DLCA QB

Ans:) Address Sequence Technique

- microinstructions are stored in control memory in groups, with each group specifying a routine.

Step 1:

- Initial address loaded when Power turn on.
- First microinstruction that activates fetch routine
- can be sequenced by incrementing the control address,
- At the end of fetch routine, the instruction is in the instruction register of the computer.

Step 2:

- Control memory next must go through the routine that determines the operand
- effective address computation routine in control memory can be reached through a branched microinstruction
- when completed the address of operand is available to computer

Step 3:

- Generate microoperations that execute the instruction fetched from memory
- Jn processor registers depends on the operation code part of the instruction
- each instruction has its own micropRogram routine stored in a given location of control memory.
- Transformation from instruction code bits to address in control memory where routine is located is called mapping
- It is a rule which transforms instruction code into a control memory address.

Step 4:

- Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.
- Microprograms will require an external register for storing the return address
- Return addresses cannot be stored in ROM because the unit has no writing capability.
- When execution of the instruction is completed, control must return to fetch routine
- This is done by executing an unconditional branch microinstruction to the first address of the fetch routine

Ans: 2

Hardwired control unit	Microprogrammed Control Unit
- It is a circuitry approach	- This control unit is implemented by programming.
- RISC style instructions	- CISC style instructions
- No control memory is required.	- Required.
- Works well for simple instruction.	- Works well for complex instruction
- limited no. of instructions are used due to hardware.	- Control signals for many instructions can be generated.

Page No.
Date

Ans: 3 Fetch cycle.

- The address of the next instruction to be executed is in the Program counter (PC)

MAR	
MBR	
PC	0000000001100100
IR	
AC	

Step1:

- The address in the program counter is moved to the memory address register (MAR). This is the only connected to address lines of the system bus.

MAR	0000000001100100
MBR	
PC	0000000011001000
IR	
AC	

Step2:

The address in MAR is placed on the address bus. The control unit issues a READ command on the control bus and the result appears on the data bus and is copied into the memory buffer register (MBR) and program counter is incremented by one.

MAR	0000000011001100
MBR	0001000001000000
PC	0000000011001000
IR	
AC	

Page No.
Date

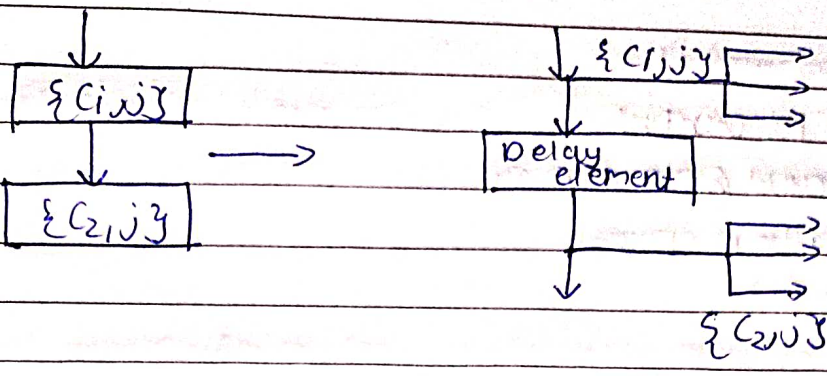
Step3:
The content of the MBR is moved to the instruction register (IR)

MAR	0000000011100100
MBR	0001000001000000
PC	0000000011100100
IR	0001000001000000
AC	

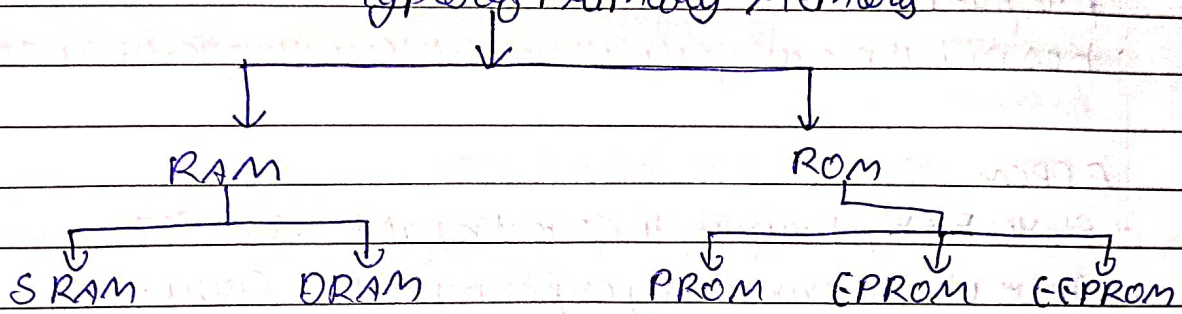
∴ t_1: MAR ← PC
t_2: MBR ← MEMORY
PC ← (PC) + I
t_3: IR ← (MBR)

Ans. 4 Delay element method
- In this control unit is represented in the form of flowchart
- Every step of flowchart at time t_i will activate $\{C_i, j\}$ where C_i is the clock stg. at time t_i for execution of instruction j
- Once the flowchart is complete, the individual clks. for each $\{C_i, j\}$ are formed.
- It is obvious that instruction j will be executed when all the steps of $\{C_i, j\}$ are performed when all the steps of $\{C_i, j\}$ are performed.
- There should be a finite time gap (delay) between every 2 steps
- The delay is introduced by D flipflop.

Delay time = $t_2 - t_1 = t_3 - t_2$ = one clk pulse.



Ans:5



RAM

- loses its contents when power is turned off
- called volatile memory
- used for storing bulk of the programs and data that are subject to change.
- each address contains a n bit word.

