LECTURE

Content : Architectural classification schemes

Flynn's classification : Michael flynn (1972) introduced a classification of various computer architectures based on notions of instructions and data streams.

1). Single Instruction stream, Single data stream (SISD)
2). Single Instruction stream, Multiple data stream (SIMD)
3). Multiple Instruction stream, Single data stream (MISD)
4). Multiple Instruction stream, Multiple data stream (MIMD)

1. **SISD** : SISD represents the organization of a single computer containing a control unit, a processor unit, and a memory unit. Instructions are executed sequentially and the system may or may not have internal parallel processing capabilities.

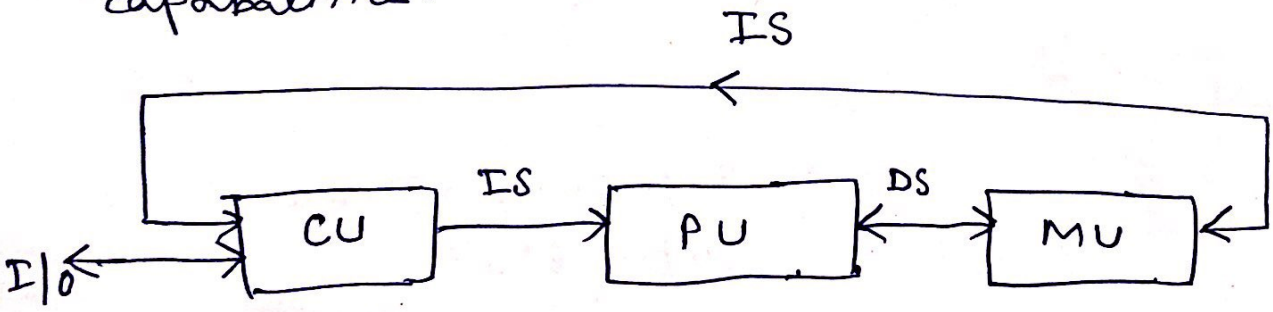


Fig: SISD

2). SIMD : SIMD represents an organization that includes many processing units under the supervision of a common control unit. All processors receives the same instruction from the control unit but operate on different items of data. The shared memory unit must contain multiple modules so that it.

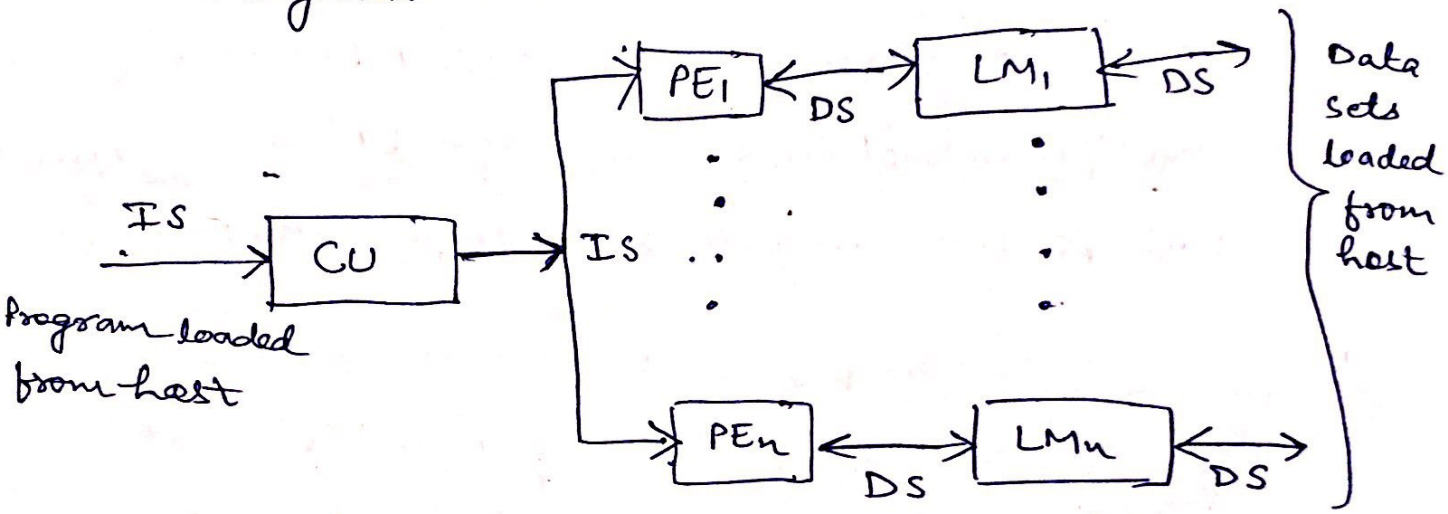


Fig: SIMD

- Can communicate with all the processors simultaneously.

3). MISD :

MISD structure is only of theoretical interest since no practical system has been constructed using this organization.

