source to destination.
Store	Transfers word from processor to memory.
Load (fetch)	Transfers word from memory to processor.
Exchange	Swaps contents of source and destination.
Clear (reset)	Transfers word of 0s to destination.
Set	Transfers word of 1s to destination.
Push	Transfers word from source to top of stack.
Pop	Transfers word from top of stack to destination.

Q.17 State any four arithmetic operations.

Ans. : Arithmetic operations :

Operation	Description
Add	Performs addition of two operands.
Subtract	Performs subtraction of two operands.
Multiply	Performs multiplication of two operands.
Divide	Performs division of two operands.
Absolute	Replaces operand by its absolute value.

Negate	Changes sign of operand.
Increment	Adds 1 to operand.
Decrement	Subtracts 1 from operand.

Q.18 List any four logical operations.

OR State the difference between shift and rotate operations.

OR State the difference between logical shift and arithmetic shift operation.

Ans. : Logical operations :

Operation	Description
AND	Performs logical AND.
OR	Performs logical OR.
NOT (complement)	Performs logical NOT.
Exclusive - OR	Performs logical XOR.
Test	Tests specified condition and sets flag(s) accordingly.
Compare	Makes logical or arithmetic comparison or two of more operands and sets flag(s) accordingly.
Set control variables	Sets controls for protection purposes, interrupt handling, timer control etc.
Logical shift	There are two logical shift instructions logical shift left (LShiftL) and logical shift right (LShiftR). These two instructions shift an operand by a number of the positions specified in a count operand. The vacant positions created within the register due to shift operation are filled with zeroes.

Arithmetic shift	Arithmetic shift right operation repeats the sign bit as the fill-in bit for the vacant position. Arithmetic shift left operation is same as logical shift left operation.
Rotate	In shift instructions, the bits shifted out of the operand are lost, except for last bit shifted out which is retained in the carry flag C. The rotate left (right) instructions, preserve all bits. They move the bits that are shifted out of one end of the operand back into the other end.

Q.19 State the use of translate and convert operations.

Ans. : Conversion operations :

Operation	Description
Translate	Translates values in a section of memory based on a table of correspondences.
Convert	Converts the contents of a word from one form to another (e.g. packed decimal to binary).

Q.20 Explain any two I/O operations.

Ans. : I/O operations :

Operation	Description
Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register).
Output (write)	Transfer data from specified source to I/O port or device.
Start I/O	Transfer instructions to I/O processor to initiate I/O operation.
Test I/O	Transfer status information from I/O system to specified destination.

Q.21 Explain any two transfer of control operations.

Ans. : Transfer control operations :

Operation	Description
Jump (branch)	Unconditional transfer; loads PC with specified address.
Jump conditional	Tests specified condition. If condition is true, loads PC with specified address; otherwise, do nothing.
Call to subroutine	Places current programs return address on stack and jump to specified address.
Return	Load return address from stack into PC.
Execute	Fetches operand from specified location and executes as instruction : do not modify PC.
Skip	Increments PC to skip next instruction.
Skip conditional	Tests specified condition. If condition is true, skip the next instruction; otherwise, do nothing.
Halt	Stops program execution..
Wait (hold)	Stops program execution; test specified condition repeatedly; resume execution when condition is satisfied.
No operation	No operation is performed, but program execution is continued.

11.6 : Addressing Modes

Q.22 Define addressing mode ? ☞ **[SPPU : May-06, Marks 2]**

Ans. : Part of the programming flexibility for each processor is the number and different kind of ways the programmer can refer to data stored in the memory or I/O device. The different ways that a processor can access data are referred to as **addressing schemes** or **addressing modes.**

Q.23 Define effective address.

Ans. : An address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction is known as **Effective Address (EA).** An effective address can be made up from as many as three elements : The **base, index and displacement.**

Q.24 List different addressing modes and explain any two with suitable diagrams and examples.

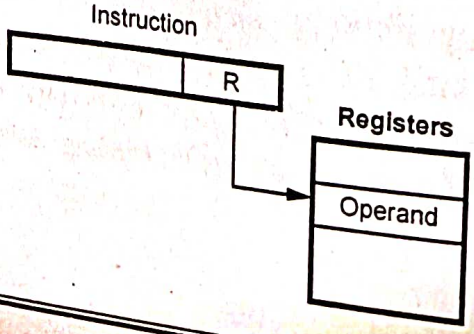
☞ [SPPU : May-06,08,17,19, Dec.-05,16,17,19, Marks 6]

Ans. : Types of addressing modes are :

1. Register addressing mode.
2. Absolute or direct addressing mode.
3. Immediate addressing mode.
4. Indirect addressing mode.
5. Register indirect addressing mode.
6. Displacement addressing mode.
7. Relative addressing mode.
8. Base register addressing.
9. Index addressing mode.
10. Auto-increment addressing mode.
11. Auto-decrement addressing mode.
12. Stack addressing mode.

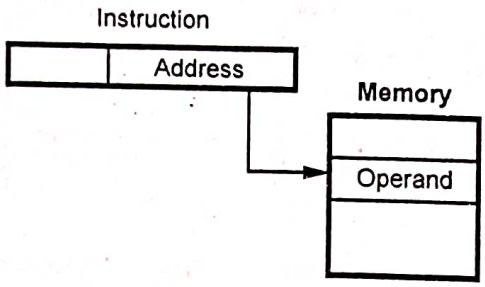
1. Register addressing mode : The operand is the contents of processor register. The name of register is specified in the instruction.

- **Example** : MOV R1, R2 : This instruction copies the contents of register R2 to register R1.



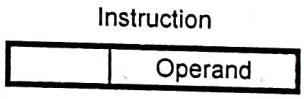
2. **Absolute or direct addressing mode** : The address of the location of the operand is given explicitly as a part of the instruction.

   - **Example** : MOV A, 2000 : This instruction copies the contents of memory location 2000 into the A register. As shown in the instruction, here, address of operand is given explicitly in the instruction.

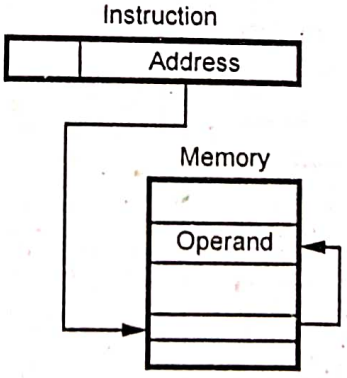


3. **Immediate addressing mode** : The operand is given explicitly in the instruction.

   - **Example** : MOV A, #20 : This instruction copies operand 20 in the register A. The sign # in front of the value of an operand is used to indicate that this value is an immediate operand.

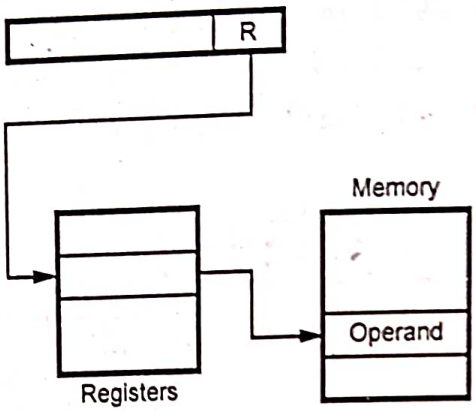


4. **Indirect addressing mode** : In this addressing mode, the instruction contains the address of memory which refers the address of the operand.



5. **Register indirect addressing mode** : The effective address of the operand is the contents of a register or the main memory location whose address is given explicitly in the instruction.

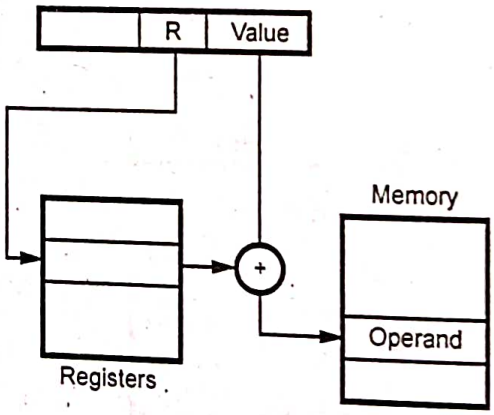
- **Example** : MOV A, (R0) : This instruction copies the contents of memory addressed by the contents of register R0 into the register A.



6. **Displacement addressing mode** : This addressing mode combines the capabilities of direct addressing and register indirect addressing. In this addressing mode, instruction has two address fields : **Value** and Referenced **register**. The effective address is computed by adding contents of referenced register to value.

   EA = Value + (R). Three common variation of displacement addressing are :

   - Relative addressing
   - Base register addressing
   - Index addressing.



7. **Relative addressing mode** : Here, the referenced register is Program Counter (PC) and hence this addressing mode is also known as **PC-relative addressing.**

   The effective address is determined by adding the contents of PC to the address field. EA = PC + Address part of instruction.

The address part is a signed number so that it is possible to have branch target location either before or after the branch instruction. This addressing mode is commonly used to specify the target address in branch instructions.

- **Example** : JNZ BACK : This instruction causes program execution to go to the branch target location identified by the name BACK, if the branch condition is satisfied.

8. **Base register addressing** : In this addressing mode, the referenced register contains the main memory address and address field contains the displacement. Displacement is usually unsigned integer number. EA = (R) + Displacement.

   - **Example** : MOV A, [R+8] : This instruction copies the contents of memory whose address is determined by adding the contents of register R and displacement 8 to the register A.

9. **Index addressing mode** : In this addressing mode, the address field references the main memory and the referenced register contains a positive displacement from that address.
   EA = Memory address + (R).

   The indexing is a technique that allows programmer to point or refer the data (operand) stored in sequential memory locations one by one. It is an efficient mechanism for performing iterative operations.

   - **Example** : MOV R, [R1 + RI] : In this instruction main memory address is given by register R1 and the referenced register RI gives the positive displacement. The contents of the memory address generated by the addition of main memory address and displacement is copied to register R.

10. **Autoincrement addressing mode** : The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are incremented to address the next location.

    - **Example** : MOV (R2) +, R0 : The above instruction copies the contents of register R0 into the memory location whose address is specified by the contents of register R2. After copy operation, the contents of register R2 are automatically incremented by 1.

11. Autodecrement addressing mode : The contents of a register specified in the instruction are decremented and then they are used as an effective address to access a memory location.

- **Example** : MOV R1, – (R0) : This instruction, initially decrements the contents of register R0 and then the decremented contents of register R0 are used to address the memory location. Finally, the contents from the addressed memory location are copied into the register R1.

12. Stack addressing mode : A stack is linear array of reserved memory locations. It is associated with a pointer called **Stack Pointer** (SP).

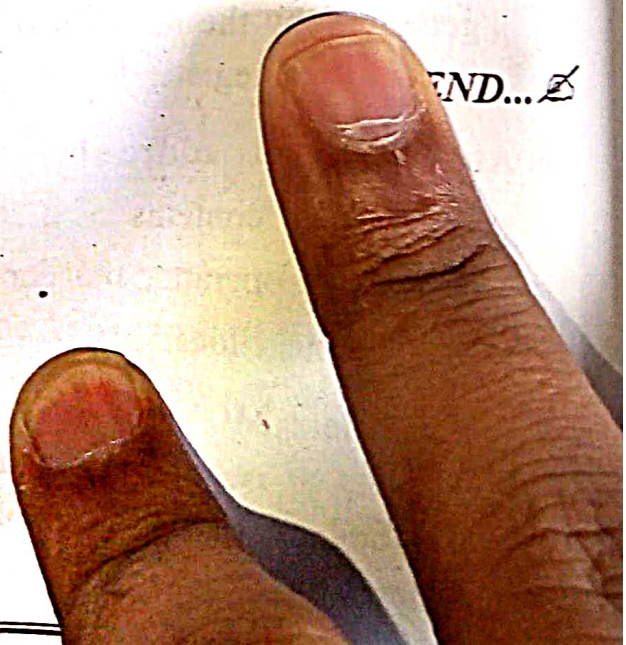
In stack addressing mode, stack pointer always contains the address of Top Of Stack (TOS) where the operand is to be stored or located. Thus, the address of the operand (source or destination) is the contents of stack pointer.

This addressing mode is the special case of register indirect addressing where referenced register is a stack pointer.

Usually, stack grows in the direction of descending addresses, (descending stack), starting from a high address and progressing to lower one. In this stack, SP is decremented before any items are appended (pushed) on stack and SP is incremented after any items popped from the stack.

- **Example :** PUSH R : This instruction decrements SP and copies the contents of register R on to the top of stack pointed by stack pointer.

END...



Unit 5

12 Processor Enchancements

12.1 : Key RISC and CISC Characteristics

Q.1 Write a note on CISC architecture.

OR Explain the key characteristics of CISC architecture.

Ans. : • CISC is an acronym for Complex Instruction Set computers or computing. It is based on the concept of using very large instruction set having simple as well as complex instructions and making instruction set more flexible to keep program length as small as possible.

- In CISC complex instructions may take multiple cycles for execution.
- CISC instructions vary in size, often specify a sequence of operations, and can require serial (slow) decoding algorithms.
- They tend to have few registers, and the registers may be special purpose, which restrict the way in which they can be used.
- In CISC, there are many instructions that support memory reference. For example, we can add memory contents to the registers.

CISC instruction sets are designed to take advantage of microcode.

- To add the flexibility in the instruction set, they support more and complex addressing modes.
- Example of CISCs are : Intel X86, Motorola 68000 series, DEC VAX, etc.

Q.2 Write a short note on RISC architecture.

OR Explain the key architecture of RISC architecture.

Ans. : RISC refers to Reduced Instruction Set Computers or Computing. RISC microprocessors are very different from CISC microprocessors. RISC use concept of keeping the instruction set as simple as possible to

allow the microprocessor's program to be written using only simple instructions.

- In RISC processors, there is an **one instruction per machine cycle.**
- RISC machine **instructions are not complicated** and can execute about as fast as, microinstruction on CISC machines.
- The machine **instructions are hardwired.** These instructions are executed faster than the instructions implemented with microinstructions.
- This architecture encourages the optimization of register use, so that frequently accessed operands remain in high-speed storage to implement register to register operations. For this RISC processors provide **multiple sets of registers.**
- RISC processor provides **limited number of instructions,** which simplifies the design of control unit.
- RISC processor uses **simple addressing modes.** Almost all instructions use simple register addressing.
- RISC processors use **simple instruction formats with fixed instruction length.** The instruction length is aligned on word boundaries. Field locations, especially opcodes are fixed.

Because of this, they provide following benefits :

- With fixed fields, opcode decoding and register operands addressing can occur simultaneously.
- It simplifies the design of control unit.
- Instruction fetching is optimized since word-length units are fetched.
- To speed-up instruction execution, RISC uses instruction pipelining.
- Examples of RISCs are : ARM, ATMEL, 8051 family, AVR, MIPS, PIC etc.

.3 Differentiate between RISC and CISC processors.