SRAM

- Easier to use and has shorter read and write cycle.
- Consists of internal flip flops that store binary inform.
- low density, high power, expensive and fast in oper.

DRAM

- Slower than SRAM
- Stores binary info in form of electric charge that are applied to capacitors.
- disappears within milisec.

ROM
- doesn't lose its contents.
- Can only read from memory
- Non-volatile memory.

PROM
- Memory can be programmed by user with special PROM programmer.
- has nichrome or poly silicon wires arranged in a matrix

EPROM
- Stores bit by charging the floating gate of an FET
- Info can be erased by exposing to UV light.

EEPROM
- Info can be changed by electrical signals at register level.
- Advantage in remote control
- Expensive.

Ans: 6 Characteristics of memory

1) location.
- location of memory device.
  i) CPU : form of CPU registers and small amount of cache.
  ii) Internal or main : like RAM or ROM, CPU can directly access the main memory.
  iii) External or secondary : comprises of sec. storage device like hard disk / magnetic tapes.

Page No.
Date

2) Capacity:
i) word size: 8 bits. commonly used are 1 byte (8 bits) 2 bytes (16 bits) 4 bytes (32 bits)
ii) No. of words: available in particular memory device.
eg: 4K x 16 means it has 16 bits and total of 4096 words in total.

3) Unit of Transfer.
- max no. of bits that can be read or written into the memory at a time.
- main memory is equal to word size.
- External memory is referred as blocks

4) Access methods.
i) Random Access
ii) Serial Access
iii) Semi random Access

5) Performance.
i) Access time: time taken by memory to complete read/ write. operation.
ii) Memory cycle time: defined only for RAM i.e sum of the access time and additional time required before the second access
iii) Transfer rate: rate at which data can be transferred into or out of memory unit.

6) Physical type: can be either semiconductor memory or magnetic surface memory.

7) Physical characteristics.
Volatile / Non-volatile: Continues to hold data even if power is turned off.

8) organization
Erasable / non-erasable: action of once programmed can be edited or not.

Ans: 7 Given:

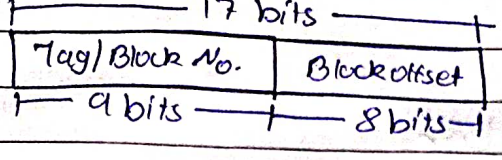
Cache memory size = 16 KB
Block size = 256 byte.
Main memory = 128 KB

No. of bits in physical Address
We have,
Size memory = 128 KB
$= 2^{17}$ bytes
$\therefore$ = 17 bits

No. of bits in Block offset.
We have,
Block size = 256 byte
$= 2^8$ byte.
= 8 bits

No. of bits in Tag
= no. of bits in physical - no. of bits in block
= 17 - 8
= 9 bits

No. of Lines in Cache
= Cache size / Line size
= 16 KB / 256 byte.
$= 2^{14} / 2^8$ byte
$= 2^6$ lines



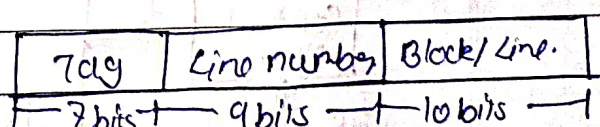
∴ Tag Directory size.
= No. of tags × Tag size
= no. of lines in cache × no. of bits in tag
= 2^6 × 9 bits
= 576 bits
= 72 bytes.

Ans: 8 Given:
memory size = 512KB
Block size = 1KB
No. of bits in Tag = 7bits

No. of bits in ~~Line Number~~ Block offset
= 1KB
= 2^{10} bytes
= 10 bits

No. of bits in Line number.
= Cache size / Line size.
= 512KB / 1KB
= 2^9 lines
= 9 bits

No. of bits in physical Address
= no. of bits in Tag + no. of bits in Line no. + no. of bits in block
= 7 bits + 9 bits + 10 bits
= 26 bits



Size of main Memory
we have,
no. of bits in physical address = 26 bits
= 2^{26} bytes
= 64MB

Tag Directory size

= No. of Tags x Tag size.

= no. of lines in cache x no. of bits in Tag

$= 2^9 \times 7$ bits

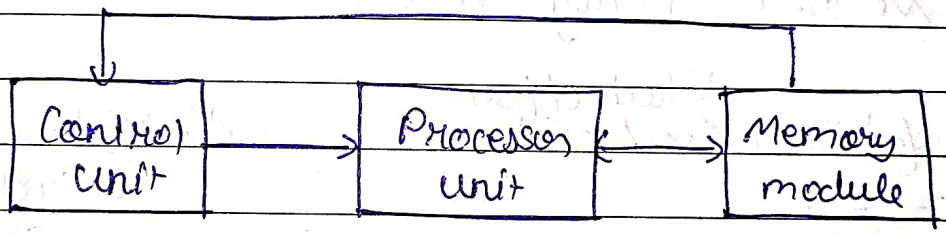
= 3584 bits

= 448 byts

Ans: 9 According to Flynn's Classification, either can be single or multiple.

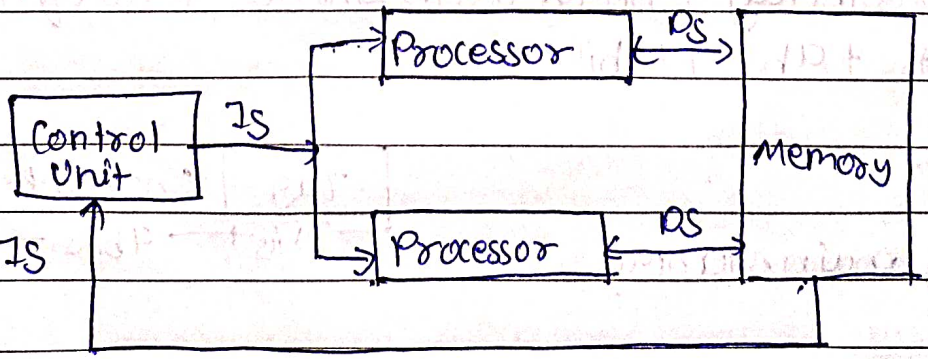
i) SISD : Single Instruction Single data stream.