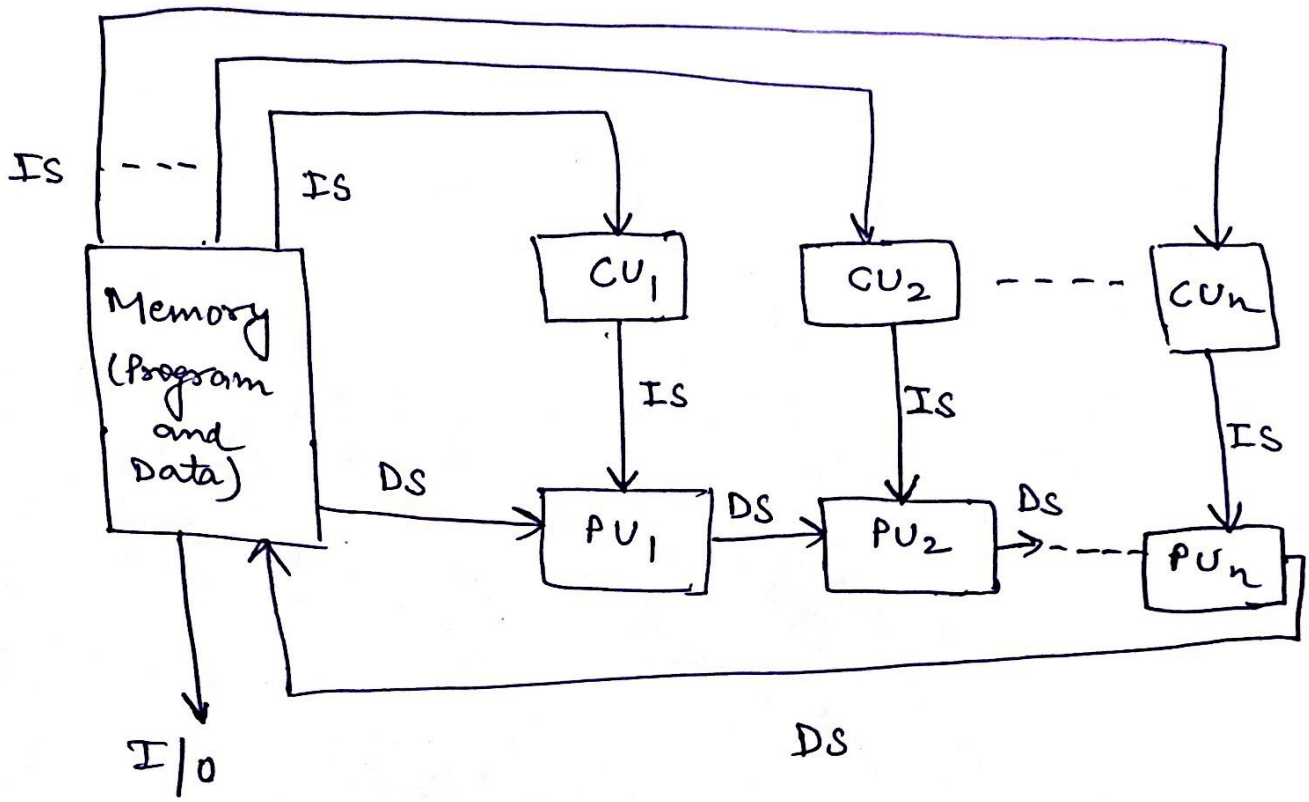


fig : MISD

4] MIMD : MIMD organization refers to a computer system capable of processing several programs at the same time.

Most multiprocessor and multicomputer system can be classified in this catagory.

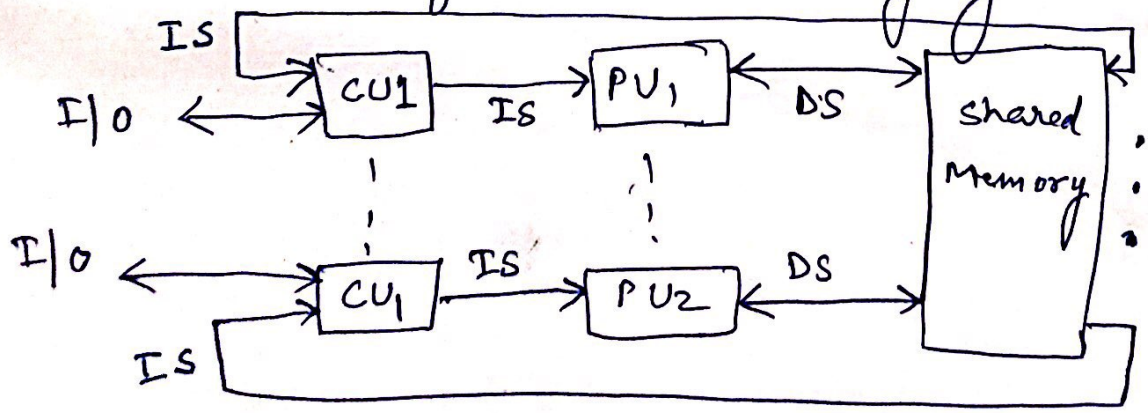


fig : MIMD

Feng's classification: Feng's classification is mainly based on degree of parallelism to classify parallel computer architecture.

There are four types of processing methods according to Feng's classification:

1- **Word Serial and Bit Serial (WSBS)**: WSBS has been called as bit serial processing because one bit ($n=m=1$) is processed at a time, which was a slow process. This was done in only first generation computers.

2- **Word Parallel and Bit Serial (WPBS)**: WPBS ($n=1$, $m>1$) has been called bit slice processing because m-bit slice is processed at a time.

3- **Word Serial and Bit Parallel (WSBP)**:

WSBP ($n>1$, $m=1$) has been called word-slice processing because one word of n bits is processed at a time. These are found in most existing computers

4- **Word Parallel and bit Parallel (WPBP)**: WPBP ($n>1$, $m>1$) is known as fully parallel processing, in which an array of $n \times m$ bits is processed at a time. This is the fastest mode of the four methods.

System Attributes to Performance :

Clock Rate and CPI :

The CPU (or simply the processor) of today's digital computer is driven by a clock with a constant cycle time (τ in nanoseconds). The inverse of the cycle time is the clock rate ($f = 1/\tau$ in megahertz).

The size of a program is determined by its instruction count (Ic), in terms of the number of machine instructions to be executed in the program. Different machine instructions may require different numbers of clock cycles to execute. Therefore, the cycle per instruction (CPI) becomes an important ~~factor~~ parameter for measuring the time needed to execute each instruction.

Performance factors :

Let Ic be the number of instructions in a given program, or the instruction count. The CPU time (T in seconds/program) needed to execute the program is estimated by finding the product of three contributing factors:

$$T = Ic \times CPI \times \tau \quad \text{——————} \quad (1.1)$$

The execution of an instruction requires through a cycle of events involving the instruction fetch, decode, operand fetch, execution and store results.

In this cycle, only the instruction decode and execution phases are carried out in the CPU. The remaining three operations may be required to access the memory.

We define a memory cycle as the time needed to complete one memory reference. Usually, a memory cycle is k times the processor cycle τ.

The value of K depends on the speed of the memory technology and processor-memory interconnection scheme used.

The CPI of an instruction type can be divided into two component terms corresponding to the total processor cycle and memory cycles needed to complete the execution of the instruction. Depending on the instruction type, the complete instruction cycle may involve one to four memory references (one for instruction fetch, two for operand fetch, and one for store results).

Therefore, we can rewrite eq. (1.1) as follows:

$$T = I_c \times (p + m \times k) \times \tau \quad \text{——— (1.2)}$$

where p is the number of processor cycles needed for the instruction decode and execution, m is the number of memory references needed, k is the ratio between memory cycle and processor cycle, I_c is the instruction count, and τ is the processor cycle time.

System Attributes : the above five performance factors (I_c, p, m, k, τ) are influenced by four system attributes: instruction-set architecture, compiler technology, CPU implementation and control, and cache and memory hierarchy as specified in Table.

Table: Performance Factors versus System Attributes

System Attributes	Performance factors				
	Inst. count I_c	Average Cycles per Inst., CPI			Processor cycle Time, τ
		Processor cycles per Inst. p	Memory reference per Inst. m	Memory Access Latency, k	
Instruction-set Architecture	X	X			
Compiler Technology	X	X	X		
Processor Implementation and Control		X			X
Cache and Memory hierarchy				X	X

MIPS Rate: Let C be the total number of clock cycles needed to execute a given program. Then the CPU time in Eq. (1.2) can be estimated as $T = C \times \tau = C/f$. Furthermore, $CPI = C/I_c$ and $T = I_c \times CPI \times \tau$

$$= I_c \times CPI / f.$$

The processor speed is often measured in terms of million instructions per second (MIPS).

$$\text{MIPS Rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6} = \frac{f \times I_c}{C \times 10^6} \quad \text{——(1.3)}$$

Based on Eq (1.3), the CPU time in Eq. (1.2) can also be written as $T = I_c \times 10^{-6} / \text{MIPS}$.