☞ [SPPU : May-19, Marks 6]

Ans. :

No.	Characteristics	CISC	RISC
1.	Instruction size	Varies	Fixed
2.	Instruction length	1, 2, 3 or 4 bytes	4 bytes
3.	Number of Instruction	More	Less
4.	Instruction decoding	Serial (slow) to decode	Easy (quick) to decode
5.	Instruction sementics	Varies from simple to complex ; possibly many dependent operations per instruction	Almost always one simple operation
6.	Addressing Modes	Supports complex addressing modes.	Complex addressing modes are synthesized in software.
7.	Instruction execution speed	Slow (depend on complexity of instruction)	Medium
8.	Instruction execution	By microprogram. Complex instructions taking multiple cycles.	By hardware. Simple instructions taking one cycle.
9.	Registers	Few, may be special purpose	Many, general purpose
10.	Memory references	Combined with operations in many different types of instructions	Not combined with operations, i.e., load/store architecture
11.	Hardware	Complicated	Simple

12.	Hardware design focus	Take the advantage of microcoded implementations	Take the advantage of implementations with one pipeline and no microcode
13.	Memory access	Frequently	Rarely
14.	Instruction format	Field placement varies	Regular, consistent placement of fields
15.	Pipelined	Not pipelined or less pipelined.	Highly pipelined.
16.	Conditional jump	Conditional jump is usually based on status register bit.	Can be based on a bit anywhere in memory.
17.	Compiler	Simple	Complicated
18.	Examples	Intel X86, Motorola 68000 series.	ARM, 8051, ATMEL, AVR, etc.

Table Q.3.1 Comparison between RISC and CISC processors characteristics

12.2 : Interrupt and its Purpose

Q.4 What is the purpose of interrupt ?

Ans. : Sometimes it is necessary to have the computer automatically execute one of a collection of special routines whenever certain conditions exists within a program or the computer system e.g. It is necessary that computer system should give response to devices such as keyboard, sensor and other components when they request for service. The purpose of interrupt is to provide above functionality .

Q.5 Define interrupt and interrupt service routine.

Ans. : The interrupt provides an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine (Interrupt Service Routine) that will service the requesting device. Once this servicing is completed, the processor would resume exactly where it left off. The

event that causes the interruption is called **Interrupt** and the special routine executed to service the interrupt is called **Interrupt Service Routine (ISR).**

Q.6 State the difference between ISR and IRET.

Ans. : The interrupt service routine is different from subroutine because the address of ISR is predefined or it is available in **Interrupt Vector Table (IVT)**, whereas subroutine address is necessarily to be given in subroutine CALL instruction. IRET instruction is used to return from the ISR whereas RET instruction is used to return from subroutine. IRET instruction restores flag contents along with CS and IP in the IA-32 architecture; however RET instruction only restores CS and IP contents.

12.3 : Types of Interrupts

Q.7 What are hardware and software interrupts ?

Ans. : • There are two main classes of interrupts :

- Hardware interrupts
- Software interrupts

• An interrupt caused by an external signal is referred as **hardware interrupt.**

• Conditional interrupts or interrupts caused by special instructions are called **software interrupts.**

Q.8 What are vector interrupts ?

OR What is vectoring ?

Ans. : • If the internal control circuit of the processor produces a CALL to a predetermined memory location which is the starting address of interrupt service routine, then that address is called vector address and such interrupts are called **vector interrupts.**

• For vector interrupts fastest and most flexible response is obtained since such an interrupt causes a direct hardware-implemented transition to the correct interrupt-handling program. This technique is called **vectoring.** When processor is interrupted, it reads the vector address and loads it into the PC.

Q.9 How does processor respond to non-vector interrupt ?

Ans. : In case of **non-vectored interrupt**, when the processor receives the interrupt signal from the external device, the processor completes the execution of the current instruction and sends a signal called INTA (interrupt acknowledge). After receiving the INTA signal, external hardware sends the interrupt vector to the processor. After receiving interrupt vector address, the processor loads it into the PC.

Q.10 Define maskable and non-maskable interrupts.

Ans. : • Most of the processors provide the masking facility. In the processor those interrupts which can be masked under software control are called **maskable interrupts**. The interrupts which can not be masked under software control are called **non-maskable interrupts**.

- Maskable interrupts are enabled and disabled under program control. By setting or resetting particular flip-flops in the processor, interrupts can be masked or unmasked, respectively.
- When masked, processor does not respond to the interrupt even though the interrupt is activated.

Q.11 Define nested interrupts.

Ans. : For some devices, a long delay in responding to an interrupt request may cause error in the operation of computer. Such interrupts are acknowledged and serviced even though processor is executing an interrupt service routine for another device. A system of interrupts that allows an interrupt service routine to be interrupted is known as **nested interrupts**.

12.4 : Interrupt Handling

Q.12 What do you mean by single level interrupts and multi level interrupts ?

Ans. : • Interrupts can be classified according to how the multiple interrupts are handled by the system.

- Single level interrupts
- Multi level interrupts

Single Level Interrupts

- In a single level interrupts there can be many interrupting devices. But all interrupt requests are made via a single input pin of the CPU.
- When interrupted, CPU has to poll the I/O ports to identify the request device. Polling software routine that checks the logic state of each device. Once the interrupting I/O port is identified, the CPU will service it and then return to task it was performing before the interrupt.
- Fig. Q.12.1 shows single level interrupt system, in which the interrupt requests from all the devices are logically ORed and connected to the interrupt input of the processor. Thus, the interrupt request from any device is routed to the processor interrupt input.

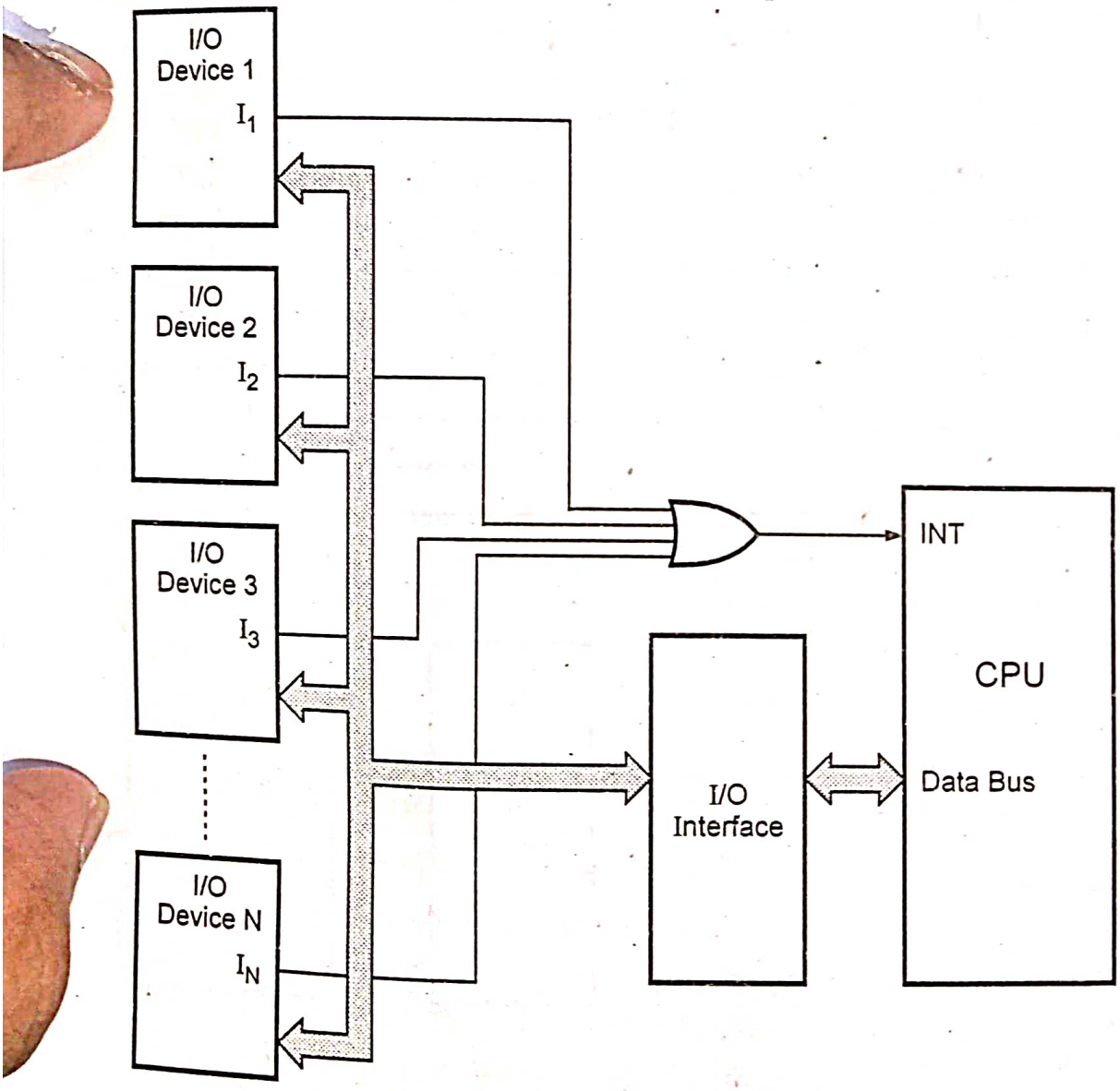


Fig. Q.12.1 Single level interrupt system

- After getting interrupted, processor identifies requesting device by reading interrupt status of each device.

Multi Level Interrupts

- In multi level interrupts, processor has more than one interrupt pins. The I/O devices are tied to the individual interrupt pins. Thus, the interrupts can be immediately identified by the CPU upon receiving an interrupt request from it. This allows processor to go directly to that I/O device and service it without having to poll first. This obviously saves the time in processing interrupt.
- Fig. Q.12.2 shows the multi level interrupt system.

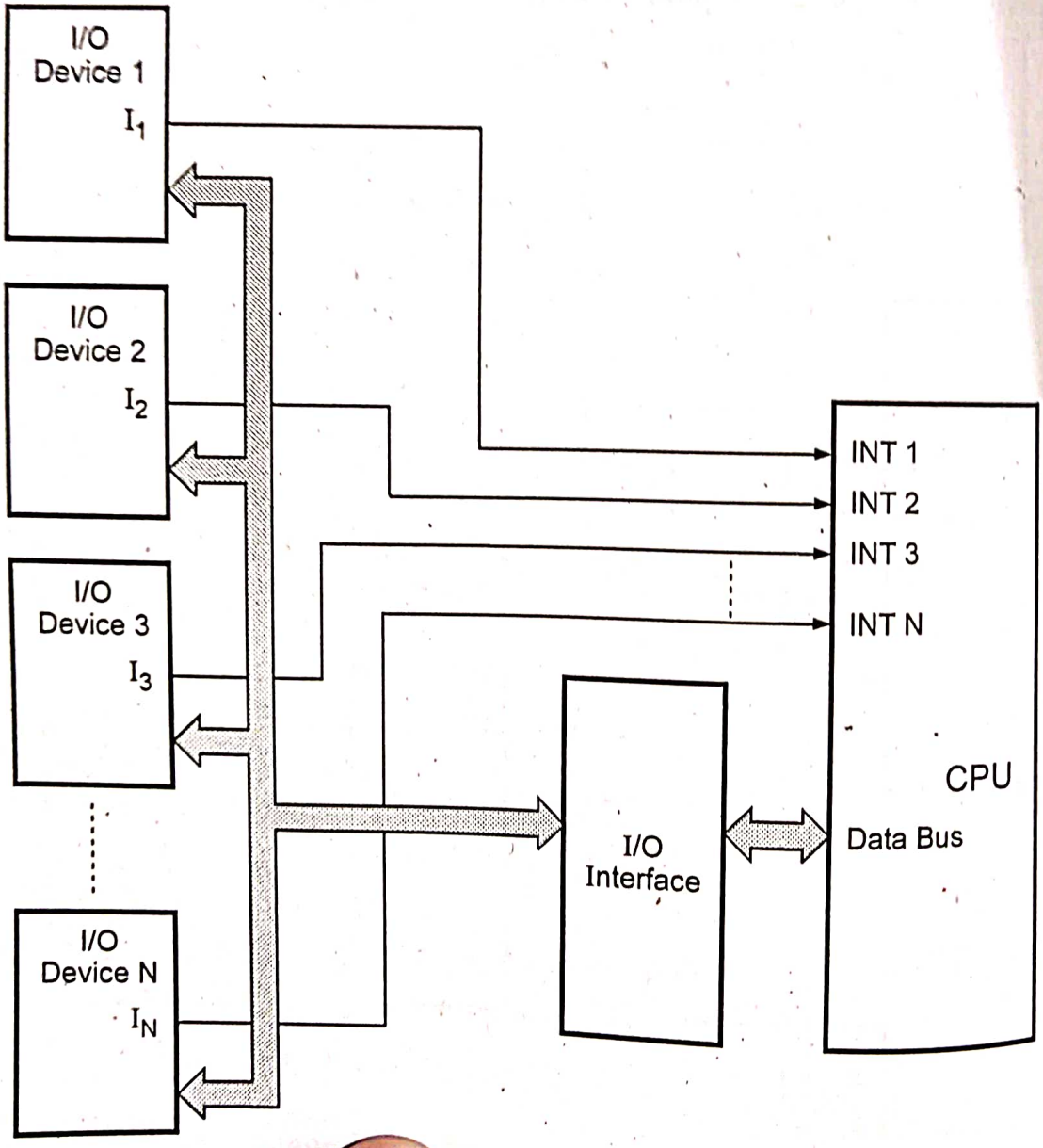


Fig. Q.12.2 Multilevel interrupt system

- In multi level interrupt system, when a processor is interrupted, it stops executing its current program and calls a special routine which "services" the interrupt.

Q.13 How does CPU recognize an interrupt ? What is the response of the CPU after recognition of the interrupt ?

☞ [SPPU : May-05, Marks 6]

Ans. : • When the external asynchronous input (interrupt input) is asserted (a signal is sent to the interrupt input), a special sequence in the control logic begins.

1. The processor completes its current instruction. No instruction is cut-off in the middle of its execution.
2. The program counter's current contents are stored on the stack. Remember, during the execution of an instruction the program counter is pointing to the memory location for the next instruction.
3. The program counter is loaded with the address of an interrupt service routine.
4. Program execution continues with the instruction taken from the memory location pointed by the new program counter contents.
5. The interrupt program continues to execute until a return instruction is executed.
6. After execution of the RET instruction processor gets the old address (the address of the next instruction from where the interrupt service routine was called.) of the program counter form the stack and puts it back into the program counter. This allows the interrupted program to continue executing at the instruction following the one where it was interrupted.

- Fig. Q.13.1 shows the response to an interrupt with the flowchart and diagram. (Refer Fig. Q.13.1 on next page)

Q.14 What is the use of interrupt priority ?

Ans. : • When interrupt requests arrive from two or more devices simultaneously, the processor has to decide which request should be serviced first and which one should be delayed. The processor takes the decision with the help of interrupt priorities.

- It accepts the request having the highest priority.