- Single instruction is executed in single clock cycle.
- Data stored in single memory.

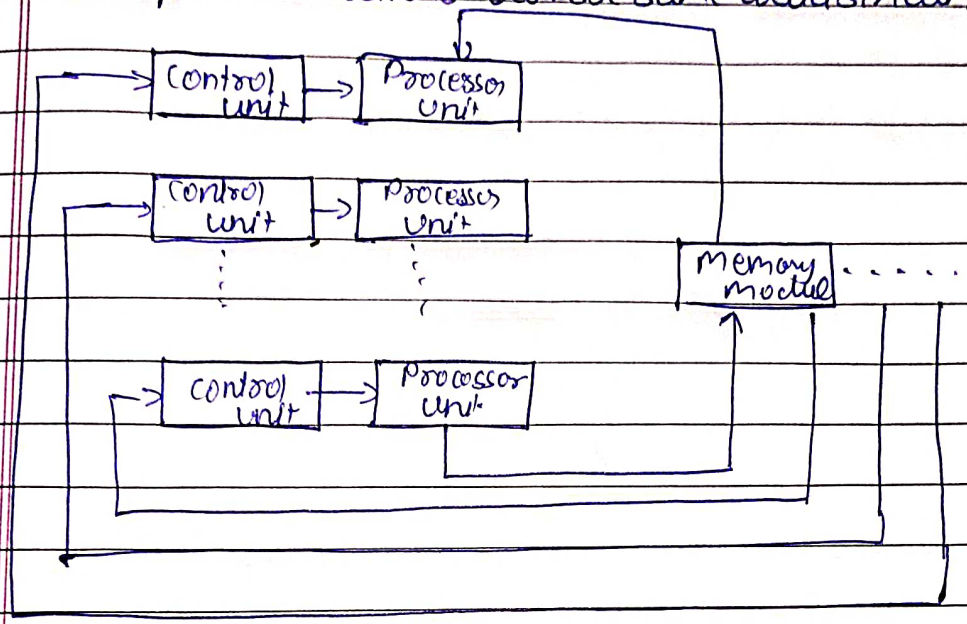


ii) SIMD : Single Instruction multiple data stream.

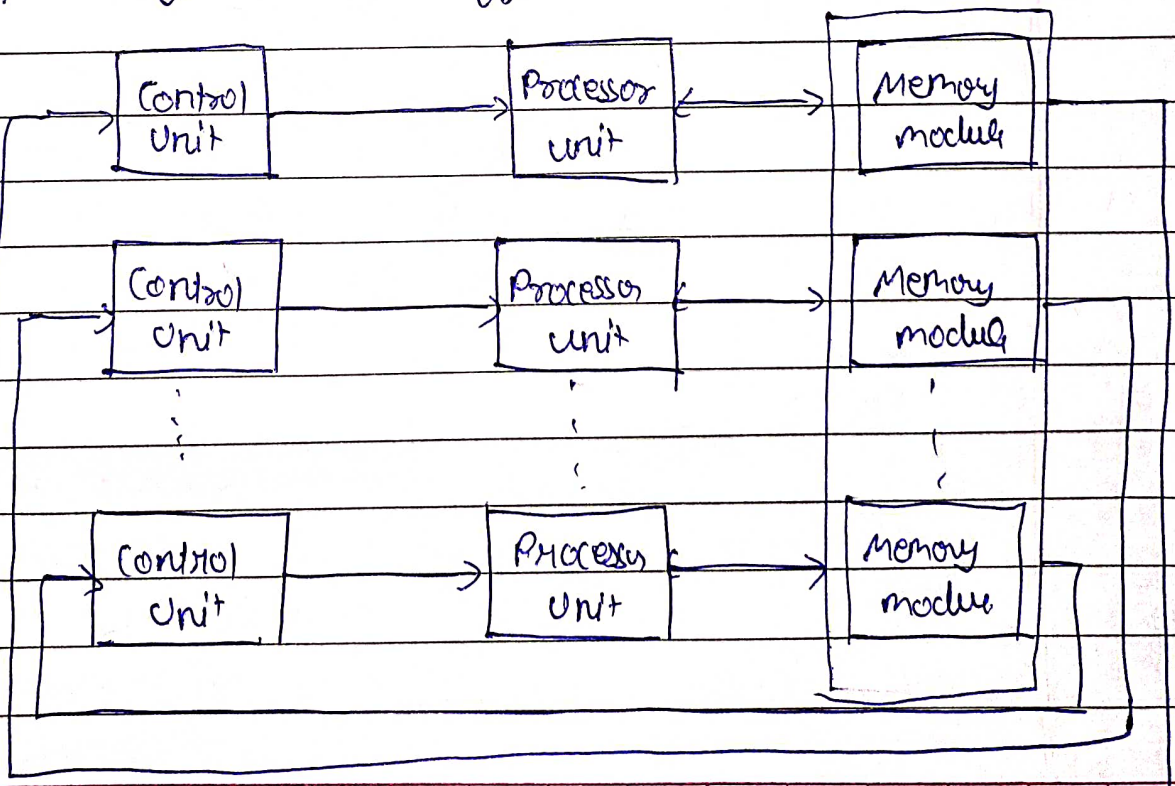
- Performs single identical urtions on multiple data
- All processors will execute same instructn. at any given clock cycle.



iii) MISD : Multiple Instructions single Data
- multiple processor, multiple control unit
- Each processor will use different algorithm
- Multiple instructions can use same data stream.