Throughput Rate: Another important concept is related to how many programs a system can execute per unit time, called system throughput W_S (in programs/second). In a multiprogrammed system, the system throughput is often lower than the CPU throughput W_p defined by:

$$W_p = \frac{f}{I_c \times CPI} \qquad (1.4)$$

Note that $W_p = MIPS \times 10^6 / I_c$ from eq. (1.3). The unit for W_p is programs/second.

LECTURE

Speed up performance law

Amdahl's Law: Amdahl's law is used to calculate the performance gain that can be obtained by improving some portion of a computer. It states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

- Speed up (Performance improvement):

  It tells us how much faster a task can be executed using the machine with the enhancement as compare to the original machine. It is defined as

$$\text{Speed up} = \frac{\text{Performance for entire task using improved machine}}{\text{Performance for entire task using old machine}}$$

① A 40-MHz Processor was used to execute a benchmark program with the following instruction mix and clock cycle counts:

Instruction type	Instruction count	clock cycle count
Integer arithmetic	45000	1
Data transfer	32000	2
Floating point	15000	2
control transfer	8000	2

Determine the effective CPI, MIPS rate, and execution time for this program.

Solution:

Total clock cycles: $45000 \times 1 + 32000 \times 2 + 15000 \times 2 + 8000 \times 2$

$= 1,55,000$

Total No. of Instruction $= 45000 + 32000 + 15000 + 8000$

$I_c = 100000$

$$\text{Effective CPI} = \frac{1,55,000}{1,00,000} = 1.55 \text{ cycles/ins.}$$

$$\text{Execution time} = \frac{1,55,000}{40 \times 10^6} = \frac{\text{Total clock cycles}}{\text{clock period}}$$

$$= 3.87 \text{ ms.}$$

$$\text{MIPS Rate} = \frac{f}{\text{CPI} \times 10^6} = \frac{40 \times 10^6}{1.55 \times 10^6} = 25.8 \text{ MIPS}$$

Problem 1.4 : Consider the execution of an object-code with 200,000 instructions on a 40-MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles (CPI) needed for each instruction type are given below based on the result of a program trace experiment.

Instruction type	CPI	Instruction Mix
Arithmetic & logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache Miss	8	10%

a- Calculate the average CPI when the program is executed on a uniprocessor with the above trace result.

b. Calculate the corresponding MIPS rate based on the CPI obtained in part(a)

Solution :

(a) Arithmetic and logic = 1 X 60% X 200,000 = 120,000

Load / Store with cache hit = 2 × 18% × 200,000

= 72,000

Branch = 4 × 12% × 200,000

= 96,000

Memory reference with cache miss = 8 × 10% × 200,000

= 160,000

(c) Total: 448000

$$CPI = \frac{C}{I_c} = \frac{448000}{200,000} = 2.24$$

(b) $$\text{MIPS Rate} = \frac{f}{CPI \times 10^6} = \frac{40 \times 10^6}{2.24 \times 10^6} =$$

= 17.86

LECTURE

Pipelining: Pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.

A pipeline can be visualized as a collection of processing segments through which a binary information flows. Each segment performs partial processing dictated by the way the task is partioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments.

The name "pipeline" implies a flow of information analogous to an industrial assembly line. It is characteristic of pipeline that several computations can be in progress in distinct segments at the same time.

Example:

$A_i * B_i + C_i$ for $i = 1, 2, 3, ---7$

$R1 \leftarrow A_i$, $R2 \leftarrow B_i$ Input A_i and B_i

$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$ Multiply & Input C_i

$R5 \leftarrow R3 + R4$ Add C_i to product

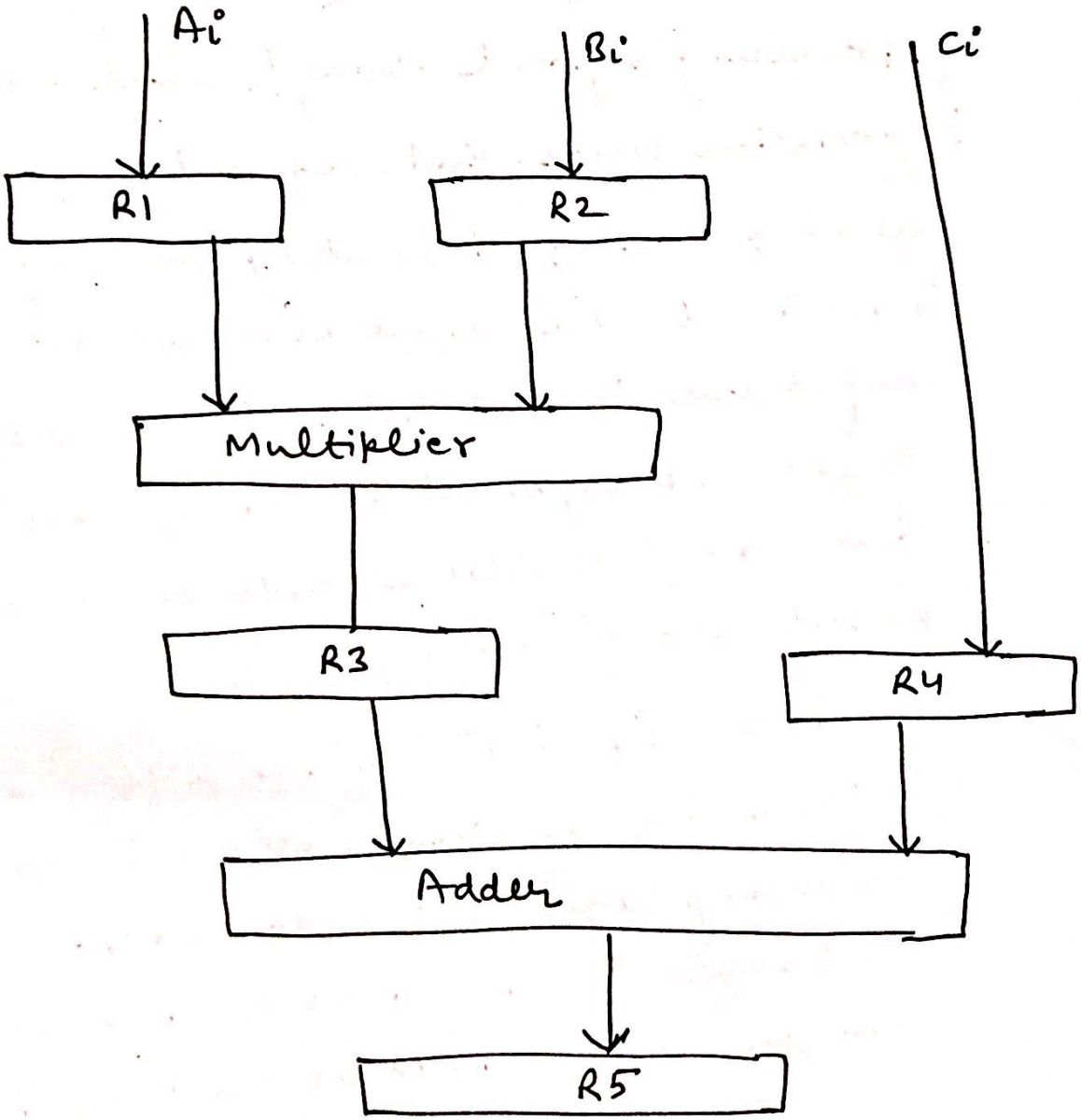
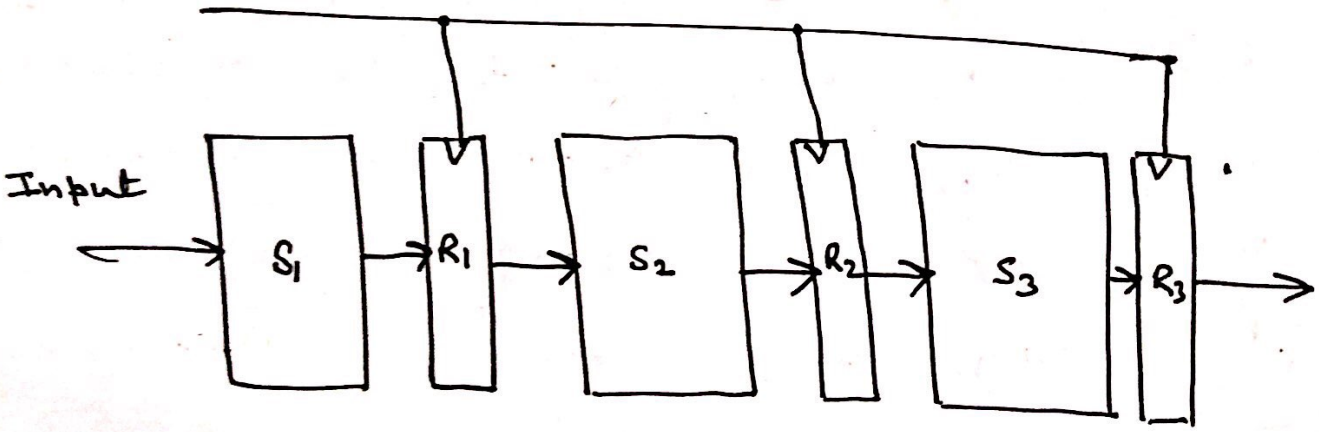