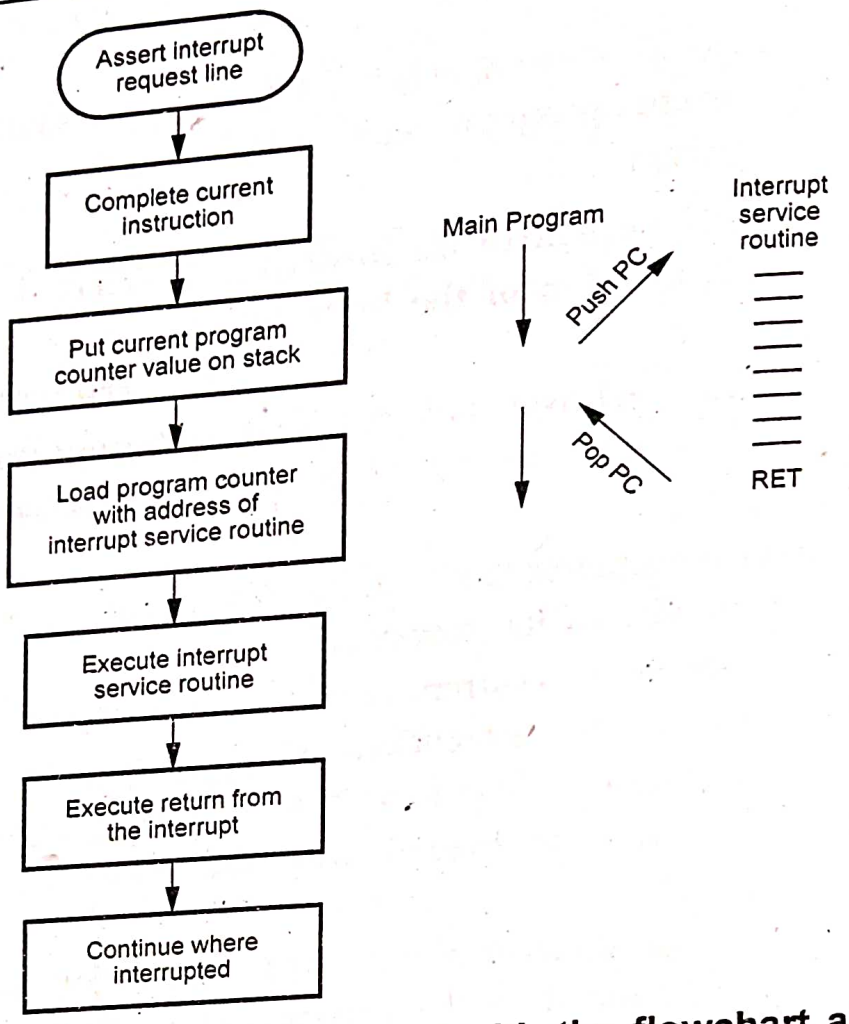


Fig. Q.13.1 Response to an interrupt with the flowchart and diagram

Q.15 What is pending interrupt ?

Ans. : The interrupt which took place but cannot be executed immediately (for instance, if another higher-priority interrupt is running), is **called pending interrupt.**

12.5 : Exceptions

Q.16 Write a note on exceptions.

Ans. : • An interrupt is an event that suspends the processing of currently executing program and begins the execution of another program. So far we have seen the interruption is activated by an I/O device in the form of a hardware signal; however, many events can cause an interrupts. All such events are called **exceptions.**

- Therefore, an I/O interrupt is a subtype of exception. Apart from I/O interrupts the exceptions can be classified as **faults, traps, or aborts**

depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported.

- **Faults :** Faults are the exceptions that are detected and serviced before the execution of the faulting instruction. For example, in virtual memory system if page or segment referenced by processor is not present, the operating system fetches the page or segment from disk using fault exception routine, and then processor restarts processing using referenced page or segment.
- **Traps :** Traps are exceptions that are reported immediately after the execution of the instructions which causes the problem. User defined interrupts are the examples of traps.
- **Aborts :** Aborts are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as hardware error, or illegal values in the system tables.

12.6 : Instruction Pipelining

Q.17 What is instruction pipeline ? Explain how it affects speed of execution ? ☞ [SPPU : May-13, Dec-15, Marks 7]

OR What are the advantages of pipelining ?

☞ [SPPU : May-16, Marks 3]

Ans. : Various cycles involved in the instruction cycle. These fetch, decode and execute cycles for several instructions are performed simultaneously to reduce overall processing time. This process is referred to as **instruction pipelining.**

Let us assume that instruction execution is divided into following five stages :

- S_1 - **Fetch Instruction** (FI) : Read the next instruction into an instruction buffer
- S_2 - **Decode Instruction** (DI) : Decode the opcode
- S_3 - **Fetch Operands** (FO) : Fetch each memory referenced operand.
- S_4 - **Execute Instruction** (EI) : Perform the indicated operation.

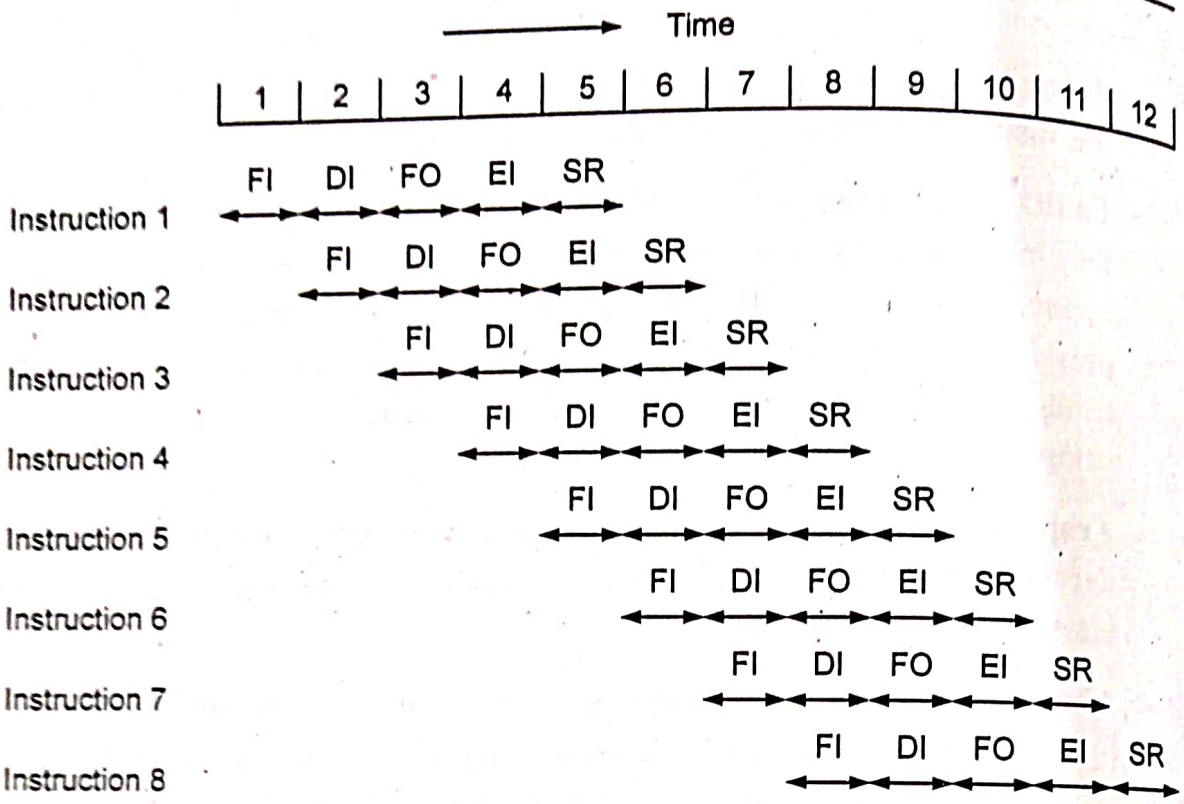


Fig. Q.17.1 Timing diagram for instruction pipeline operation

- **S_5 - Store Results** (SR) : Store the result in memory.
- With this subdivision and assuming equal duration for each stage we can reduce the execution time for 8 instructions from 40 time units to 12 time units. This is illustrated in Fig. Q.17.1.

Q.18 Write a note on pipeline hazards.

Ans. : 1. Structural hazards : These hazards are because of conflicts due to insufficient resources when even with all possible combination, it may not be possible to overlap the operation.

2. **Data or data dependent hazards :** These result when instruction in the pipeline depends on the result of previous instructions which are still in pipeline and not completed.

3. **Instruction or control hazards :** They arise while pipelining branch and other instructions that change the contents of program counter. The simplest way to handle these hazards is to stall the pipeline. Stalling of the pipeline allows few instructions to proceed to completion while stopping the execution of those which results in hazards.

Q.19 Explain with example working of multistage pipelining ?

☞ [SPPU : May-16, Marks 3]

Ans. : • The Fig. Q.19.1 shows the implementation of four-stage instruction pipelining. As shown in the Fig. Q.19.1 the CPU is directly connected to a cache memory, which is split into instruction and data parts, called the **I-cache** and **D-cache**, respectively. This splitting of the cache permits both an instruction word and a memory data word to be accessed in the same clock cycle.

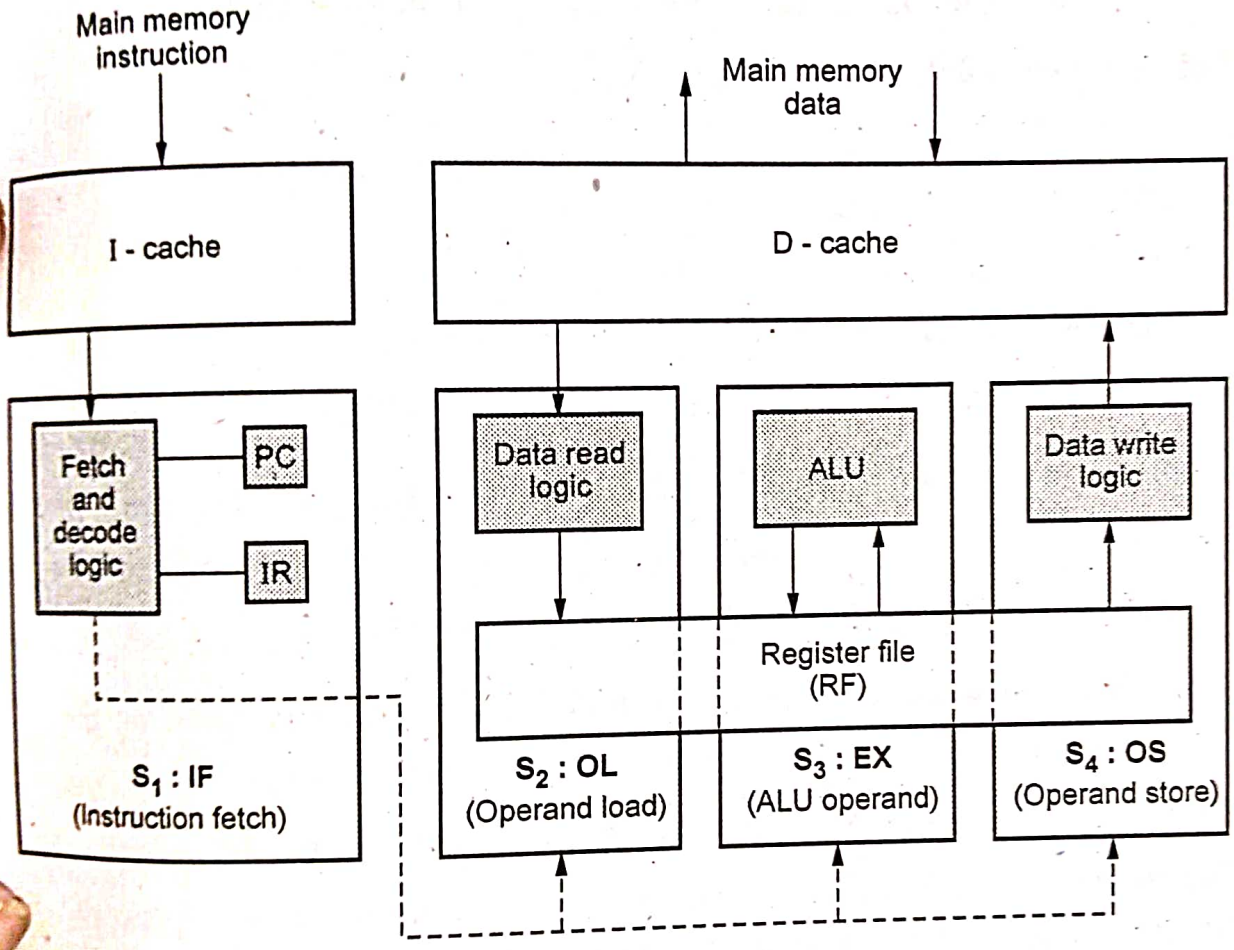


Fig. Q.19.1 Organization of CPU for four-stage instruction pipelining

- The four stages S_1 : S_4 shown in Fig. Q.19.1 perform the following functions :
  - **S_1 : IF** : Instruction fetching and decoding using the I cache.
  - **S_2 : OL** : Operand loading from the D-cache to register file.
  - **S_3 : EX** : Data processing using the ALU and register file.
  - **S_4 : OS** : Operand storing to the D-cache from register file.

- Stages S_2 and S_4 implements memory load and store operations, respectively.
- Stages S_2, S_3 and S_4 share the CPU's local Register File (RF). The registers in the register file act as interstage buffer registers.
- The stage 3 implements data transfer and data processing operations of the register to register type using ALU of the CPU.

Q.20 Write a note on branch prediction techniques.

Ans. : • Prediction techniques can be used to check whether a branch will be valid or not valid. These techniques reduce the **branch penalty.**

- The common prediction techniques are :
  - Predict Never Taken
  - Predict Always Taken
  - Predict By Opcode
  - Taken/Not Taken Switch
  - Branch History Table
- In the first two approaches if prediction is wrong a page fault or protection violation error occurs. The processor then halts prefetching and fetches the instruction from the desired address.
- In the third prediction technique, the prediction decision is based on the opcode of the branch instruction. The processor assumes that the branch will be taken from certain branch opcodes and not for others.
- The fourth and fifth prediction techniques are dynamic; they depend on the execution history of the previously executed conditional branch instruction.

Q.21 Explain various performance measures and instruction pipeline.

Ans. : • A pipeline's performance can be measured by its throughput in terms of millions of instructions executed per second or **MIPS.**

- Another popular measure of performance is the number of clock cycles per instruction or **CPI.** These quantities are related by the equation.

$$CPI = f/MIPS$$

where f is the pipeline's clock frequency in MHz, and the values of CPI and MIPS are average figures that can be determined experimentally by processing number of representative programs.

- Another general measure of pipeline performance is the **speedup S(m)** defined by

$$S(m) = \frac{T(1)}{T(m)}$$

where T(m) is the execution time for some target program on an m-stage pipeline and T(1) is the execution time for the same program on a similar, nonpipelined processor.

Q.22 Define stage delay and interstage delay.

Ans. : Each pipeline stage is a combinational logic circuit. It requires a specific amount of time in terms of propagation delay to process the input data. The precessing time in each stage is known as **stage delay**. It is denoted as τ. The delays are also introduced due to interstage transfer data. These time delays are known as **interstage delay** and it is denoted as d.

Q.23 How does the time periods for the clock cycle determined ?