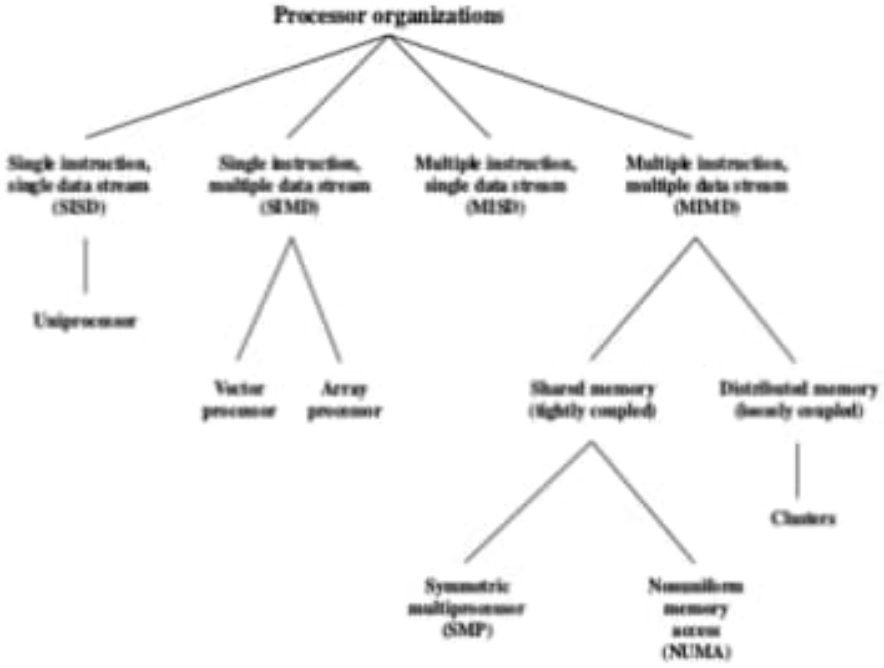
iv) MIMD
- Multiple memory modules connected together.
- most modern systems use MIMD
- Each processor in the system will operate asynchronously / independently on same or different data sets.



Flynn's Classification

Introduction

The most popular taxonomy of computer architecture was defined by Flynn in 1966. Flynn's classification scheme is based on the notion of a stream of information. Two types of information flow into a processor: instructions and data. The instruction stream is defined as the sequence of instructions performed by the processing unit. The data stream is defined as the data traffic exchanged between the memory and the processing unit.



According to Flynn's classification, either of the instruction or data streams can be single or multiple.

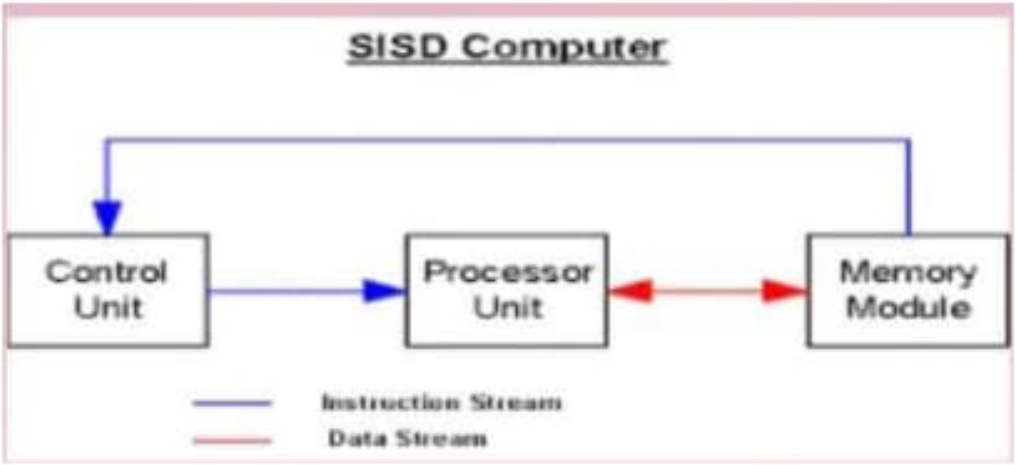
- Single-instruction single-data streams (SISD);
- Single-instruction multiple-data streams (SIMD);
- Multiple-instruction single-data streams (MISD); and
- Multiple-instruction multiple-data streams (MIMD).

Conventional single-processor von Neumann computers are classified as SISD systems. Parallel computers are either SIMD or MIMD. When there is only one control unit and all processors execute the same instruction in a synchronized fashion, the parallel machine is classified as SIMD. In a MIMD machine, each processor has its own control unit and can execute different instructions on different data.

SISD Architecture

In computing, SISD is a computer architecture in which a single uni-core processor, executes a single instruction stream, to operate on data stored in a single memory.