Table: Content of Registers in pipeline example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	–	–	–
2	A_2	B_2	$A_1 * B_1$	C_1	–
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8			$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9			–	–	$A_7 * B_7 + C_7$

General considerations

Any operation that can be decomposed into a sequence of suboperations of about the same complexity can be implemented by a pipeline processor. The technique is efficient for those applications that need to repeat the same task many times with different sets of data. The general structure of a four-segment pipeline is illustrated in figure.



Four-segment pipeline

Each segment consists of a combinational circuit S_i that performs a suboperation over the data stream flowing through the pipe. The segments are separated

information flows between adjacent [illegible]
the control of a common clock applied to all the registers simultaneously. We define a task as the total operation performed going through all the segments in the pipeline.

Space-time diagram:

The behaviour of a pipeline can be illustrated with space-time diagram. This space diagram shows the segment utilization as a function of time.

Segment	1	2	3	4	5	6	7	8	9	→ clock cycles
1	T_1	T_2	T_3	T_4	T_5	T_6				
2		T_1	T_2	T_3	T_4	T_5	T_6			
3			T_1	T_2	T_3	T_4	T_5	T_6		
4				T_1	T_2	T_3	T_4	T_5	T_6	

Space-time diagram for pipeline

Speed Up: A nonpipeline unit that performs the same operation and takes a time equal to t_n to complete each task. The total time required for n tasks is nt_n. The speed up of a pipeline processing over an equivalent nonpipeline processing is defined by the ratio

$$S = \frac{nt_n}{(K+n-1)t_p}$$

As the number of tasks increases, n becomes much larger than $K-1$, and $K+n-1$ approaches the value of n. Under this condition, the speed up becomes

$$S = \frac{t_n}{t_p}$$

if we assume that the time it takes to process a task is the same in the pipeline and nonpipeline circuits, we will have $t_n = Kt_p$

$$S = \frac{Kt_p}{t_p} = K$$

Types of Pipeline : The two types of pipelines are :

1). Arithmetic pipeline

2). Instruction pipeline

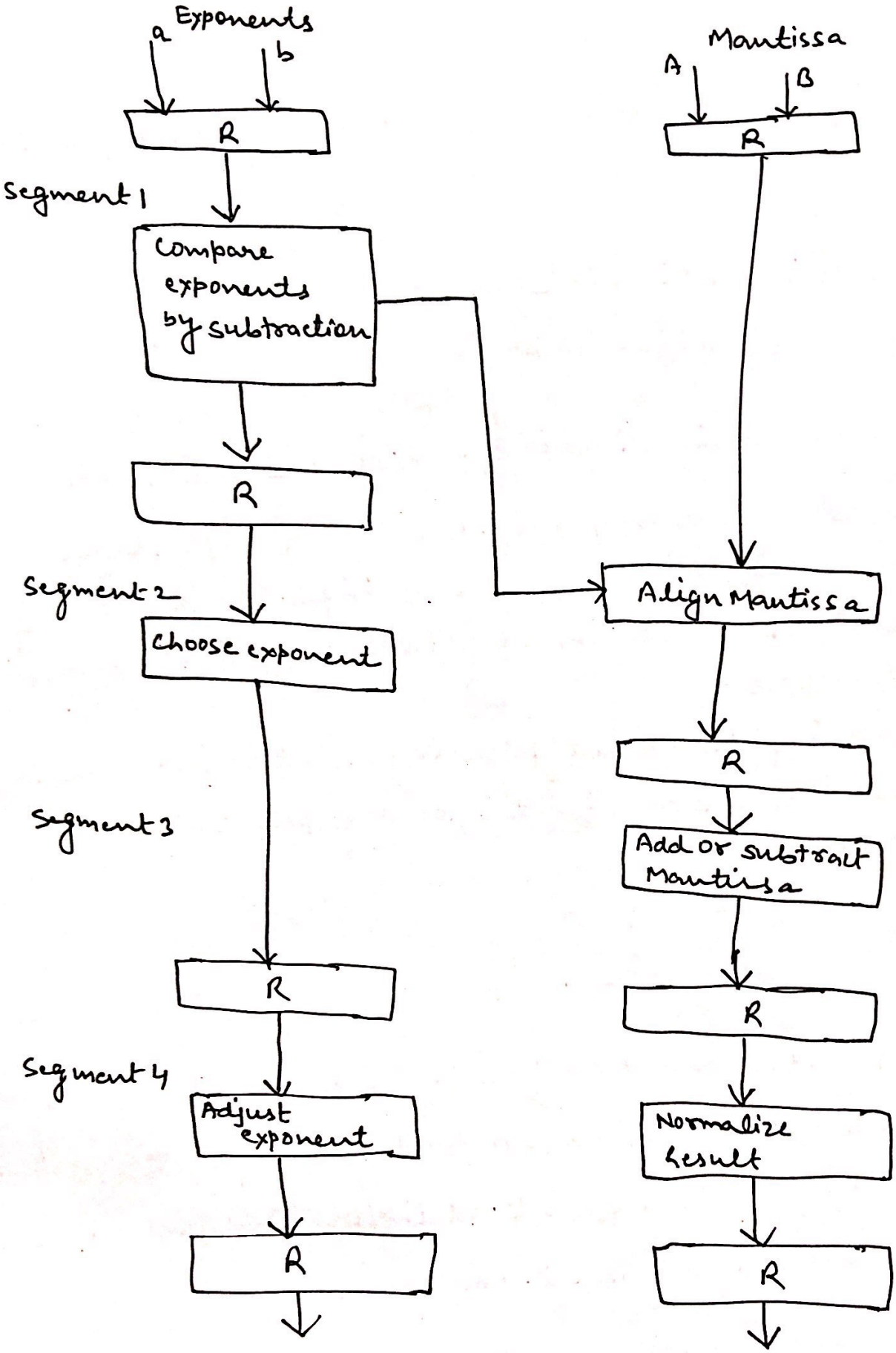
1. Arithmetic Pipeline : Pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating point operations, multiplication of fixed-point numbers.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers.