Ans. : To determine the time period for the clock cycle it is necessary to consider stage delay and interstage delay. The interstage delays can be same between the stages. However, stage delays may not be same for different stages. In such cases, maximum stage delay, denoted as τ_m is considered to determine the time period for the clock cycle. The time period for clock cycle, τ can be given as

$$\tau = \max_i \{\tau_i\}_1^m + d = \tau_m + d$$

Usually, $\tau_m >> d$ and maximum stage delay dominates the clock period. Therefore, **pipeline frequency** which is inverse of clock period can be given as

$$f = \frac{1}{\tau}$$

Q.24 What is clock skewing ?

Ans. : Ideally, we expect the clock pulse to arrive at all stage latches at the same time. In practice, the same clock pulse may arrive at different

stages with a time offset of s. This problem is known as clock skewing. The offset s is large for longer logic paths within the stages.

Q.25 Draw and explain the space-time diagram of four stage pipeline.

Ans. : Fig. Q.25.1 shows the space-time diagram of a four stage pipeline processor. As shown in the Fig. Q.25.1, once the pipe is filled up, it outputs one result per clock period independent of the number of stages in the pipe. Ideally, a linear pipeline with m stages can process n tasks in $T_m = m + (n - 1)$ clock cycles, where m cycles are needed to complete the execution of the first task and the remaining n – 1 tasks require n – 1 cycles. Thus the total time required is

$$T_m = [m + (n - 1)]\tau$$

where τ is the clock period.

- The space-time diagram shown in Fig. Q.25.1 has four stages and five tasks. Therefore, the ideal total time required is

$$T_m = [4 + (5 - 1)]\ \tau$$

$$= 8 \text{ clock cycles}$$

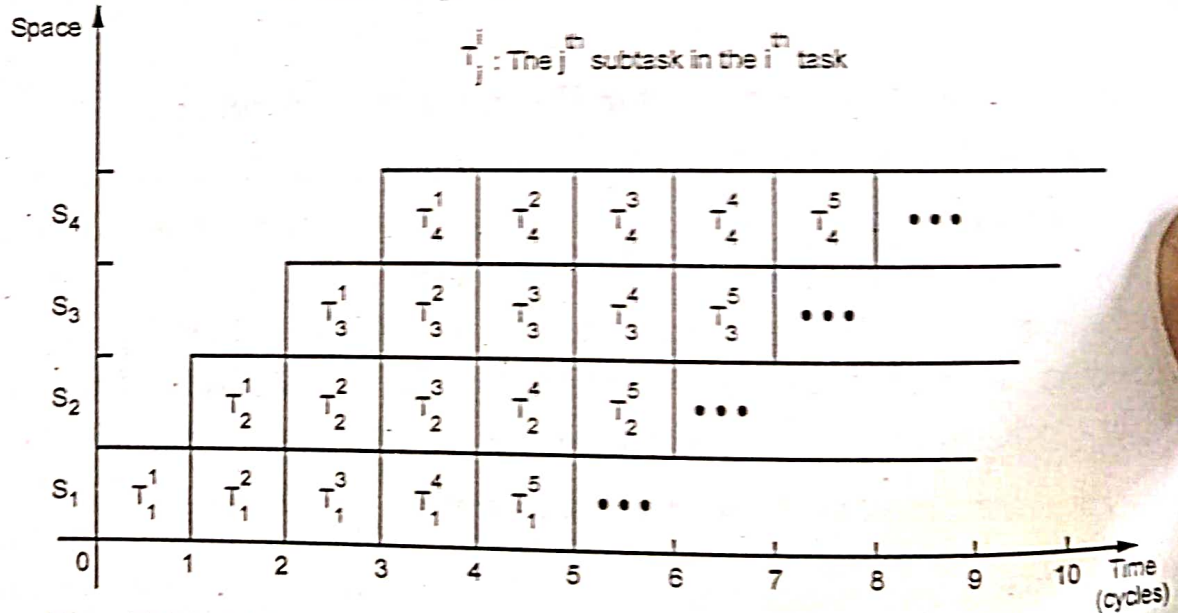


Fig. Q.25.1 The space-time diagram showing the overlapped operations

Q.26 What is speedup factor ? Explain with the help of example.

Ans. : The speedup factor of a m-stage pipeline over an equivalent non-pipelined processor is defined as

$$S_m = \frac{T_1}{T_m} = \frac{n\,m\tau}{[m+(n-1)]\tau} = \frac{n\,m}{m+(n-1)}$$

- For example, if pipeline has four stages and five tasks, its speedup factor is

$$S_4 = \frac{4\times 5}{4+(5-1)} = \frac{20}{8}$$

$$= 2.5$$

Q.27 Define efficiency of linear pipeline.

Ans. : The efficiency of a linear pipeline is defined as a ratio of speedup factor and the number of stages in the pipeline. The efficiency of m-stage pipeline is given as

$$E_m = \frac{S_m}{m} = \frac{nm}{m+(n-1)} \div m$$

$$= \frac{n}{m+(n-1)}$$

- For example, if speedup factor is 2.5 and number of stages are four, the efficiency can be given as

$$E_m = \frac{2.5}{4} = 0.625$$

- It can be noticed that the efficiency approaches to unity when $n \to \infty$. When n = 1, efficiency is minimum. It is 1/m.

Q.28 Define throughput of the pipeline.

Ans. : The pipeline throughput H_m is defined as the number of results (tasks) that can be completed by a pipeline per unit time. It is given as

$$H_m = \frac{n}{[m+(n-1)]\tau}$$

$$= \frac{E_m}{\tau} \qquad \because E_m = \frac{n}{[m+(n-1)]}$$

$$= E_m f \qquad \because f = 1/\tau$$

- In the ideal case, $H_m = 1/\tau = f$ when $E_m \to 1$. This means that the maximum throughput of a linear pipeline is equal to its frequency, which corresponds to one output result per clock period. The overall throughput of pipeline is always less than f. This is because usually $E_m < 1$.

Q.29 Define performance / cost ratio.

Ans. : The maximum throughput for m-stage pipeline can be given as

$$f = \frac{1}{\tau} = \frac{1}{(t/m+d)}$$

- The total pipeline cost is roughly estimated by c + mh, where c represents cost of all logic stages and h represents the cost of each latch. With this information, in 1973, Larson has defined a **pipeline performance/cost ratio** as

$$PCR = \frac{f}{c+mh} = \frac{1}{(t/m+d)(c+mh)}$$

12.7 : Multiprocessor Systems and Multicore Processor

Q.30 What is parallel processing ?

Ans. : • To fulfil increasing demands for higher performance it is necessary to process data concurrently to achieve better throughput instead of processing each instruction sequentially as in a conventional computer. Processing data concurrently is known as **parallel processing.**

Q.31 State the basic ways to achieve parallelism.

Ans. : There are two basic ways by which we can achieve parallelism. These are :

- **Multiple Functional Units :** System may have two or more ALUs so that they can execute two or more instructions at the same time.
- **Multiple Processors :** System may have two or more processors operating concurrently.

Q.32 Define multiprocessor system, task-level parallelism, parallel processing program and cluster.

Ans. : • **Multiprocessor system :** A computer system with at least two processors is called multiprocessor system.

- **Task-level parallelism or process-level parallelism :** Utilizing multiple processors for executing independent programs simultaneously is known as Task-level parallelism or process-level parallelism.

- **Parallel processing program :** It is referred to a single program that runs on multiple processors simultaneously.
- **Cluster :** A set of computers connected over a local area network that function as a single large multiprocessor is called cluster.

Q.33 Why Multicore ?

Ans. : • A multicore is an architecture design that places multiple processors on a single die (computer chip) to enhance performance and allow simultaneous processing of multiple tasks more efficiently. Each processor is called a core. The multicore architecture designs are known as **Chip Multiprocessors (CMPs)** because they allow single chip multiprocessing.

Q.34 Explain the limitations to increase clock frequency or processor speed to increase performance.

Ans. : • One way to increase system performance is to increase clock frequency or processor speed. However, this approach has following limitations in terms of cost effectiveness :

- Higher frequency requires more power.
- More power consumption results it harder and more expensive to cool the system.
- More power consumption also affects sizing and packaging considerations.

• Thus, instead of trying to make the processor faster to gain performance, adding more processors on a single chip (multicore architecture) is a better approach.

12.8 : Flynn's Taxonomy of Parallel Processor Architectures

Q.35 Write about Flynn's taxonomy for multiple processor organizations. ☞ [SPPU : Dec.-16, Marks 6]

OR What is Flynn's taxonomy for multiple processor organizations ? Explain with diagram. ☞ [SPPU : June-17, Marks 6]

Ans. : • The classification made by Micheal J. Flynn divides computers into four major groups.

- Single Instruction Stream-Single Data stream (SISD).
- Single Instruction Stream-Multiple Data streams (SIMD).
- Multiple Instruction Streams-Single Data stream (MISD).
- Multiple Instruction Streams-Multiple Data streams (MIMD).

1. **Single Instruction stream Single Data Stream (SISD) :** Most conventional machines with one CPU containing a single arithmetic-logic unit capable only of scalar arithmetic fall into this category. SISD computers and sequential computers are thus synonymous. In SISD computers instructions are executed sequentially but may overlap in their execution stages (Pipelining). They may have more than one functional unit, but all functional units are control by a single control unit.

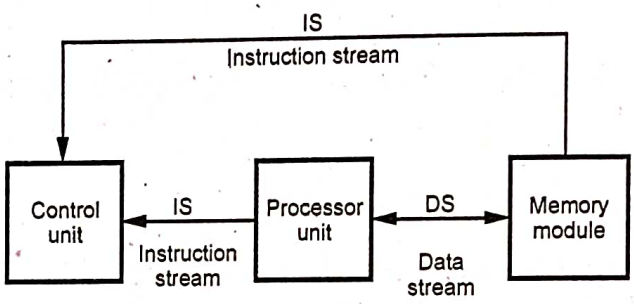


Fig. Q.35.1 SISD computer

2. **Single Instruction stream Multiple Data streams (SIMD) :** This category corresponds to array processors. They have multiple processing/execution units and one control unit. Therefore, all processing/execution units are supervised by the single control unit. Here, all processing elements received same instruction from control unit but operate on different data sets from distinct data streams. This category is also known as single program, multiple data stream (SPMD).

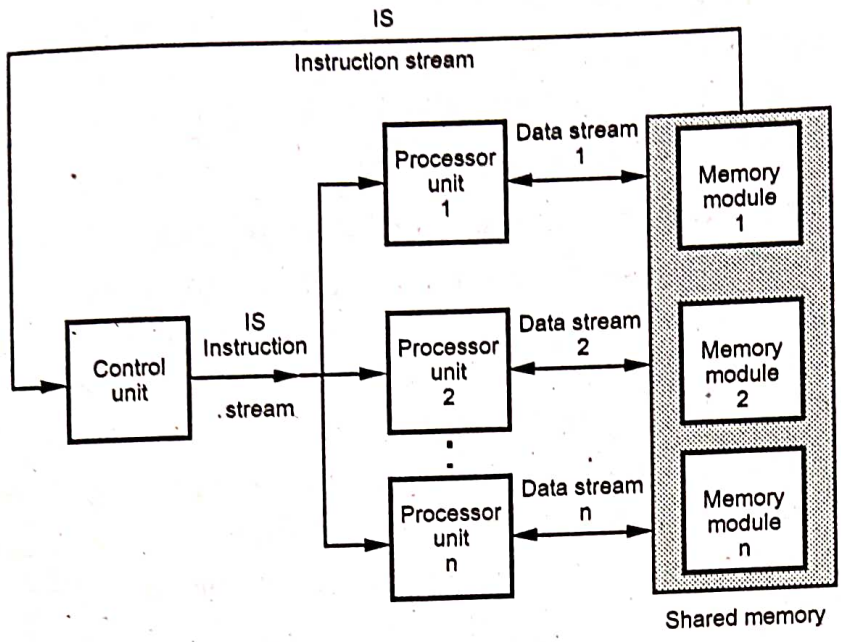


Fig. Q.35.2 SIMD computer

3. **Multiple Instruction streams Single Data stream (MISD) :** Not many parallel processors fit well into this category. In MISD, there are n processor units. Each receiving distinct instructions operating over the same data stream and its derivatives. The results of one processor

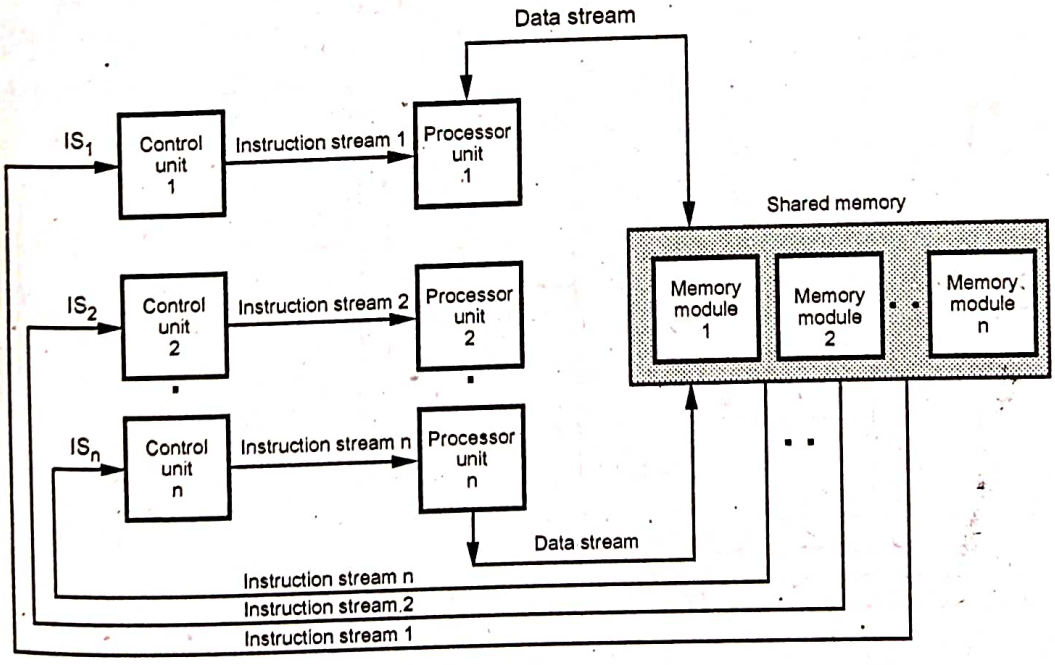


Fig. Q.35.3 MISD computer

become the input of the next processor in the micropipe. The fault-tolerant computers where several processing units process the same data using different programs belongs to the MISD class. The results of such apparently redundant computations can be compared and used to detect and eliminate faulty results.

4. **Multiple Instruction streams Multiple Data streams (MIMD) :** Most multiprocessors system and multiple computers system can be classified in this category. In MIMD, there are more than one processor unit having the ability to execute several program simultaneously. MIMD computer implies interactions among the multiple processors because all memory streams are derived from the same data space shared by all processors. If the n data streams are derived from disjointed sub spaces of the shared memories then we would have the so-called Multiple SISD (MSISD) operation.