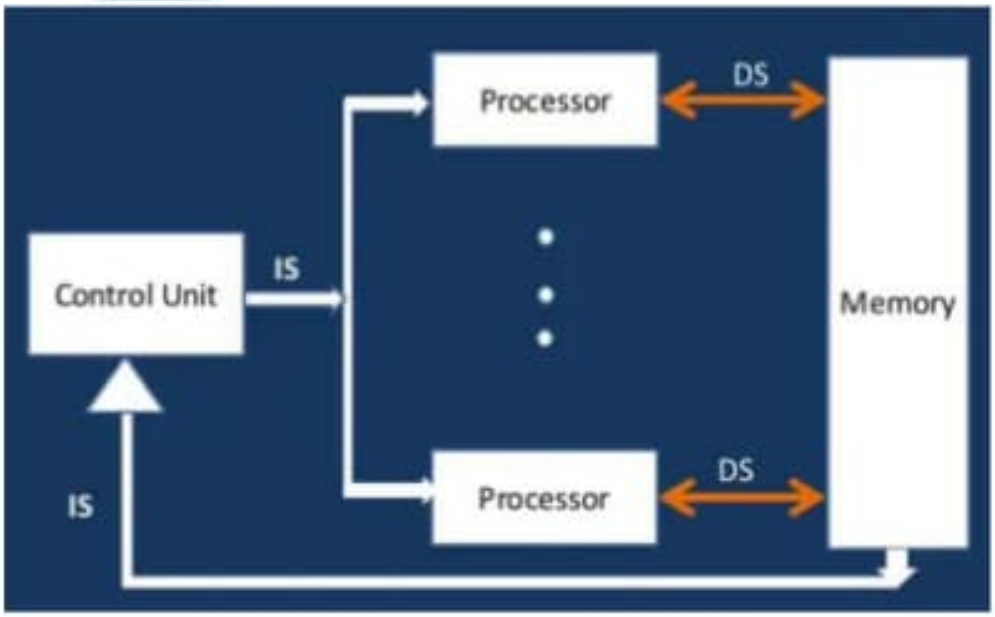
1. Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle
2. Single data: only one data stream is being used as input during any one clock cycle Deterministic execution
3. This is the oldest and until recently, the most prevalent form of computer Examples: most PCs, single CPU workstations and mainframes



SIMD Architecture

The SIMD architecture performs a single, identical action simultaneously on multiple data pieces. Here we have a single control unit (CU) and more than one processing unit (PU). For e.g. a single instruction to fetch multiple files.

- Single instruction: All processing units execute the same instruction at any given clock cycle
- Multiple data: Each processing unit can operate on a different data element
- This type of machine typically has an instruction dispatcher, a very high-bandwidth internal network, and a very large array of very small-capacity instruction units.
- Best suited for specialized problems characterized by a high degree of regularity, such as image processing.



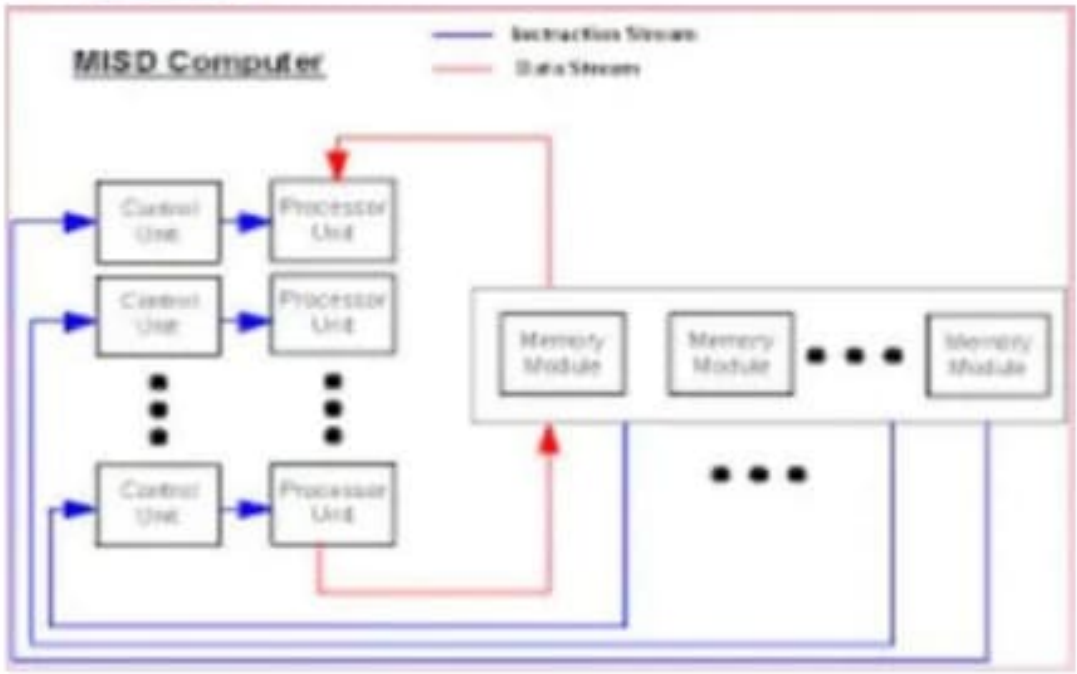
- Two varieties: Processor Arrays and Vector Pipelines Examples:
  - Processor Arrays: Connection Machine CM-2, Maspar MP-1, MP-2
  - Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820

MISD Architecture

Multiple Instruction, Single Data (MISD) computers have multiple processors.Each processor uses a different algorithm but uses same shared input data. MISD computers can analyze same set of data using several different operations at same time. The number of operations depends upon number of processors.

There aren't many actual examples of MISD computers,for e.g. fault-tolerant computers executing the same nstructions redundantly in order to detect and mask errors.

- A single data stream is fed into multiple processing units.
- Each processing unit operates on the data independently via independent instruction streams.
- Few actual examples of this class of parallel computer have ever existed. One is the experimental Carnegie-Mellon C.mmp computer (1971).
- Some conceivable uses might be:
  - Multiple frequency filters operating on a single signal stream
  - Multiple cryptography algorithms attempting to crack a single coded message.