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

A and B are two fractions that represents the mantissa and, a and b are the exponents. The floating point addition and subtraction can be performed in four segments.



Pipeline for floating-point Addition and subtraction

The registers labeled R are placed between the segments to store intermediate results. The sub-operations that are performed in the four segments are:

1- Compare the exponents.
2- Align the mantissa
3- Add or subtract the mantissa
4- Normalize the result.

Example:

$$X = 0.9504 \times 10^3$$

$$Y = 0.8200 \times 10^2$$

The two exponents are subtracted in the first segment to obtain $3-2=1$. The larger exponent 3 is chosen as the exponent of the result. The next segment shifts the mantissa of Y to the right to obtain

$$X = 0.9504 \times 10^3$$

$$Y = 0.0820 \times 10^3$$

This aligns the two mantissa under the same exponent.

$$Z = 1.0324 \times 10^3$$

After Normalization

$$\boxed{Z = 0.10324 \times 10^4}$$

2- Instruction Pipeline :

data stream
. but in the

Pipeline processing can occur not only in the instruction stream as well. An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments.

In the most general case, the computer needs to process each instruction with the following sequence of steps:

1- Fetch the instructions from memory
2- Decode the instruction
3- Calculate the effective address
4- Feth the operands from memory
5- Execute the instruction
6- Store the result in the proper place.

The design of an instruction pipeline will be most efficient- if the instruction cycle is divided into segments of equal duration.

The operation of the instruction pipeline is shown in figure. The time in the horizontal axis is divided into steps of equal duration. the four segments are represented in the diagram with an abbreviated symbols.

1- FI is the segment that fetches an instruction

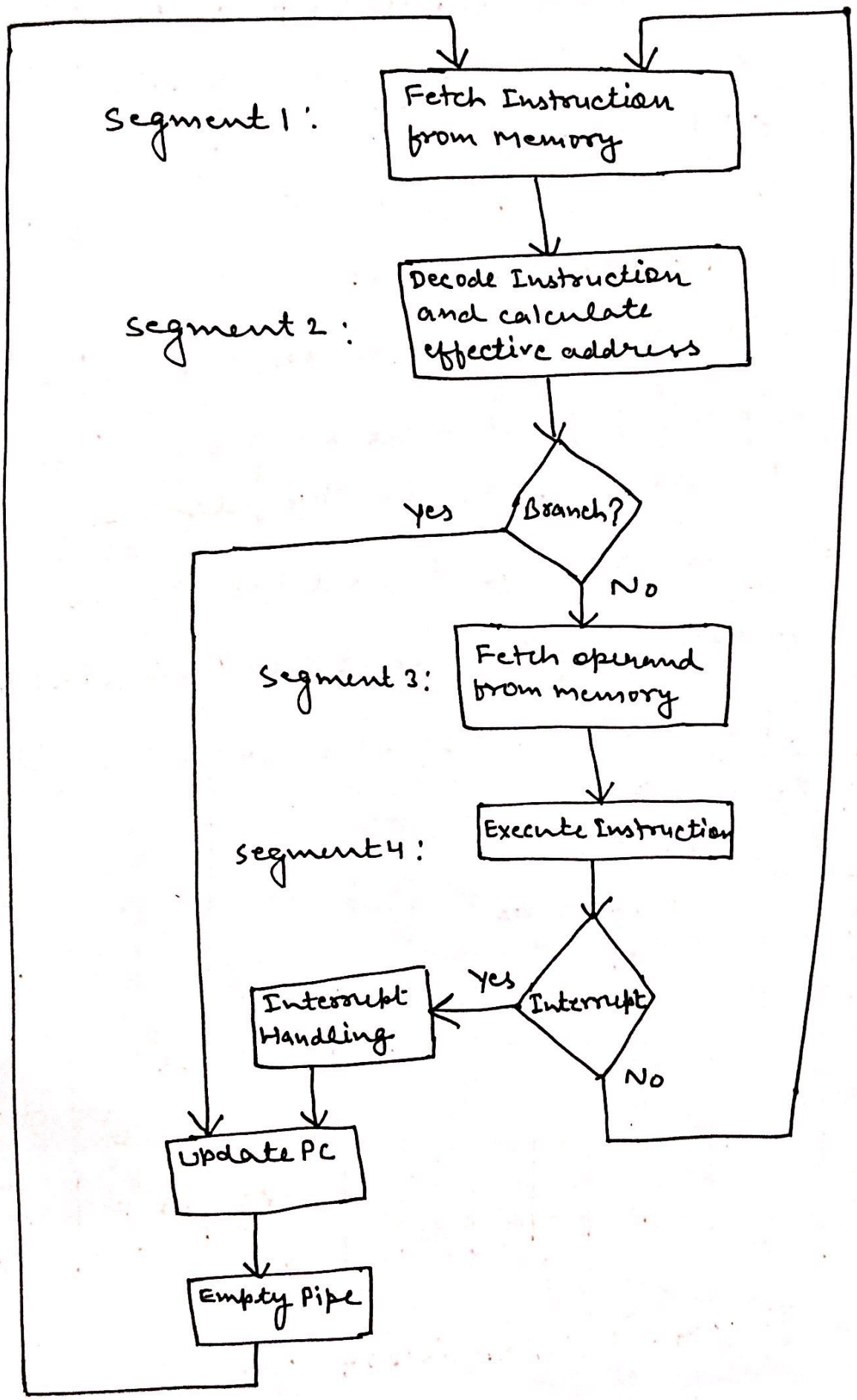
2- DA is the segment that decodes the instruction and calculates the effective address.