5. **Vector systems :** A more efficient interpretation of SIMD is called a vector architecture. Rather than having 64 ALUs perform 64 additions simultaneously, like the old array processors, the vector architectures pipelined the ALU to get good performance at lower cost.

- Vector architecture consists of a set of vector registers.
- Vector architectures collect data elements from memory, put them in order into a large set of registers, operate on them sequentially in registers and then write the results back to memory.

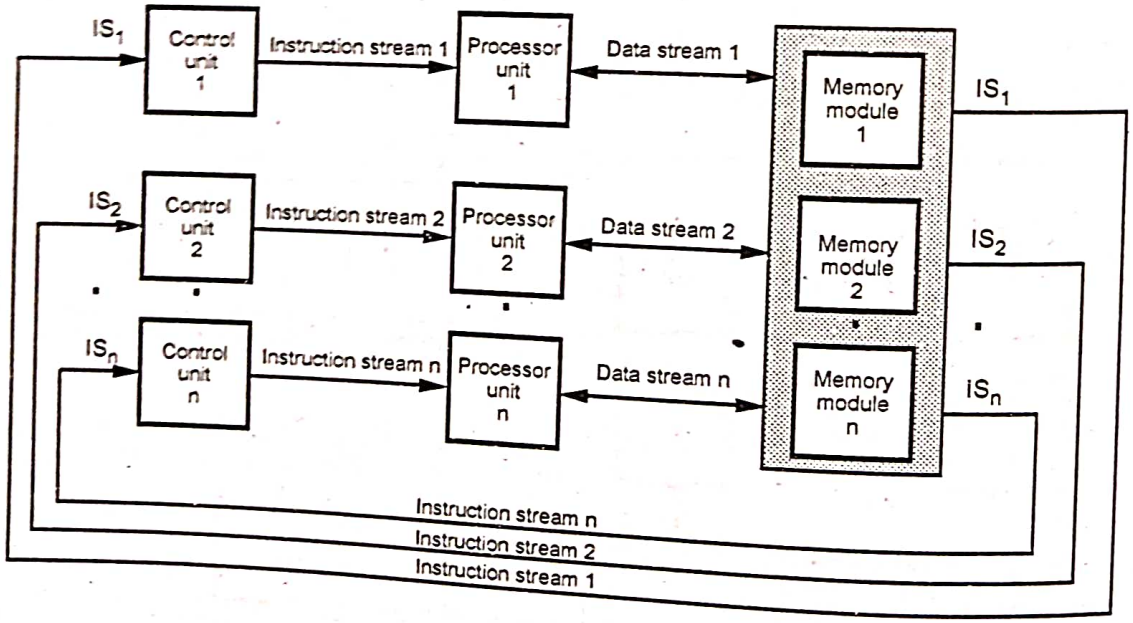


Fig. Q.35.4 MIMD computer

12.9 : Closely Coupled and Loosely Coupled Multiprocessor Systems

Q.36 What are the advantages of the multiprocessor systems ?

OR Define multiprocessing system.

OR State the types of multiprocessor system.

Ans. : Definition : If a microprocessor system contains two or more components that can execute instructions independently, then the system is called **multiprocessor system**. Multiprocessor system uses a distributed approach. Here, more than one processors are used to do the subtasks instead of doing entire task by a single processor.

Advantages : This system has following advantages :

1. Improves cost/performance ratio of the system.
2. Several processors may be combined to fit the needs of an application while avoiding the expense of the unneeded capabilities of a centralized system. Yet this system provides room for expansion.
3. Tasks are divided among the modules. If failure occurs, it is easier and cheaper to find and replace the malfunctioning processor than replacing the failing part of complex processor.

Types : The multiprocessor systems are implemented using one of the two basic architectures : **Loosely coupled architecture** and **closely coupled architecture**. The systems using these architectures are known as loosely coupled systems and closely coupled systems respectively.

Q.37 Explain closely coupled and loosely coupled microprocessor system. ☞ [SPPU : June-17, Dec.-17,18, May-19, Marks 7]

Ans. : Closely Coupled Multiprocessor System : • In the Closely Coupled System (CCS) the processors or supporting processors (coprocessor, maths processor) share clock generator, bus control logic, entire memory and I/O subsystem. Such systems communicate through a shared main memory. Hence the rate at which data can communicate from one processor to the other is on the order of the bandwidth of the memory. One of the limiting factors to the expansion of a CCS is the performance degradation due to memory contentions which occur when two or more processors attempt to access the same memory unit

concurrently. However, when high-speed or real-time processing is desired, Closely Coupled Systems (CCS) may be used.

Loosely Coupled Multiprocessor System : • In loosely coupled systems, each processor has a set of input-output devices and a large local memory where it accesses most of the instructions and data. The processor, its local memory and input-output interfaces are together called **computer module**. Processes which execute on different computer modules communicate by exchanging messages through a **Message Transfer System (MTS)**. The coupling in such a system is very loose. Hence, such systems are also referred to as a **distributed systems**. These systems are usually efficient when the interactions between tasks are minimal.

Q.38 State the advantages of loosely coupled system.

Ans. :

1. Better system throughput by having more than one processor.
2. Each processor may have a local bus to access local memory or I/O devices so that a greater degree of parallel processing can be achieved.
3. System structure is more flexible. As the system consists of different modules, one can easily add or remove modules to change the system configuration ; without affecting the other modules in the system.
4. A failure in one module normally does not cause a breakdown of the entire system. The faulty module can be detected and replaced.

Q.39 Give the comparison between closely coupled and loosely coupled systems.

Ans. :

Sr. No.	Closely Coupled System	Loosely Coupled System
1.	A multiprocessor system with common shared memory is known as closely coupled system.	A multiprocessor system in which each processor has its own private local memory is known as loosely coupled system.

2.	Here, the information can be shared among the CPUs by placing it in the common global memory.	Here, the information is transferred from one processor to other by message-passing system.
3.	Parallelism can be implemented less efficiently.	Parallelism can be implemented more efficiently.
4.	System structure is less flexible.	System structure is more flexible.

12.10 : Symmetric Multiprocessors (SMP)

Q.40 Explain Symmetric Multiprocessor (SMP) Organization with features.

☞ [SPPU : Dec.-16, Marks 7]

Ans. : • In computing, **symmetric multiprocessing** or **SMP** involves a multiprocessor computer-architecture where two or more identical processors can connect to a single shared main memory. Most common multiprocessor systems today use an SMP architecture. In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors.

Features / Characteristics of SMP :

1. Two or more similar processors are employed in a stand-alone system.
2. The processors share the same main memory and I/O facilities.
3. The processors access memory and I/O in approximately equal amount of time.
4. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
5. All processors are capable of performing the same functions.
6. The operating system controls the interaction between processors and their programs at different levels.
7. The distribution of workload between processors is performed by the operating system in such a way that multiple independent or dependent

tasks are shared between processors without any special consideration in the application program.

- Fig. Q.40.1 shows a block diagram of a typical SMP machine. There are two or more processors. Each processor unit consists of ALU, registers, control unit and typically one or more levels of cache. Each processor can access a shared main memory and the I/O devices through some form of interconnection mechanism. Many times, memory is organized to provide simultaneous accesses to separate blocks of memory. Apart from shared memory, each processor may have its own private main memory and I/O channels.

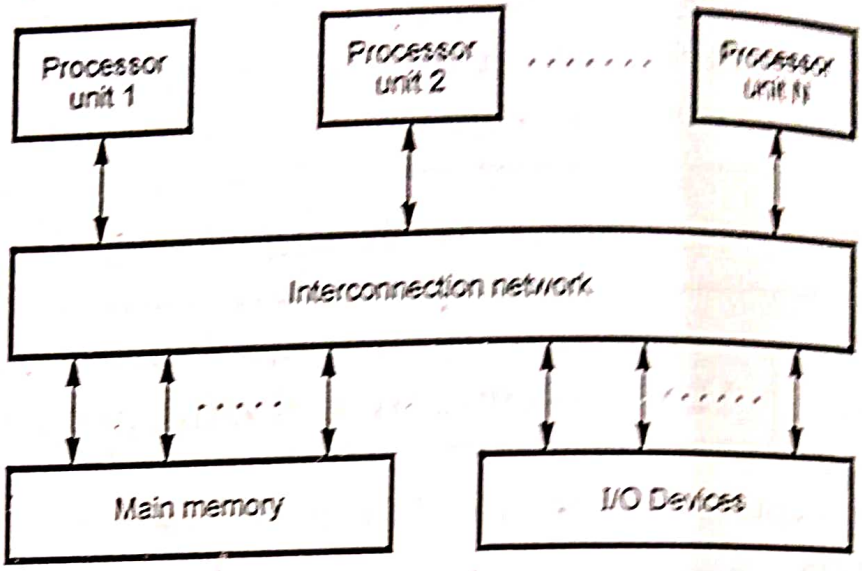


Fig. Q.40.1 Block diagram of typical SMP machine

Q.41 State the advantages and disadvantages of SMP.

Ans. : Advantages of SMP :

1. **Performance** : In situations where more than one program executes at the same time, an SMP system will have considerably better performance than a uni-processor because different programs can run on different processors simultaneously.
2. **Availability :** In a symmetric multiprocessor, all processors are capable of performing same functions. Thus, the failure of a single processor does not halt the machine; the task of failure processor is done by other processors in the system. But this action reduces the performance of the system.

3. **Incremental growth (scaling) :** The machine is scalable. This means that we can add additional processor to increase the performance or to maintain the performance level in case of increased data processing.

Disadvantages of SMP :

1. System programmers must build support for SMP into the operating system; otherwise, the additional processor remain idle and the system functions as a uniprocessor system.
2. In cases where an SMP environment processes many jobs, administrators often experience a loss of hardware efficiency. Software programs should be developed to schedule jobs so that the processor utilization reaches its maximum potential.

Q.42 List various interconnections mechanisms used in the SMP.

Ans. : According to interconnection mechanism used in the SMP, they are classified as

- Time-shared or common bus
- Multi-part memory
- Central control unit.

Q.43 Write a note on time shared bus.

Ans. : • Fig. Q.43.1 shows a block diagram of typical SMP machine with a common time shared bus. The common time-shared bus provides the simplest and most cost-effective way to interconnect the processors together. It also, however, limits the number of processors that can be used. Since all memory and I/O references pass through the shared bus, the performance of the system is limited by the maximum band-width of this bus. One way to reduce the shared bus bottle-neck is the extensive use of cache memory. Most SMP systems have at least two levels of cache memory as shown in Fig. Q.43.1. Typically level 1 cache is on the same chip as the processor and level 2 cache may or may not be internal to the processor.

• Using cache memory in a multiprocessor system introduces the problem of cache coherency : how to guarantee that when a data in a specific cache are altered, this change is reflected in other caches and in main memory.

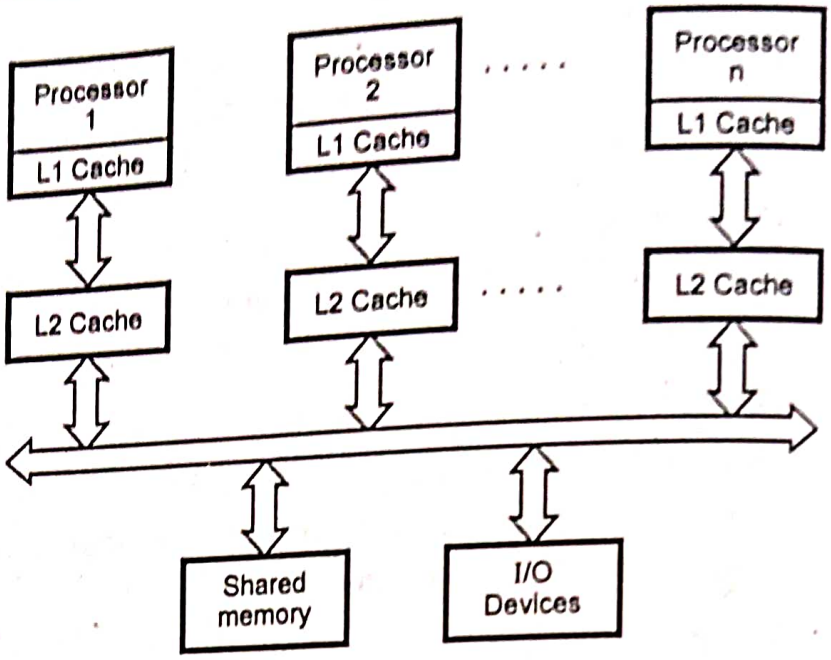


Fig. Q.43.1 SMP machine with a common time shared bus

Q.44 Explain the crossbar switch system organisation for multiprocessors.

Ans. : Fig. Q.44.1 shows the crossbar switch system organisation for multiprocessors which provides separate path for each memory module. The interconnection network shown in Fig. Q.44.1 is called **nonblocking crossbar.**

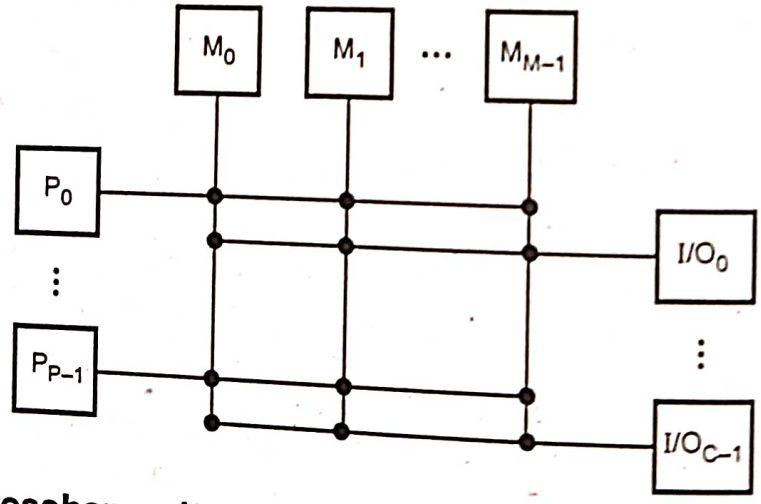


Fig. Q.44.1 Crossbar switch system organisation for multiprocessors

- The crossbar switch has a separate bus associated with each memory module. Therefore, the maximum number of transfers that can take place simultaneously is limited by the number of memory modules and the bandwidth-speed product of the buses rather than the number of paths available.

- Important characteristics of a system utilizing a crossbar matrix are :
  1. Extreme simplicity of the switch-to-functional unit interfaces.
  2. Ability to support simultaneous transfers for all memory units.
- To provide these features, it requires hardware capabilities which are capable of switching parallel transmissions and capable of resolving multiple requests for access to the same memory module occurring during a single memory cycle.

Q.45 Write a short note on multiport memory.

Ans. : • The multiport memory approach allows the direct, independent access of main memory modules by each processor and I/O module as shown in Fig. Q.45.1.