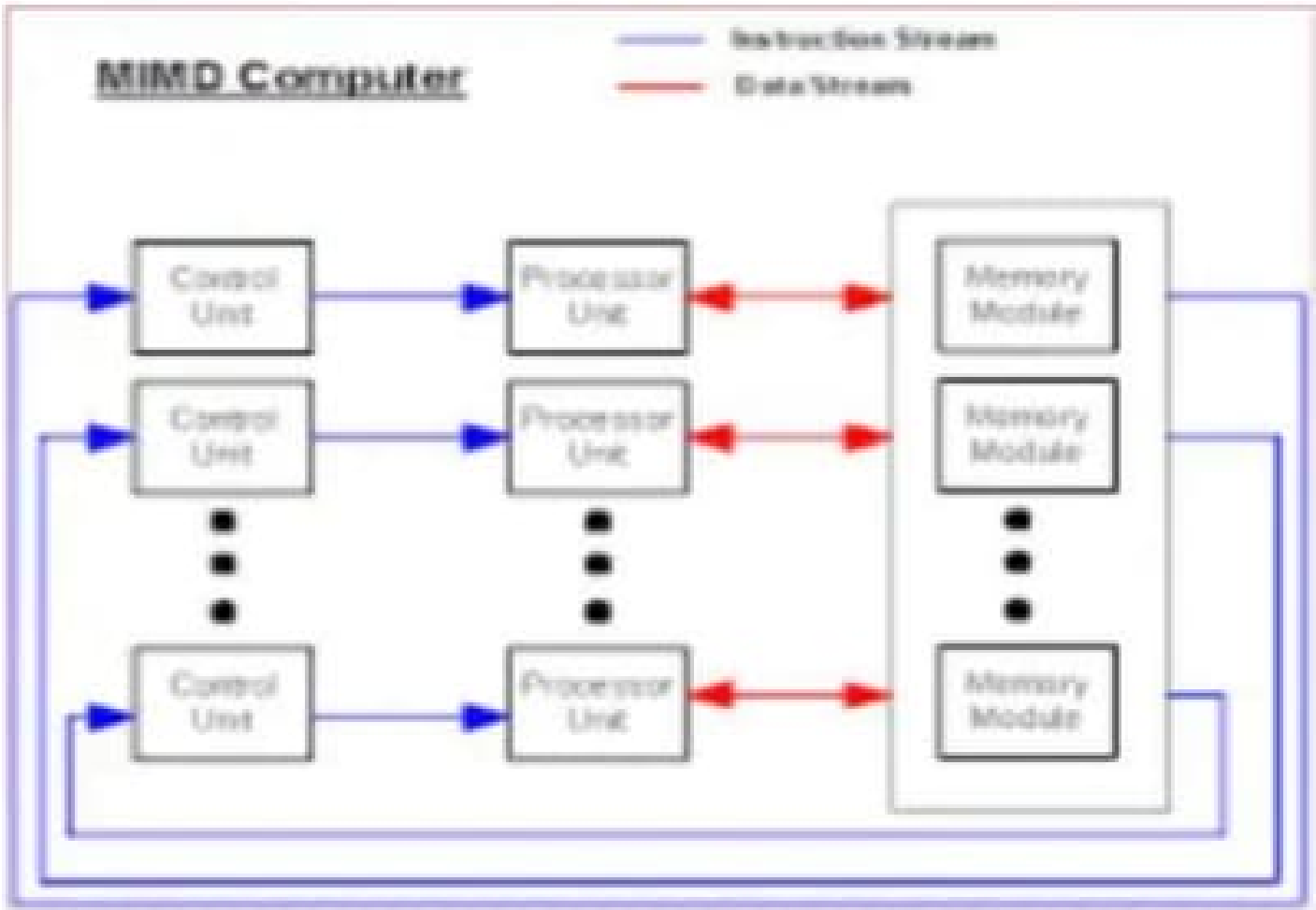
MIMD Architecture

Multiple-instruction multiple-data streams (MIMD) parallel architectures are made of multiple processors and multiple memory modules connected together via some interconnection network. It means that parallel units have separate instructions, so each of them can do something different at any given time.

While technically it's true that most modern desktop/laptops are MIMD. Using the MIMD, each processor in a multiprocessor system can execute asynchronously different set of the instructions independently on the different set of data units.

They fall into two broad categories: shared memory or message passing. Processors exchange information through their central shared memory in shared memory systems, and exchange information through their interconnection network in message passing systems.

- Currently, the most common type of parallel computer. Most modern computers fall into this category.
- Multiple Instruction: every processor may be executing a different instruction stream Multiple Data: every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Examples: most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.
- A shared memory system typically accomplishes interprocessor coordination through a global memory shared by all processors. These are typically server systems that communicate through a bus and cache memory controller.


MIMD Computer
Control Unit
Control Unit
Control Unit
Processor Unit
Processor Unit
Processor Unit
Memory Module
Memory Module
Memory Module

10)

PIPELINING

→ Pipelining is the process of arrangement of hardware elements of CPU such that its overall performance is increased.

→ The pipeline design provides a way to start a new task before an old one has been completed.

Performance of Pipelined Execution-

The following parameters serve as criterion to estimate the performance of pipelined execution-

Speed Up
Efficiency
Throughput

1. Speed Up-

It gives an idea of "**how much faster**" the pipelined execution is as compared to non-pipelined execution.

It is calculated as-

$$\text{Speed Up (S)} = \frac{\text{Non-pipelined execution time}}{\text{Pipelined execution time}}$$

2. Efficiency-

The efficiency of pipelined execution is calculated as-

$$\text{Efficiency } (\eta) = \frac{\text{Speed Up}}{\text{Number of stages in Pipelined Architecture}}$$

OR

$$\text{Efficiency } (\eta) = \frac{\text{Number of boxes utilized in phase-time diagram}}{\text{Total number of boxes in phase-time diagram}}$$

3.Throughput-

Throughput is defined as number of instructions executed per unit time.