3- FO is the segment that fetches the operand

4- EX is the segment that executes the instruction

Step / Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
Branch 3			FI	DA	FO	EX							
4				FI	-	-	FI	DA	FO	EX			
5								FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Timing of Instruction Pipeline



Four-segment CPU pipeline

Linear Pipeline Processors:

A linear pipeline processor is a cascade of processing stages which are linearly connected to perform a fixed function over a stream of data flowing from one end to the other.

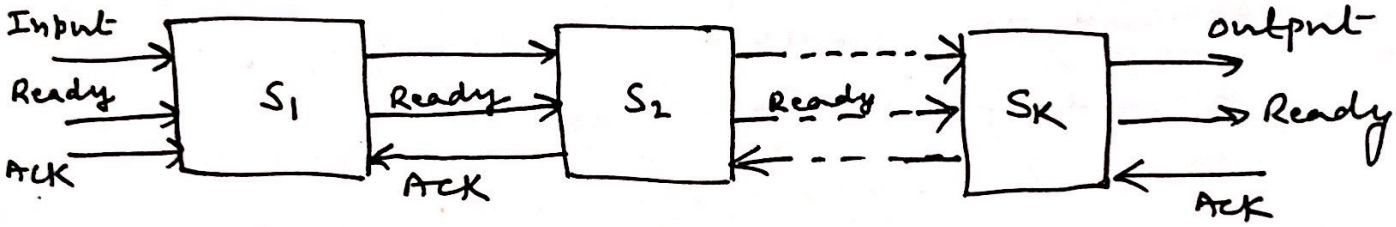
There are two types of linear pipeline processors:

Asynchronous and Synchronous Models:

A linear pipeline processor is constructed with k processing stages. External inputs (operands) are fed into pipline at the first stage S_1. The processed results are passed from stage S_i to stage S_{i+1}, for all $i = 1, 2, 3, - - - k-1$. The final result emerges from the pipeline at the last stage S_k.

Depending on the control of data flow along the pipeline, we model linear pipelines in two categories: asynchronous and synchronous.

Asynchronous Model: As shown in figure data flow between adjacent stages in an asynchronous pipeline is controlled by a handshaking protocol. When stage S_i is ready to transmit, it sends a ready signal to stage S_{i+1}. After stage S_{i+1} receives the incoming data, it returns an acknowledge signal to S_i.



An Asynchronous pipeline Model

Synchronous Model :

Synchronous pipelines are illustrated in figure. clocked latches are used to interface between stages. The latches are made with master-slave flip-flops, which can isolate inputs from outputs. upon the arrival of a clock pulse, all latches toansfer data to the next stage simultaneously.

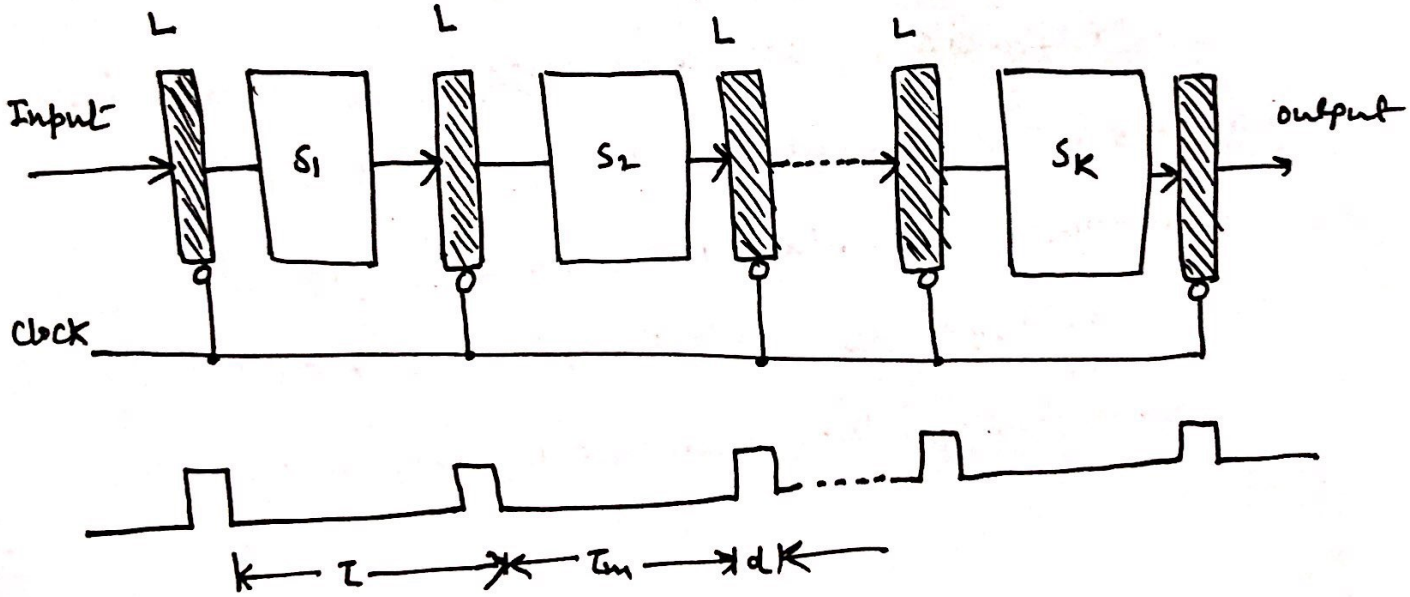


Fig: A Synchronous pipeline Model

Caption : S_i = Stage i | T_m = Maximum Stage Delay
L = Latch | d = Latch delay
τ = clock period | Ack = Acknowledge Signal

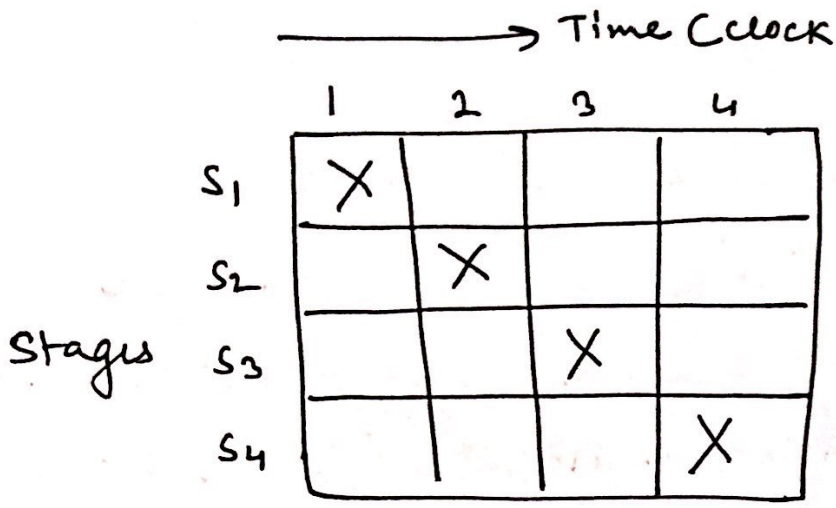


Fig: Reservation table of a four-stage linear pipeline

The utilization pattern of successive stages in a synchronous pipeline is specified by a reservation table. For a linear pipeline, the utilization follows the diagonal streamline pattern shown in above figure.