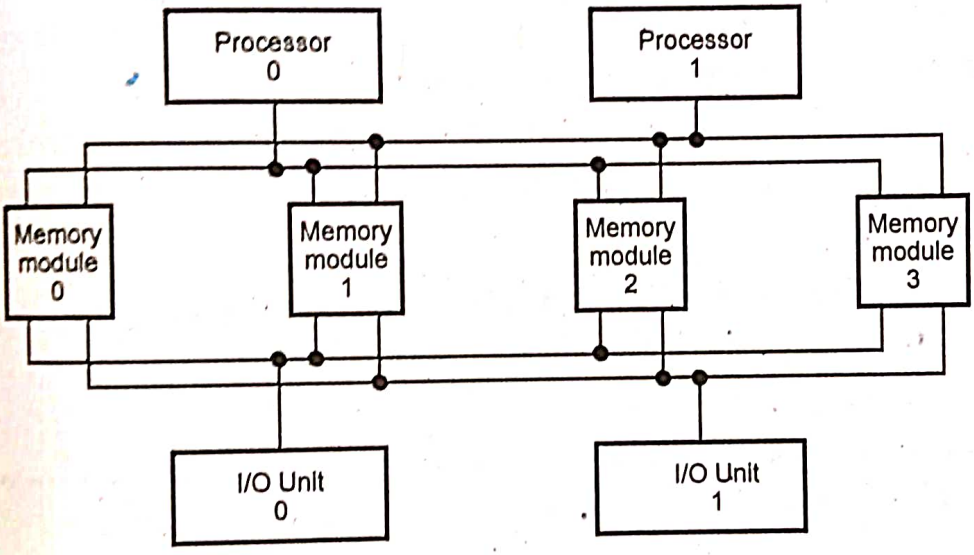


Fig. Q.45.1 Multiport memory without fixed priority assignment

- Here, memory-access conflicts are resolved by assigning permanently designated priorities at each memory port.
- In these organisations, it is possible to designate portions of memory as private to certain processors, input/output units or combinations thereof.
- In Fig. Q.45.2 memory modules 0 and 3 are private to processors 0 and 1 respectively. Such organisation has an advantage that due this there is increase in protection against unauthorized access and it also permit the storage of recovery routines in memory areas that are not susceptible to modifications by other processors.

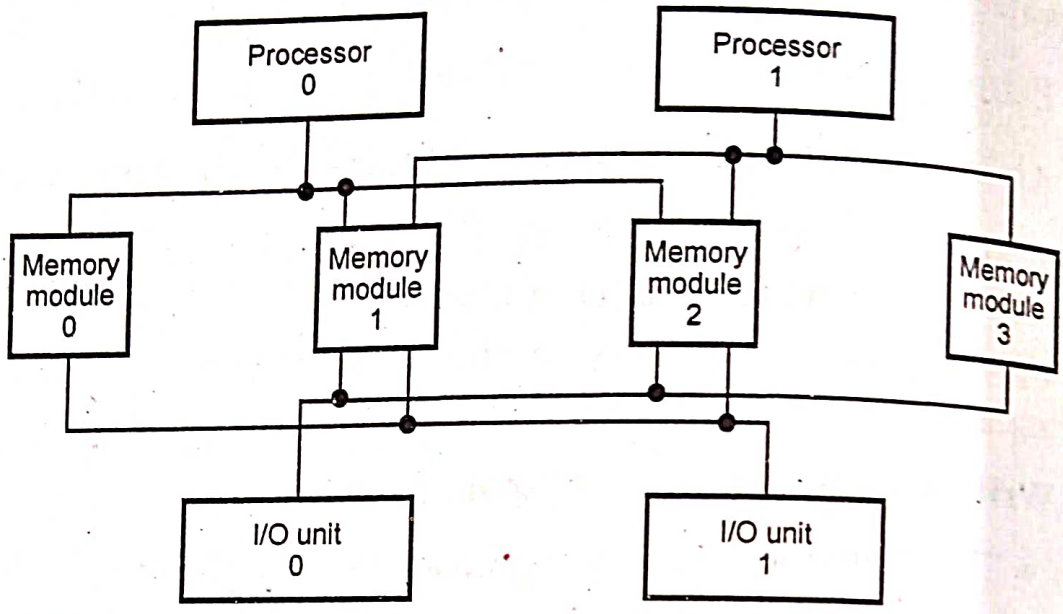


Fig. Q.45.2 Multiport memory with private memories

- However, the main disadvantage of such system is the system recovery if the other processors are not able to access control and status information in a memory block associated with a faulty processor. This organization can also support non blocking access to memory if a full-connected topology is used.

Q.46 Give comparison between time shared bus, crossbar switch and multiport memory organizations.

Ans. :

Sr. No.	Time shared bus	Crossbar switch	Multiport memory
1.	Lowest overall system cost for hardware and Least complex	Most complex. Highest total transfer rate.	Most expensive since most of control and switching circuitry is in the memory unit.
2.	Easy to modify hardware system configuration.	Functional units are simplest and cheapest.	Functional units permit low cost uniprocessor.

3.	Overall system capacity limited by bus transfer rate. Bus fail then whole system fail.	Usually cost-effective for multiprocessors only as basic switching matrix is required to assemble any functional units into a working configuration.	Potential for a very high total transfer rate in overall system.
4.	Expanding may degrade system performance.	System expansion improves overall performance.	Difficult to modify as the design decision is made quite early.
5.	Lowest efficiency.	Expansion of system is limited only by size of the switch matrix.	Large number of cables and connector are required.
6.	Organization is suitable for smaller systems only.	Reliability of system can be improved by segmentation or redundancy within switch.	

Table Q.46.1

12.11 : Multithreading

Q.47 What is multithreading ? ☞ [SPPU : May-18, Marks 2]

Ans. : • In multithreading, the instruction stream is divided into several smaller streams, called **threads**, such that the threads can be executed in parallel. Here, a high degree of **instruction-level parallism** can be achieved without increasing circuit complexity or power consumption.

Q.48 Define process, resource ownership, scheduling / execution process switch, thread and thread switch.

Ans. :

- **Process** : A process is an instance of a program running on a computer. The process image is the collection of program data, stack and attributes that define the process. The process image is stored at a virtual address space. Two important characteristics of a process are discussed below.

- **Resource ownership :** A process may get control of resources such as main memory, I/O channels, I/O devices and files from time to time.
- **Scheduling / execution :** A process execution takes places through one or more programs. This execution may interleaved with that of other processes. An operating system decides an execution state of each process such as running, ready, dispatching priority.

- **Process switch :** A process switch is an operation that switches the process or control from one process to another. It first saves all the process control data, registers and other information and then replaces them with the process information for the second.
- **Thread :** A thread includes the program counter, stack pointer and its own area for a stack. It executes sequentially and can be interrupted to transfer control to an another thread.
- **Thread switch :** A thread switch is an operation that switches the processor control from one thread to another within the same process. This is cheaper than a process switch.

Q.49 Give comparison of process switch and thread switch.

Sr. No.	Process switch	Thread switch
1.	It is an operation that switches the process or control from one process to another.	It is an operation that switches the processor control from one thread to another thread.
2.	When the processor control is transferred from one process to another, the control or ownership of resources is also transferred. So process switch is time consuming than thread switch.	The multiple threads within a process share the same resources. So a thread switch is much less time consuming than a process switch.
3.	It is much costly than a thread switch.	It is much less costly than process switch.

Q.50 What do you mean by implicit and explicit multithreading.

Ans. :

- User level threads which are visible to the application program and kernel-level threads which are visible only to operating system, both are referred to as explicit threads.
- Implicit multithreading refers to the concurrent execution of multiple threads extracted from a single sequential program.
- Explicit multithreading refers to the concurrent execution of instructions from different explicit threads, either by interleaving instructions from different threads on shared pipelines or by parallel execution on parallel pipelines.

Q.51 Discuss various approaches to explicit multithreading.

OR What is multicore ?

Ans. : Approaches to Explicit Multithreading :

- **Interleaved or fine-grained multithreading :** The processor executes two or more threads at a time. It switches from one thread to another at each clock cycle. During execution, if a thread is blocked because of data dependencies or memory latencies, that thread is skipped and a ready thread is executed.
- **Blocked or coarse-grained multithreading** : The processor executes instructions of a thread sequentially and if an event (e.g. cache miss) that causes any delay occurs, it switches to another thread.
- **Simultaneous multithreading (SMT)** : The wide superscalar instruction is executed by executing multiple threads simultaneously using multiple execution units of a superscalar processor.
- **Chip multiprocessing** : The processor is replicated on a single chip and each processor executes separate threads. This approach effectively utilizes the available logic data on a chip without increasing pipeline design complexity. This is referred to as **multicore**. Chip multiprocessing enables simultaneous execution of instructions from different threads.

12.12 : Clusters and Cluster Configurations

Q.52 What are clusters ?

OR What is cluster computing ? ☞ [SPPU : June-17, May-18,19, Marks 2]

Ans. : • An important and relatively recent development in computer system design is clustering. Clustering is an alternative to symmetric multiprocessing (SMP) as an approach to provide high performance and high availability and is particularly attractive for server applications. A **computer cluster** is a group of linked computers, working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Here, computer means a system that can run its own, a part from the cluster. Such a computer in a cluster is typically referred to as a **node.** Clusters are usually deployed to improve performance and/or availability provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Q.53 State the advantages of clustering ?

OR Explain benefits of clustering. ☞ [SPPU : Dec.-16, May-18, Marks 3]

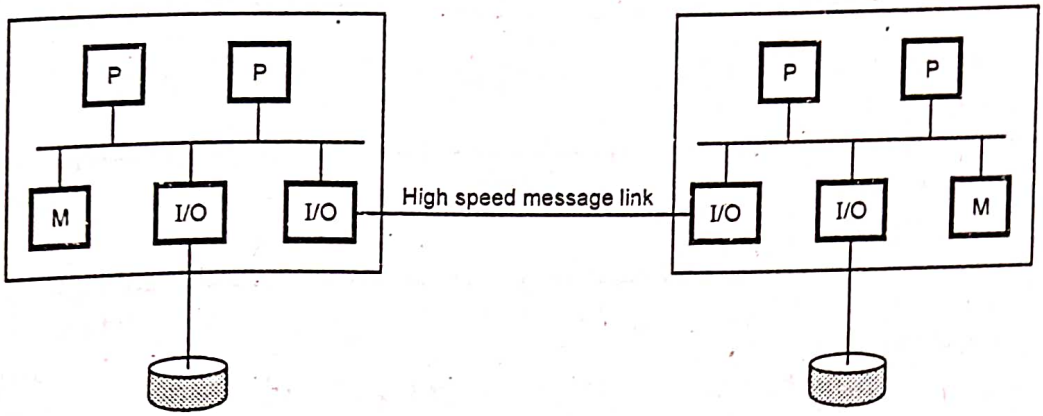
Ans. :

1. **Absolute scalability :** It is possible to have a cluster with dozens or even hundreds of machines and each machine can be a multiprocessor.
2. **Incremental scalability :** It is possible to upgrade existing machines in the cluster with modern machine or machines with higher performance.
3. **High availability :** Because each node in a cluster is a standalone computer, the failure of one node does not halt the service. Another node is assigned to perform the task of failure node.
4. **Cost effective :** Clusters are cost-effective than single computers of comparable speed or computing power.

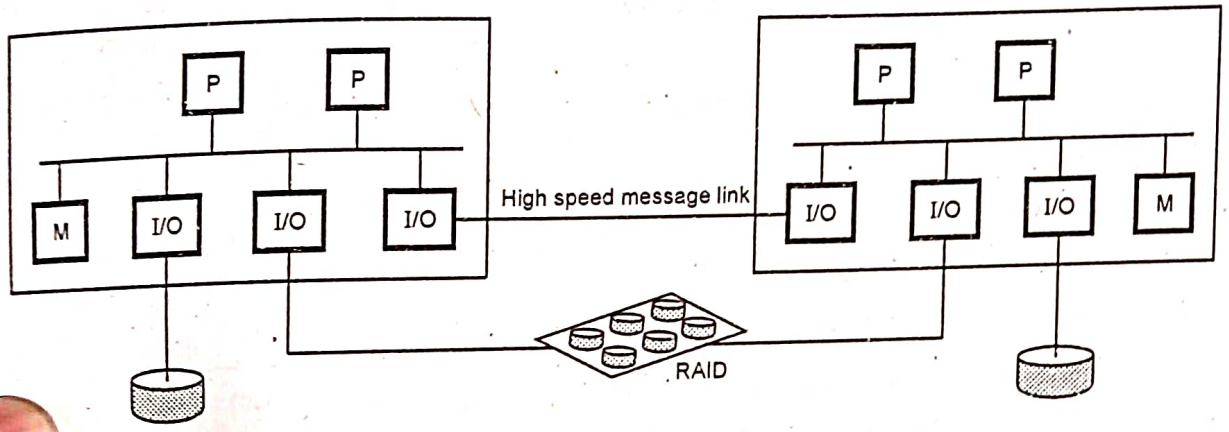
Q.54 Explain cluster configurations. ☞ [SPPU : Dec.-16,17,19, Marks 4]

OR What are the types of clustering ? ☞ [SPPU : June-17, Marks 4]

Ans. : • Clusters are classified in a number of different ways. The classification can be based on whether the computers in a cluster share access to the same disk, whether every single node in the cluster is



(a) Two-node cluster with no shared disk



(b) Two-node cluster with shared disk

Fig. Q.54.1

exactly same and so on. The Fig. Q.54.1 shows the classification of cluster according to shared access. The Fig. Q.54.1 (a) shows a two-node cluster with no shared memory between two nodes. The interconnection between two nodes is by means of high-speed link. On the other hand, Fig. Q.54.1 (b) shows a two-node cluster with shared disk.

Homogeneous clusters : • It is a cluster type in which every single node is exactly same, from mother and memory, to the disk drives and the network controller cards.

Heterogeneous clusters : • Heterogeneous clusters come in two general forms :

1. Made from different kinds of computers. For e.g. a few Sun SPARC station IPXs, a few Intel 486 machines, and a DEC Alpha.
2. Made from different machines in the same architecture family: For e.g. a collection of Intel boxes where the machines are of different generations (e.g. a mixture of 486, Pentium I and Pentium II).

Q.55 State various clustering methods, their benefits and limitations.

Ans. : • Table Q.55.1 shows various clustering methods, their benefits and limitations.

Sr. No.	Clustering method	Description	Benefit	Limitations
1.	Passive Standby	• Older method • Primary server handles all of the processing load while secondary server remains inactive. • Secondary server takes over in case of failure of primary server. • Since only one machine does processing work it is not referred to as a cluster.	It is easy to implement.	Effective cost is high since only one server is active at a time.
2.	Active standby or Active secondary	• Secondary server is also used for processing tasks. • They are classified as - Separate servers - Shared nothing - Shared memory	Effective cost is reduced because secondary servers are used for processing tasks.	Complexity is more.
3.	Active server with separate servers	• Separate servers with own disk. • There is no shared disk • Data is continuously copied among systems so that each system has access to the current data of the other systems. This makes it possible to take over the processing by other system in case of failure of any one system.	High performance. High availability.	High data exchange overhead.

4.	Active server with servers connected to disks	• The common disks are partitioned into volumes, and each volume is owned by single computer. • If one computer fails, the other computer takes the ownership of the volumes of the failed computer.	Reduces data exchange overhead.	Requires disk mirroring or RAID technology to compensate for risk of disk failure.
5.	Active server with shared disks	• Multiple computers share same disk at same time, so that each computer has access to all of the volumes on all of the disks.	Reduces data exchange overhead. Reduces risk of downtime caused by disk failure.	Required lock manager software.