It is calculated as-

$$\text{Throughput} = \frac{\text{Number of instructions executed}}{\text{Total time taken}}$$

AMDAHL'S LAW

11

AMDAHL'S LAW.

→ Amdahl's law relates the improvement of system's performance with the parts that didn't perform well.

→ This law is often used in parallel computing to predict the theoretical speedup when using multiple processors.

→ Amdahl's law is expressed mathematically as

$$\text{Speed up max} \leq \frac{1}{F + \frac{(1-F)}{P}}$$

$$S(P) \leq \frac{1}{F + \frac{(1-F)}{P}}$$

where P : no. of processors
F : sequential fraction of program
1-F : parallel fraction of program.

PROOF:
Consider execution of a task by a single processor as well as P no. of processors.
Let F be the sequential part of program
∴ 1-F will be the parallel part of program.

$$\text{Speed up} = \frac{\text{Time taken by single processor}}{\text{Time taken by } p \text{ no of processors}}$$

$$= \frac{T(s)}{T(p)}$$

Let $T(s) = T$

$\therefore T(p) =$ Sequential time + Parallel time

$$= F \cdot T + \frac{(1-F) \cdot T}{p}$$

$$S(p) = \frac{T}{F \cdot T + \frac{(1-F) \cdot T}{p}}$$

$$S(p) = \frac{1}{F + \frac{(1-F)}{p}}$$

$S(p)$ will always be less than or equal to 1

$$\therefore S(p) \leq \frac{1}{F + \frac{(1-F)}{p}}$$

to the branch instruction to be executed in any case before actually branching to the target address.

– Therefore, utilizing some time out of the time wasted in the pipeline stall due to pipeline flushing. This instruction which is shifted to the delayed branch slot must be an instruction that is an independent instruction that does not affect the outcome of the branch directly or indirectly.

UEx. 11.2.1

MU - Q. 2(a), Dec. 18, Q. 2(B), May 19, 10 Marks

A program having 10instructions (without Branch and Call instructions) is executed on non-pipeline and pipe line processors. All instructions are of same length and having 4pipeline stages and time required to each stage is 1 nsec.

i) Calculate time required to execute the program on Non-pipeline and Pipeline processor.

ii) Calculate Speedup

☑ **Soln. :**

The number of instructions executed are 10.

Therefore n = 10

For pipeline stages are given as 4. Therefore k = 4

Now the time required for Non-pipelined processor is given as

$$S_{np} = n * k = 10 * 4 = 40 \text{ nS}$$

And time required for Pipelined Processor (4 Stage) is given as

$$S_p = n + k - 1 = 10 + 4 - 1 = 13 \text{ nS}$$

Further the Speed-up is given as Ratio of times for Non-Pipelined and Pipelined processors –

$$S = (n*k) / (n + k - 1) = S_{np} / S_p = 40/13$$

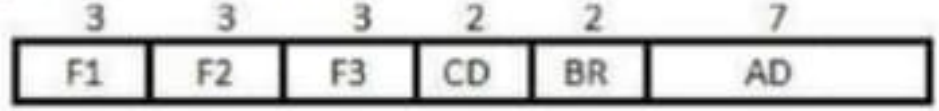
Therefore, Speedup S = 3.08

Microinstruction Format

The microinstruction format for the control memory is shown in figure 4.5. The 20 bits of the microinstruction are divided into four functional parts as follows:

1. The three fields F1, F2, and F3 specify microoperations for the computer.

   The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.
2. The CD field selects status bit conditions.
3. The BR field specifies the type of branch to be used.
4. The AD field contains a branch address. The address field is seven bits wide, since the control memory has $128 = 2^7$ words.



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

Figure 4.5: Microinstruction Format

- As an example, a microinstruction can specify two simultaneous microoperations from F2 and F3 and none from F1.

  DR ← M[AR] with F2 = 100
  PC ← PC + 1 with F3 = 101
- The nine bits of the microoperation fields will then be 000 100 101.
- The CD (condition) field consists of two bits which are encoded to specify four status bit conditions as listed in Table 4.1.

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

Table 4.1: Condition Field

14)

A block set associative cache consists of 64 blocks divided in 4 block sets. The main memory contains 4096 blocks, each 128 words of 16 bit length.

i) How many bits are there in main memory addr?

ii) How many bits are there in cache memory address (tag, set and word field)?

DATA:

No. of blks in cache memory = 64

No. of blocks in each set of cache = 4.

Main memory size = 4096 blocks

Block size = 128 words

1 word = 16 bit

SOLN:-

Main memory size = 4096 blocks

$= 4096 \times 128$ words

$= 2^{12} \times 2^{7}$ words

$= 2^{19}$ words

∴ No. of bits required to address main memory = 19

Block size = 128 words

= 2^7 words

∴ No. of bits in block offset or word = 7 bits.

No. of sets in cache = $\frac{\text{No. of blks in cache}}{\text{Set size}}$

$= \frac{64}{4}$

$= 16$

$= 2^4$

∴ No. of bits in set number = 4 bits.