Clocking and Timing Control :

The clock cycle τ of a pipeline is determined below.

Let τ_i be the time delay of the circuitry in stage S_i and d the time delay of a latch.

Clock cycle and Throughput : Denote the maximum stage delay as τ_m and we can write τ as

$$\tau = \max_i \{\tau_i\}_1^K + d = \tau_m + d \quad \text{——(1)}$$

pipeline frequency is defined as the inverse of the clock period :

$$\boxed{f = \frac{1}{\tau}} \quad \text{——(2)}$$

Clock skewing : Ideally, we expect the clock pulses to arrive at all stages (latches) at the same time. However, due to a problem known as clock skewing, the same clock pulse may arrive at different stages with a time offset of s.

When clock skew takes effect

$$d + t_{max} + s \leq \tau \leq \tau_m + \tau_{min} - s \quad \text{——(3)}$$

in the ideal case $s=0$, $t_{max} = \tau_m$ and $\tau_{min} = d$

$$\boxed{\tau = \tau_m + d}$$

Speedup, Efficiency and Throughput:

a linear pipeline of k stages can process n tasks in $k+(n-1)$ clock cycles. The amount of time it takes to execute n tasks on this non-pipelined processor is $T_1 = nk\tau$

$$T_k = [k+(n-1)]\tau \quad \text{——— (4)}$$

Speedup: The speedup factor of a k-stage pipeline over an equivalent non-pipelined processor is defined as

$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{k\tau+(n-1)\tau} = \frac{nk}{k+(n-1)} \quad \text{— (5)}$$

Efficiency & Throughput: The efficiency E_k of a linear k-stage pipeline is defined as

$$E_k = \frac{S_k}{k} = \frac{n}{k+(n-1)}$$

The pipeline throughput H_k is defined as the no. of tasks (operations) performed per unit time.

$$H_k = \frac{n}{[k+(n-1)]\tau} = \frac{nf}{k+(n-1)}$$

Q: Consider the execution of a program of 15000 instructions by a linear pipeline processor with a clock rate of 25 MHz. Assume that the instruction pipeline has five stages and that one instruction is issued per clock cycle. The penalties due to branch instructions and out-of-sequence executions are ignored.

(a) calculate the speedup factor in using this pipeline to execute the program as compared with the use of an equivalent nonpipelined processor with an equal amount of flow-through delay.

(b) what are the efficiency and throughput of this pipelined processor.

Sol:

$n = 15000$ instructions.

$f = 25$ MHz

$K = 5$ stages

1 - issued processor

Speedup $(S_K) = \frac{T_1}{T_K} = \frac{nK\tau}{K\tau + (n-1)\tau}$

$= \frac{nK}{K+(n-1)} = \frac{15000 \times 5}{5 + (15000 - 1)}$

$= \frac{75000}{15004} = 4.999$

$S_K = 4.999$

Throughput $(H_K) = \frac{nf}{K + (n-1)}$

$= \frac{15000 \times 25}{5 + (15000 - 1)}$

$= \frac{3,75,000}{15,004}$

$H_K = 24.99$ MIPS

Efficiency $(E_K) = \frac{S_K}{K} = \frac{4.999}{5} = 0.9998$

$= $ [illegible]

$E_K = 0.9998$

Nonlinear Pipeline Processors

A dynamic pipeline can be reconfigured to perform variable functions at different times. The traditional linear pipelines are static pipilines because they are used to perform fixed functions. A dynamic pipeline allows feedforward and feedback connections in addition to the streamline connections. For this reason, we call such a structure a nonlinear pipeline.

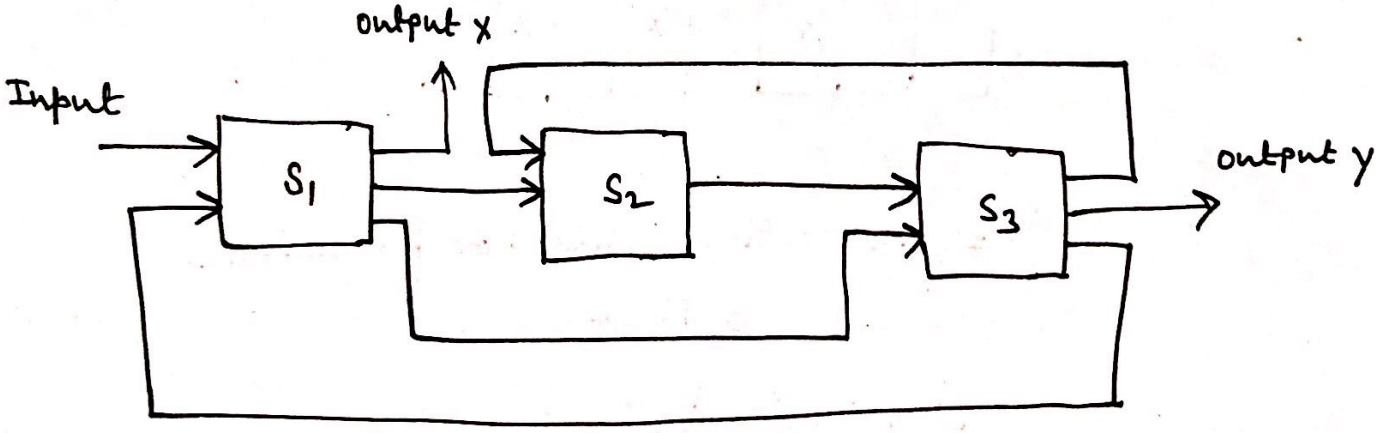
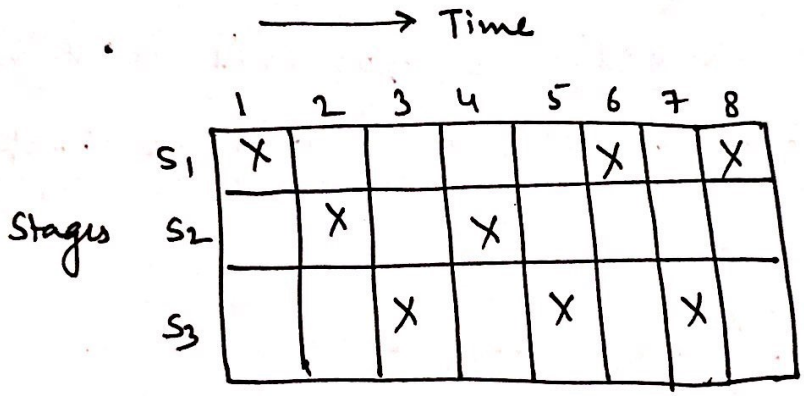
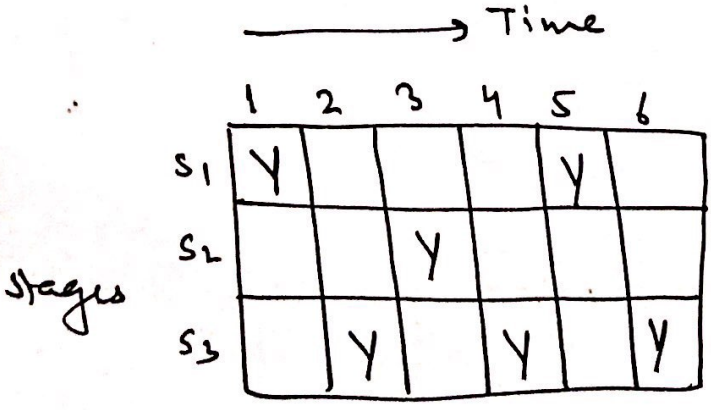