Table Q.55.1 Clustering methods, their benefits and limitations

Q.56 State the operating system design issues in the clustered organization.

Ans. :

1. **Failure management :** Fail over recovery means that if one of the participants in the cluster is temporary or permanently unavailable, for instance because of a hardware crash, its functions are undertaken by another participant. As a result, requests are automatically redirected to a working server in the cluster and users and applications are not aware of what is happening, because the system continues to process requests properly.

   The function of switching an application and data resources over from a failed the participant to the working server in the cluster is referred to as **failover.** A related function is the restoration of application and data resources to the original system once the system is out of failure is referred to as **failback.**

   The operating system should support failover and failback functions.

2. **Load balancing :** The operating system should distribute the tasks properly among available computers in the cluster to achieve the optimum performance when a new computer is added to the cluster, the operating system should restructure the distribution of task.

3. **Parallelizing computation :** The operating system should support parallel computation. This can be achieved by following three approaches:

- **Parallelizing compiler :** A parallelizing compiler determines, at compile time which parts of an application can be executed in parallel. These are then separated and assigned to different computers in the cluster.
- **Parallelized application :** In this approach, the programmer writes the application which is suitable to run on a cluster and uses message passing to move data, as required, between cluster nodes. This places a high burden on the programmer but may be the best approach for exploiting clusters for some applications.
- **Parametric computing :** This approach is used if the essence of the application is an algorithm or program that must be executed a large number of times, each time with a different set of starting conditions or parameters.

Q.57 Draw and explain the typical cluster architecture.

☞ **[SPPU : May-19, Marks 4]**

Ans. : • Fig. Q.57.1 shows a typical cluster architectures. The Fig. Q.57.1 shows the group of nodes connected through a very high-speed network/switch.

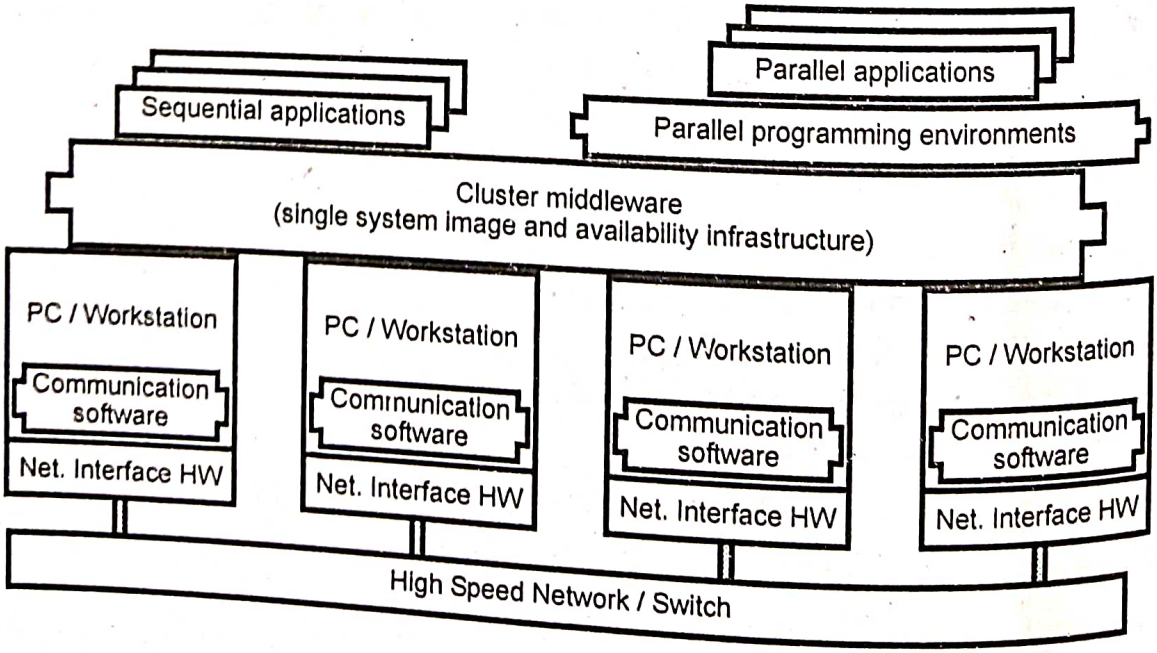


Fig. Q.57.1 Typical cluster architecture

- Each node has a network interface card, which connects the node to the network dedicated to the cluster. The **communication software (Sockets (TPC/IP), pipes etc.)** interfaces the node with the network interface card and hence the network. All requests for computation and replies are done through this communication software.
- The **Cluster Middleware (an optional component)** glues together all the nodes beneath the cluster to project a single system image to the applications working above it and to the final user. This single system image plays a major role in the efficient working and managing of the cluster. It is responsible for providing high availability, by means of load balancing and responding to failures in individual components.

Q.58 Give comparison between Clusters and SMPs.

☞ [SPPU : May-19, Marks 4]

Ans. :

Sr. No.	Cluster	SMP
1.	Multiprocessor system.	Multiprocessor system.
2.	Difficult to manage and configure.	Easy to manage and configure.
3.	Requires more physical space.	Requires less physical space
4.	Draws more power.	Draws less power.
5.	Not much closer to the original single processor model.	Much closer to the original single processor model.
6.	Cluster products are less established and stable than SMP products.	SMP products are well established and stable.
7.	Have more incremental and absolute scalability.	Incremental and absolute scalability is less than clusters.

8.	Since the scalability is more, can dominate high performance server market.	Since the scalability is low, less preferred in high performance server market.
9.	Clusters have higher availability.	SMPs have lower availability than clusters.

12.13 : UMA (Uniform Memory Access)

Q.59 Write a short note on UMA. ☞ [SPPU : Dec.-18, Marks 2]

Ans. : • Uniform Memory Access (UMA) is a shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly. In a UMA architecture, access time to a memory location is independent of which processor makes the request or which memory chip contains the transferred data.

- Uniform Memory Access computer architectures are often contrasted with Non-Uniform Memory Access (NUMA) architectures. In the UMA architecture, each processor may use a private cache. Peripherals are also shared. The UMA model is suitable for general purpose and time sharing applications by multiple users. It can be used to speed up the execution of single large program in time critical applications.

Types of UMA architectures

1. UMA using bus-based SMP architectures.
2. UMA using crossbar switches.
3. UMA using multistage switching networks.

12.14 : NUMA and CC-NUMA

Q.60 What is NUMA ? ☞ [SPPU : Dec.-18,19, Marks 2]

Ans. : • Non Uniform Memory Access or Non-Uniform Memory Architecture (NUMA) is a computer memory design used in multiprocessors, where the memory access time depends on the memory location relative to a processor. Under NUMA, a processor can access its

own local memory faster than non-local memory, that is, memory local to another processor or memory shared between processors.

Q.61 Draw and explain the CC-NUMA organization.

☞ [SPPU : Dec.-18, Marks 2]

Ans. : • The term CC-NUMA stands for **Cache-Coherent Non Uniform Memory Access.** In the CC-NUMA model, the system runs one operating system and shows only a single memory image to the user eventhough the memory is physically distributed over the processors. Since processors can access their own memory much faster than that of other processors, memory access is non uniform (NUMA). In this architecture the contents of the various processor caches should be coherent requiring extra hardware and a cache coherency protocol. A NUMA computer fulfilling these requirements is called a **CC-NUMA** machine.

- Fig. Q.61.1 shows the CC-NUMA organization. There are multiple independent nodes, each of which, in effect, an SMP organization. Thus, each node consists of multiple processors, each with its own L_1 and L_2 caches along with the main memory.

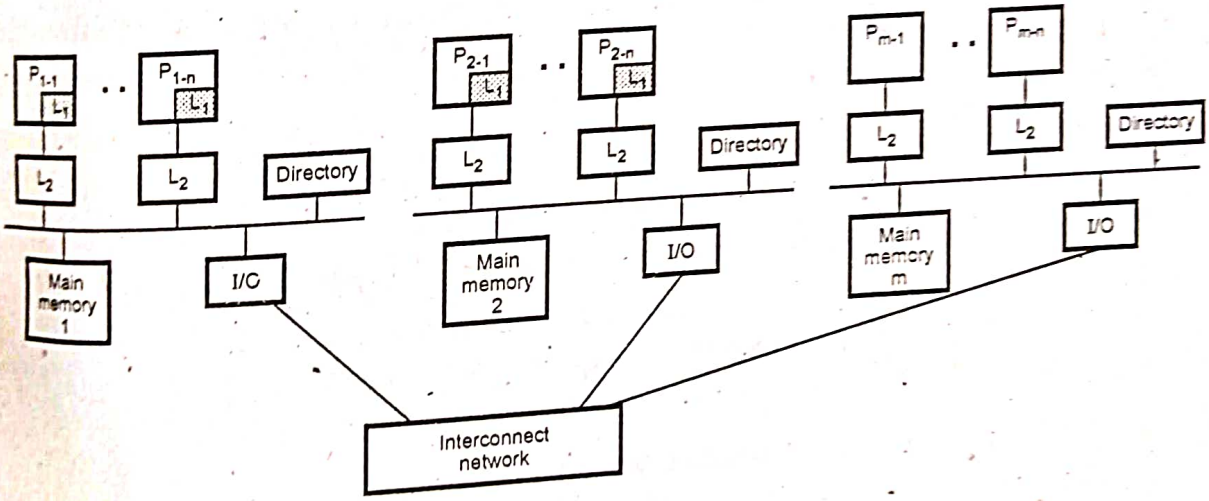


Fig. Q.61.1 CC-NUMA organization

- Comparing a CC-NUMA computer to cluster-based systems and SMP (Symmetric Multiprocessing) systems shows that CC-NUMA machines can be regarded as an attempt to get the best out of both worlds.
- SMP computers have multiple processors using the same memory which makes it easy to parallelize a code on loop level using compiler options and/or directives. However, due to the fact that processors have

to perform data exchange with their memory over the same bus having only a limited bandwidth, these systems will only scale to 10s of processors.

- Cluster-based systems do not have this drawback. The nodes in cluster-based machines have their own private memory and therefore each node possesses only a part of the data. Such machines will scale upto a very large number of processors if the computation to communication ratio of the program is high. The programming model for these computers is based on message passing and processors explicitly have to perform the communication with other processors by sending and receiving data. Programs have to take care of data distribution over the processors and have to be adapted for explicit communication. This implies that programs developed for single processor and SMP machines cannot be applied to these machines straight away. A substantial amount of redesigning and reprogramming of these codes is necessary.
- CC-NUMA machines combine the benefits of cluster-based systems and SMP machines. The fact that CC-NUMA machines behave like shared memory computers from a user point of view simplifies the porting of programs developed for single processor or SMP machines. Moreover, CC-NUMA computers allow for loop-level parallelism by means of compiler options or compiler directives similar to SMP systems. The good scalability properties are inherited from MPP/cluster-based systems since memory is distributed over the nodes.

12.15 : Multicore Architectures

Q.62 Write a note on multicore architecture.

☞ [SPPU : June-17,18, Marks 7]

OR What is multicore computers ? ☞ [SPPU : Dec.-16,17,18, Marks 3]

Ans. : • Multicore architectures are classified according to :

- Number of processor : Two processors (dual core), four processors (quad core), and eight processors (octa-core) configurations.
- Approaches to processor - to - processor communication.

- Cache and memory implementations.
- Implementation of I/O bus and the Front Side Bus (FSB).

- Fig. Q.62.1 shows three common configurations that support multiprocessing.

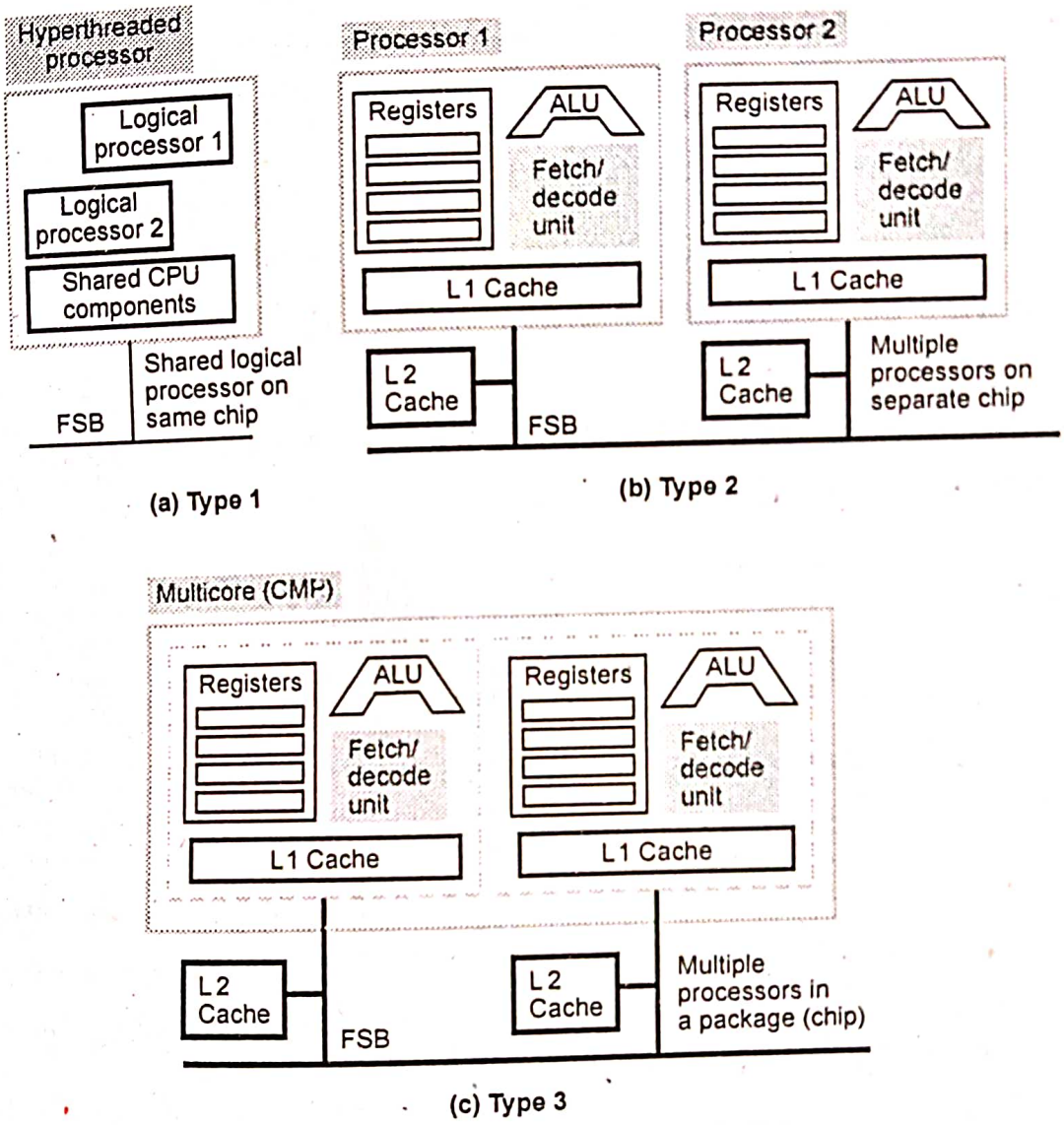


Fig. Q.62.1 Common configurations that support multiprocessing

- **Type 1 :** It uses hyperthreading technology. It allows a single processor to act like two separate processors to the operating system and the application programs that use it; however, there is a single processor running multiple threads. Thus, in hyperthreaded technology the multiple processors are logical instead of physical. In hyperthreaded technology has some duplication of hardware but not enough to qualify a separate physical processor.