No. of bits in tag =

No. of bits in physical addr −

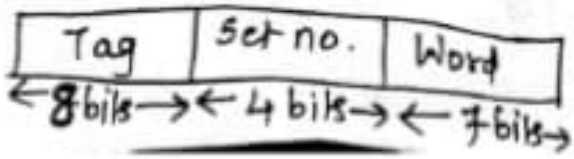
(No. of bits in set no. +

No. of bits in word)

$= 19 - (4 + 7)$

$= 8$ bits

∴ No. of bits in tag = 7 bits



SRAM	DRAM
It stores information as long as the power is supplied.	It stores information as long as the power is supplied or a few milliseconds when power is switched off.
Transistors are used to store information in SRAM.	Capacitors are used to store data in DRAM.
Capacitors are not used hence no refreshing is required.	To store information for a longer time, contents of the capacitor needs to be refreshed periodically.
SRAM is faster needcompared to DRAM.	DRAM provides slow access speeds.
It does not have a refreshing unit.	It has a refreshing unit.
These are expensive.	These are cheaper.
SRAMs are low-density devices.	DRAMs are high-density devices.
In this bits are stored in voltage form.	In this bits are stored in the form of electric energy.
These are used in cache memories.	These are used in main memories.
Consumes less power and generates less heat.	Uses more power and generates more heat.

15)

10

SUMS ON SET ASSOCIATIVE CACHE MAPPING

1. A block-set associative cache memory consists of 128 blocks divided into four block sets . The main memory consists of 16,384 blocks and each block contains 256 eight bit words.

   How many bits are required for addressing the main memory?

   How many bits are needed to represent the TAG, SET and WORD fields?

   Solution-

   Given-

   - Number of blocks in cache memory = 128
   - Number of blocks in each set of cache = 4
   - Main memory size = 16384 blocks
   - Block size = 256 bytes
   - 1 word = 8 bits = 1 byte

   Main Memory Size-

   We have-

   Size of main memory

   = 16384 blocks

   = 16384 x 256 bytes

   = 2^{22} bytes

   Thus, Number of bits required to address main memory = 22 bits

   Number of Bits in Block Offset-

   We have-

   Block size

   = 256 bytes

   = 2^8 bytes

   Thus, Number of bits in block offset or word = 8 bits

   Number of Bits in Set Number-

Number of sets in cache

= Number of lines in cache / Set size

= 128 blocks / 4 blocks

= 32 sets

= 2^5 sets

Thus, Number of bits in set number = 5 bits

Number of Bits in Tag Number-

Number of bits in tag

= Number of bits in physical address – (Number of bits in set number + Number of bits in word)

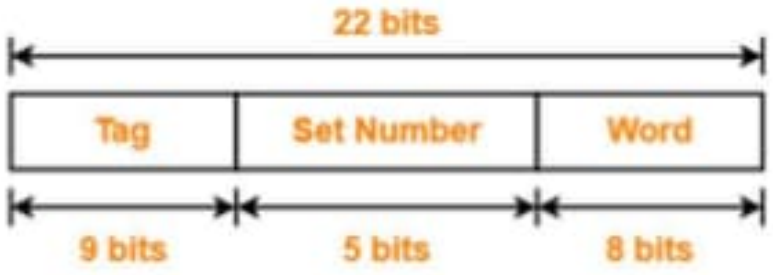
= 22 bits – (5 bits + 8 bits)

= 22 bits – 13 bits

= 9 bits

Thus, Number of bits in tag = 9 bits

Thus, physical address is-



17)

2. A 4-way set associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is_.

Solution-

Given-

- Set size = 4 lines
- Cache memory size = 16 KB
- Block size = 8 words
- 1 word = 32 bits = 4 bytes
- Main memory size = 4 GB

Number of Bits in Physical Address-

We have,

Main memory size

= 4 GB

= 2^{32} bytes

Thus, Number of bits in physical address = 32 bits

Number of Bits in Block Offset-

We have,

Block size

= 8 words

= 8 x 4 bytes

= 32 bytes

= 2^5 bytes

Thus, Number of bits in block offset = 5 bits

Number of Lines in Cache-

Number of lines in cache

= Cache size / Line size

= 16 KB / 32 bytes

= 2^{14} bytes / 2^5 bytes

= 2^9 lines

= 512 lines

Thus, Number of lines in cache = 512 lines

Number of Sets in Cache-

Number of sets in cache

= Number of lines in cache / Set size

= 512 lines / 4 lines

= 2^9 lines / 2^2 lines

= 2^7 sets

Thus, Number of bits in set number = 7 bits

Number of Bits in Tag-

Number of bits in tag

= Number of bits in physical address – (Number of bits in set number + Number of bits in block offset)

= 32 bits – (7 bits + 5 bits)

= 32 bits – 12 bits

= 20 bits

Thus, number of bits in tag = 20 bits

18)

STATE TABLE METHOD

→ In this the behaviour of control unit is represented in the form of a table which is known as state table as shown below.

	Input combinations Cin				
State	I_1	I_2	I_3		I_m
S_1	S_{11}, Z_{11}	S_{12}, Z_{12}	. . . .		S_{1m}, Z_{1m}
S_2	S_{21}, Z_{21}	S_{22}, Z_{22}	. . . .		S_{2m}, Z_{2m}
S_3	S_{31}, Z_{31}	S_{32}, Z_{32}	. . . .		
.	.	.	.		.
.	.	.	.		.
.	.	.	.		.
S_n	S_{n1}, Z_{n1}	S_{n2}, Z_{n2}	. . . .		S_{nm}, Z_{nm}

Next state ↑ ↑ O/p ckt signal to be generated (Cout)

→ The table shows the corresponding o/p that should be generated when different inputs are applied at various states.

→ Let Cin and Cout denote the i/p and o/p variables of control unit

→ Each row in the state table corresponds to an internal state {Si} of the control unit.

→ A state is determined by the information stored in the processor at that unit of time.

→ Each column denotes different set of external i/p signals applied to the control unit {Cin}.

→ The intersection of row Si and column Ij means the following:

i) When Input Ij is applied to state Si, we get Sij, Zij

ii) Zij is the set of o/p signals to be activated..

iii) Sij should become the next state of control unit.

Advantage: Simplest method to implement hardwired control unit.