Fig: @ A three-stage pipeline

A multifunction dynamic pipeline is shown in fig.(a). This pipeline has three stages. Besides the streamline connections from S_1 to S_2 and from S_2 to S_3, there is a feedforward connection from S_1 to S_3 and two feedback connections from S_3 to S_2 and from S_3 to S_1.

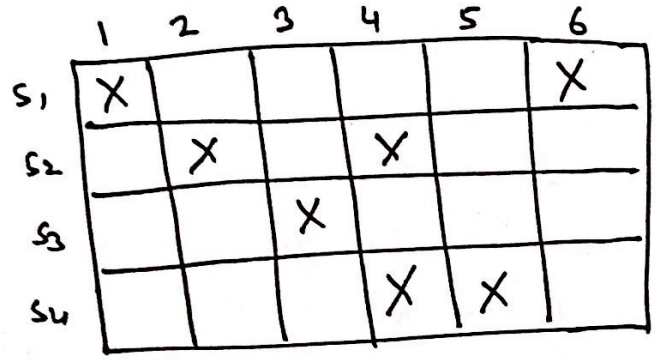


Fig(b) Reservation Table for function X



Fig(c) Reservation Table for function Y

6.6 : Consider the following reservation table for a four-stage pipeline with a clock cycle $\tau = 20$ ns



(a) What are the forbidden latencies and the initial collision vector.

(b) Draw the state transition diagram for scheduling the pipeline.

(c) Determine the MAL associated with the shortest greedy cycle

(d) Determine the pipeline throughput corresponding to the MAL and given τ.

Solution : (a) The Forbidden latencies are

$\{(6-1), (4-2), (5-4)\} = \{5, 2, 1\}$

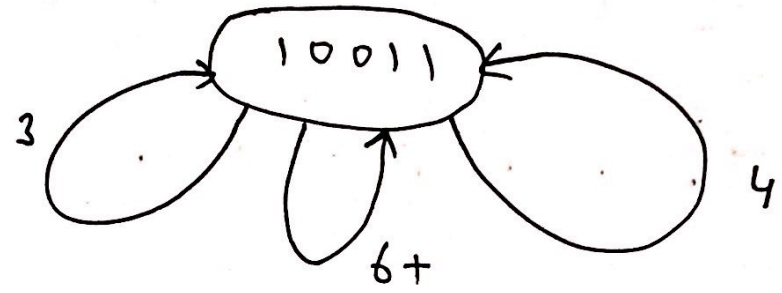
so $\{1, 2, 5\}$

permissible latencies = { 3, 4 }

Collision vectors = { C_5, C_4, C_3, C_2, C_1 }

= 1 0 0 1 1

(b)



(c)

simple cycle	average cycle
3	3
4	4
6	6

greedy cycle = 3

MAL = 3

(d)

① Throughput = $\frac{1}{MAL} = \frac{1}{3} = 0.33$

② The τ is $\frac{1}{2} = 0.5$

6.9

	1	2	3	4
S_1	X			X
S_2		X		
S_3			X	

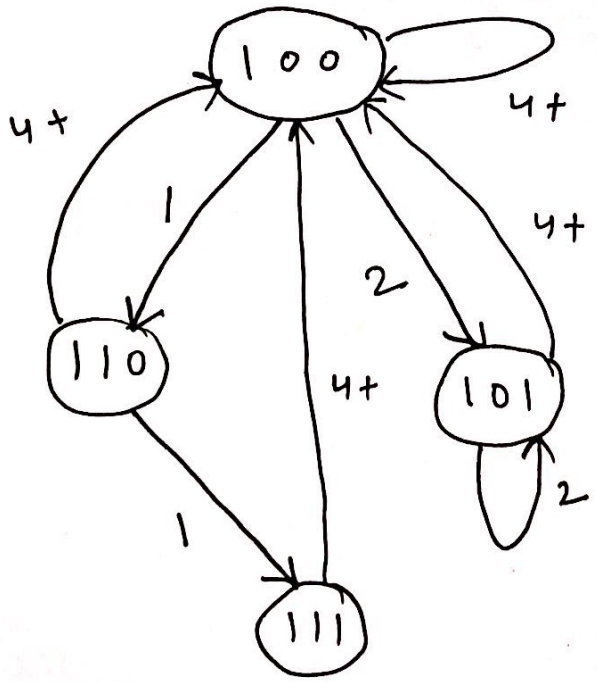
(a) Forbidden latencies = {(4-1)} = {3}

permissible latencies = {1, 2}

collision vector = { C_3, C_2, C_1 }

{ 1 0 0 }

(b)



State diagram

(c)

simple cycle	average cycle
(1,4)	2.5
4	4
2	2
(2,4)	3
(1,1,4)	2

≤ 1

greedy cycle = (2), (1,4)

MAL = 2

RISC	CISC
1- RISC stands for Reduced Instruction set computer	1- CISC stands for complex Instruction set computer
2- RISC processors have simple instructions taking about one cycle.	2- CISC processor has complex instructions that take up multiple clocks for execution.
3- Performance is optimized with more focus on software.	3- Performance is optimized with more focus on hardware.
4- It has no memory unit & uses a separate hardware to implement instructions.	4- It has a memory unit to implement complex instructions.
5- It has a hard-wired unit of programming.	5- It has a microprogrammed unit.
6- RISC processors are highly pipelined	6- They are normally not pipelined or less pipelined.
7- Execution time is very less.	7- Execution time is very high.
8- It does not require external memory for calculations	8- It requires external memory for calculations.