- **Type 2 :** It is the classic multiprocessor. A multiprocessor is a tightly coupled computer system having two or more processing units (Multiple Processors) on a separate chip with its own hardware.
- **Type 3 :** It represents multicore system. It provides two or more complete processors on a single chip.

12.16 : Hardware Performance Issues

Q.63 Explain hardware performance issues of same.

☞ [SPPU : Dec.-16, Marks 3]

OR Describe hardware and software performance issues.

☞ [SPPU : May-19, Marks 5]

Ans. : Hardware Performance Issues : • Increase in Parallelism

- Pipelining
- Superscalar (multi-issue)
- Simultaneous multithreading (SMT)

All these improvement increase the complexity.

- **Power Consumption :** To maintain the trend of higher performance
  - Number of transistors per chip increase,
  - Designers have to choose more complicated processor designs (pipelining, superscalar, SMT) and to use high clock frequencies.
  - However, power requirements have grown exponentially as chip density and clock frequency have risen.

Software Performance Issues

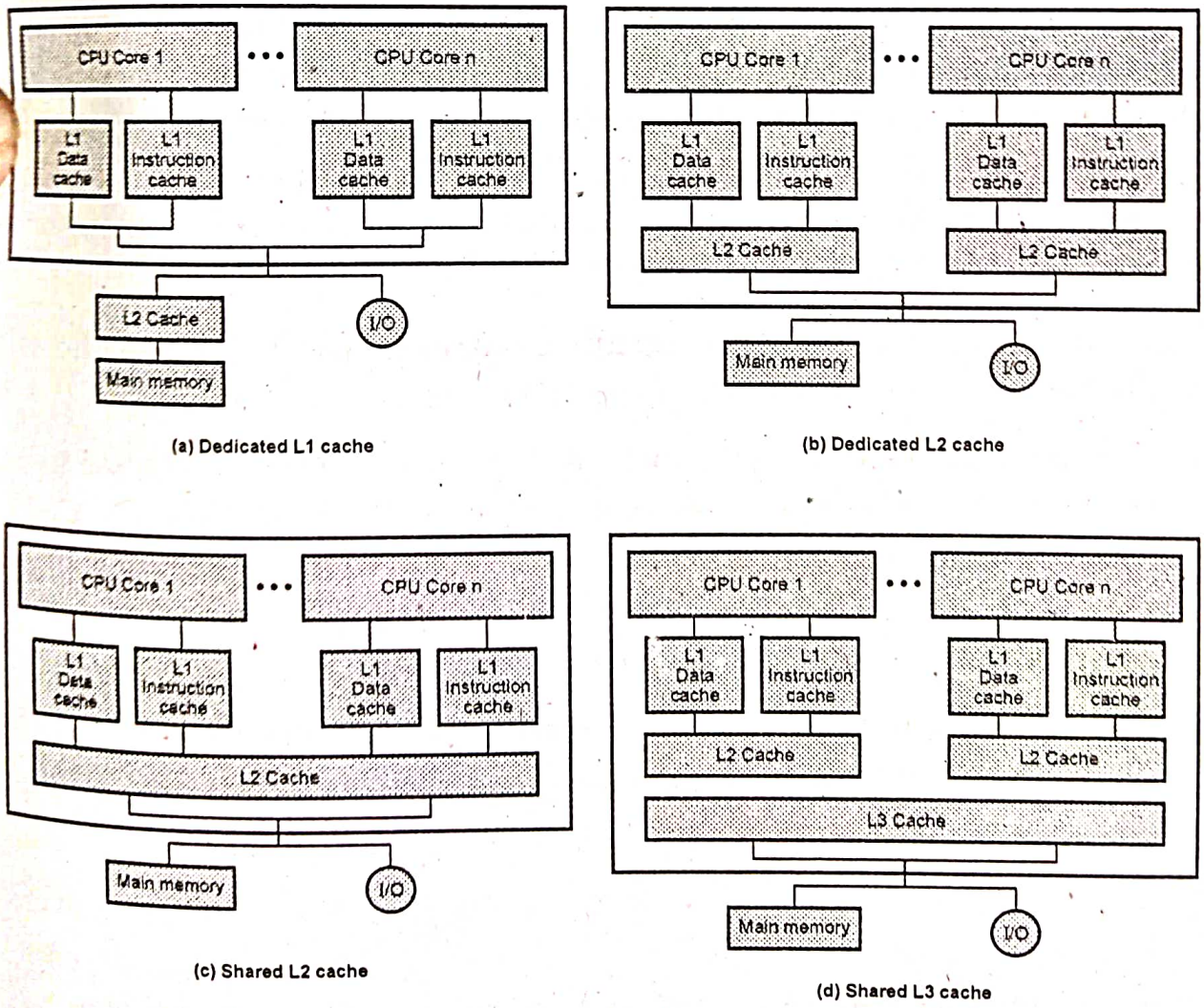
- Performance benefits dependent on effective exploitation of parallel resources.
- Even small amounts of serial code impact performance due to communication, distribution of work and cache coherence overheads. 10% inherently serial on 8 processor system gives only 4.7 times performance.
- Some applications effectively exploit multicore processors.

12.17 : Multicore Organizations

Q.64 Explain various multicore organization. ☞ [SPPU : May-19, Marks 2]

Ans. :

- Multicore organizations are classified according to :
  - The number of core processors on the chip.
  - The number of levels of cache memory.
  - The amount of cache memory that is shared.
- Fig. Q.64.1 shows four general organizations for multicore systems. The organization shown in Fig. Q.64.1 (a) has the only on-chip cache is L1 cache, with each core having its own dedicated L1 cache. Most of the times, the L1 cache is divided into instruction and data caches. An example of this organization is the ARM11 MPCore.



(a) Dedicated L1 cache

(b) Dedicated L2 cache

(c) Shared L2 cache

(d) Shared L3 cache

Fig. Q.64.1 Multicore organizations

- The organization shown in Fig. Q.64.1 (b) has no on-chip cache sharing. In this, there is enough area available on the chip to allow for L2 cache. An example of this organization is the AMD Opteron.
- The organization shown in Fig. Q.64.1 (c) has a similar allocation of chip space to memory, but with the use of a shared L2 cache. The Intel Core Duo has this organization.
- The organization shown in Fig. Q.64.1 (d) has shared L3 cache, with dedicated L1 and L2 caches for each core processor. The Intel Core i7 is an example of this organization.

12.18 : Intel X86 Multicore Organizations

Q.65 Draw and explain the organization of Intel Core Duo.

☞ [SPPU : May-18, Marks 3]

Ans. : • Caches and Thermal Unit : The Intel Core Duo, implements two x86 superscalar processors with a shared L2 cache. Fig. Q.65.1 shows the general structure of the Intel Core Duo. As shown in the Fig. Q.65.1, each core has a 32-kB instruction cache and a 32-kB data cache and an independent thermal control unit. The Core Duo thermal control unit is designed to manage chip heat dissipation to maximize processor performance within thermal constraints.

- **Advanced Programmable Interrupt Controller (APIC) :** It performs a number of functions, including the following :
  - Can provide interprocessor interrupts, which allow any process to interrupt any other processor or set of processors. In Core Duo, a thread in one core can generate an interrupt, which is accepted by the local APIC, routed to the APIC of the other core, and communicated as an interrupt to the other core.
  - Accepts I/O interrupts and routes these to the appropriate core.
  - Timer provided by each APIC can be set by the OS to generate an interrupt to the local core.
- **Power Management Logic :** It is responsible for reducing power consumption when possible, thus increasing battery life. It also monitors thermal conditions and adjusts voltage levels. It includes an advanced power-gating capability that allows for an ultra fine-grained

logic control that turns on individual processor logic subsystems only if and when they are needed. With power management logic, data required in some modes of operation can be put in a low power state when not needed.

- **Shared 2-MB L2 cache** : The cache logic allows for a dynamic allocation of cache space based on current core needs, so that one core can be assigned up to 100% of the L2 cache. The L2 cache includes logic to support the MESI protocol for the attached L1 caches.
- **Bus Interface** : It connects to the external bus, known as the Front Side Bus, which connects to main memory, I/O controllers, and other processor chips.

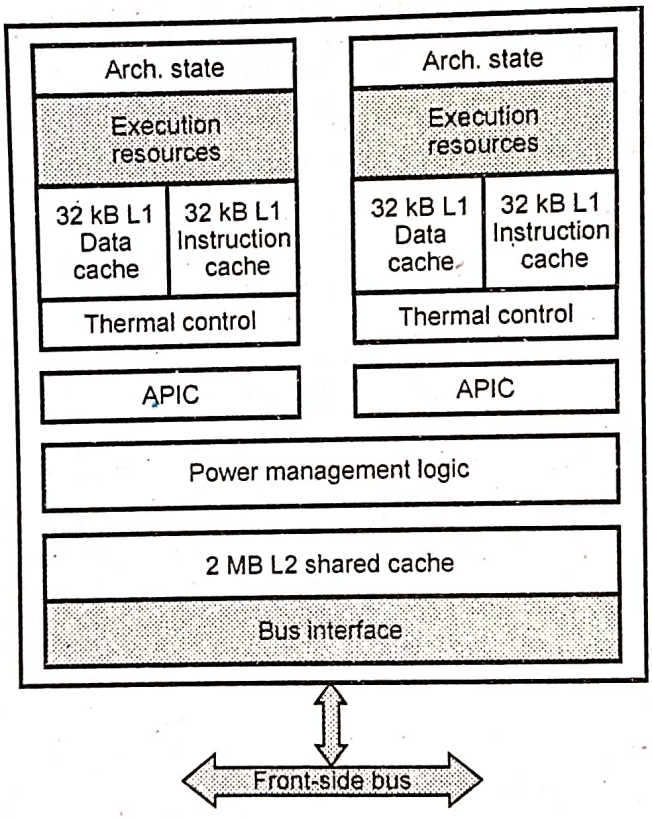


Fig. Q.65.1 Intel Core Duo block diagram

Q.66 Draw and explain the organization of Intel Core i7.

☞ [SPPU : May-18, Dec.-19, Marks 6]

Ans. : • The Intel Core i7, implements four x86 SMT processors, each with a dedicated L2 cache, and with a shared L3 cache. Fig. Q.66.1 shows the general structure of the Intel Core i7. Each core has its own dedicated L2 cache and the four cores share an 8-MB L3 cache. Prefetching is used to make its caches more effective.

- With the use of the dedicated L2 caches and a relatively high-speed access to the L3 cache, the Core i7 gives improved performance.
- The Core i7 chip supports two forms of external communications to other chips : The **DDR3 memory controller** and **QuickPath Interconnect (QPI).**
- **DDR3 memory controller :** It supports three channels that are 8 bytes wide for a total bus width of 192 bits, for an aggregate data rate of up to 32 GB/s. With the memory controller on the chip, the Front Side Bus is eliminated.
- **QuickPath Interconnect (QPI) :** It is a cache-coherent, point-to-point link based electrical interconnect specification for Intel processors and chipsets. It enables high-speed communications among connected processor chips. The QPI link operates at 6.4 GT/s (transfers per second). At 16 bits per transfer, that adds up to 12.8 GB/s, and since QPI links involve dedicated bidirectional pairs, the total bandwidth is 25.6 GB/s.

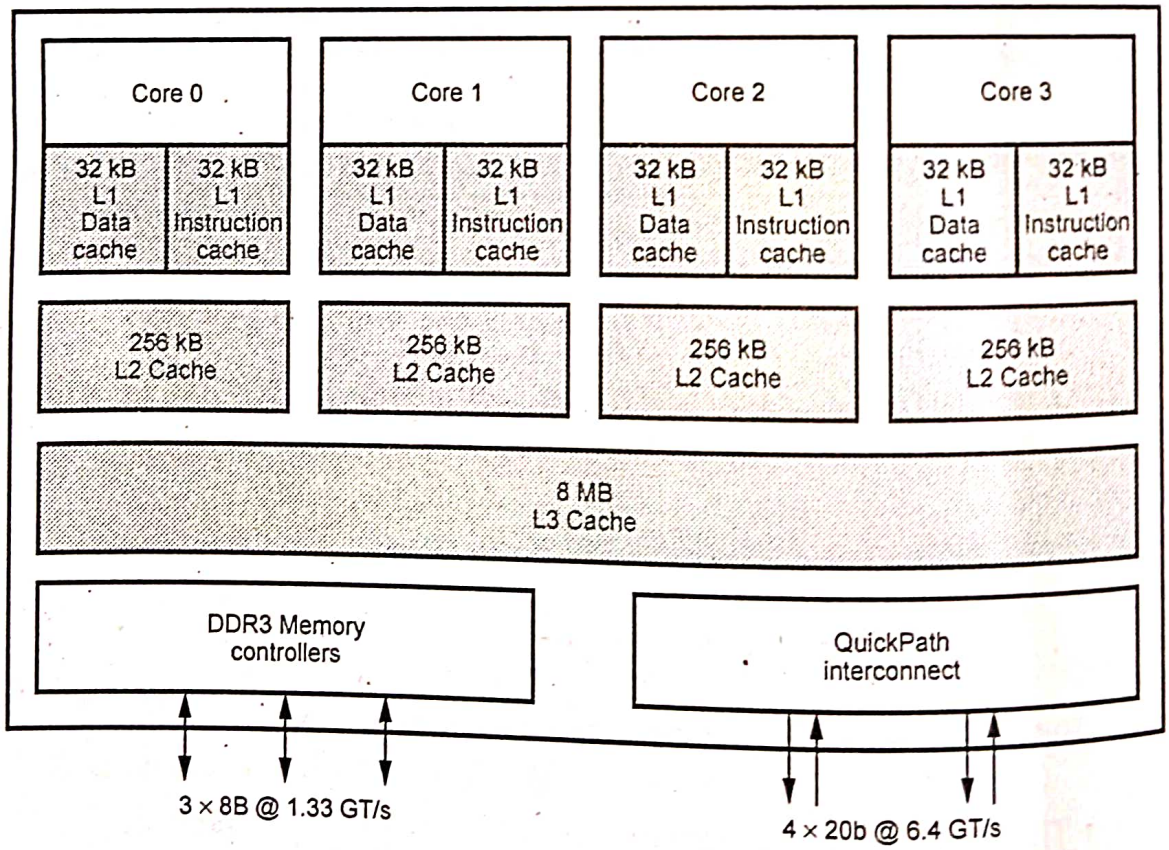


Fig. Q.66.1 Intel Core i7 block diagram

END...