Computer Architecture module 5

Computer Organization & Architecture (Mahatma Gandhi University)

Module 5

PIPELINING AND VECTOR PROCESSING

Module 5: Pipelining and Vector processing: Introduction to parallel processing, Pipeline computers, Array Processors, Multiprocessing systems, Architectural classification scheme-SISD, SIMD, MISD, MIMD, Introduction to pipelining, Instruction and Arithmetic pipelines (design), Vector processing.

INTRODUCTION TO PARALLEL PROCESSING: -

- ✓ Parallel processing is an efficient form of information Processing which emphasizes the ***execution of concurrent events.***
- ✓ i.e., parallel processing demand ***simultaneous execution of many programs in the computer.***
- ✓ ***Parallel computers have those systems that emphasize parallel processing.***
- ✓ They can be divided into three architecture configurations.
    1. pipeline computers
    2. Array processors
    3. Multiprocessor systems.
- ✓ **Pipeline computer**s perform **overlapped computation** to achieve parallelism.
- ✓ **Array processors** use **multiple synchronized ALU's** to achieve parallelism.
- ✓ **Multiprocessor systems** achieve parallelism through a set of interactive processors with shared resources (**multiple processors**).

PIPELINE COMPUTERS: -

- ✓ The process of **executing an instruction** in a digital computer involves four Major steps.
    1. Instruction fetch (IF)→ from main memory
    2. Instruction Decoding (ID) → identify the operation to be performed

3. Operand Fetch (OF) → retrieving operands from main memory.
4. Execution (EX) → Execution of the decoded operation.

✓ In a non-pipelined computer, these four steps must be computed before the next instruction can be issued.

✓ In pipeline computer, successive instructions are executed in an overlapped fashion. Four pipeline stages are arranged into a linear cascade.

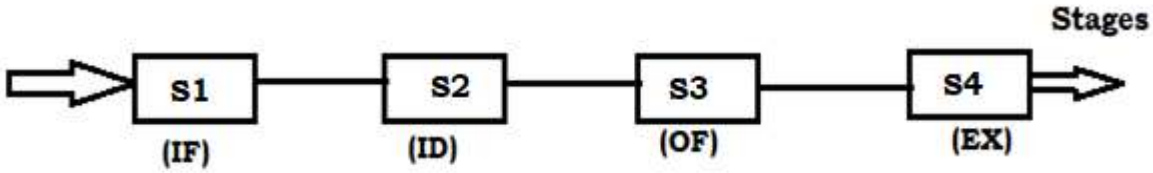


Figure: Pipelined Processor 4 stages

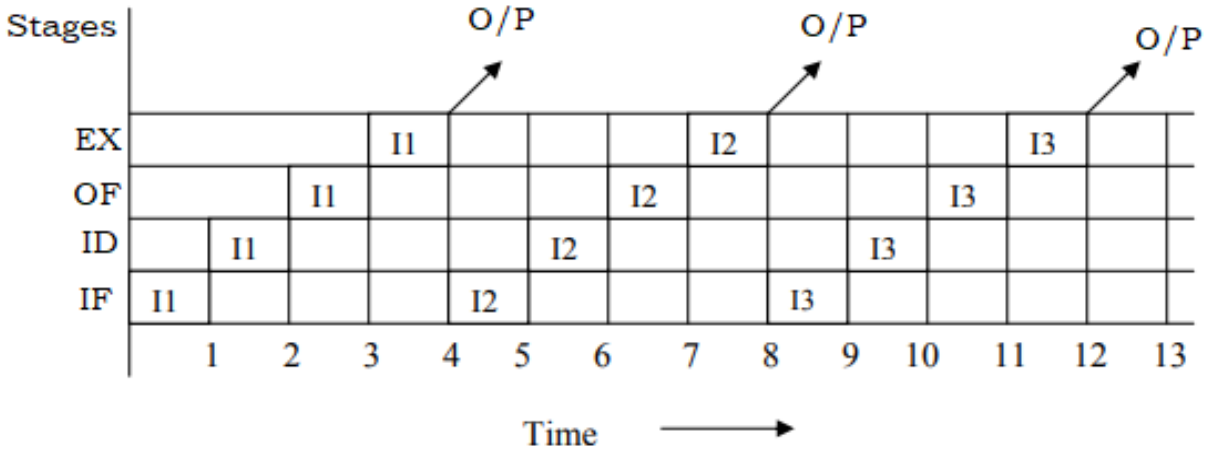
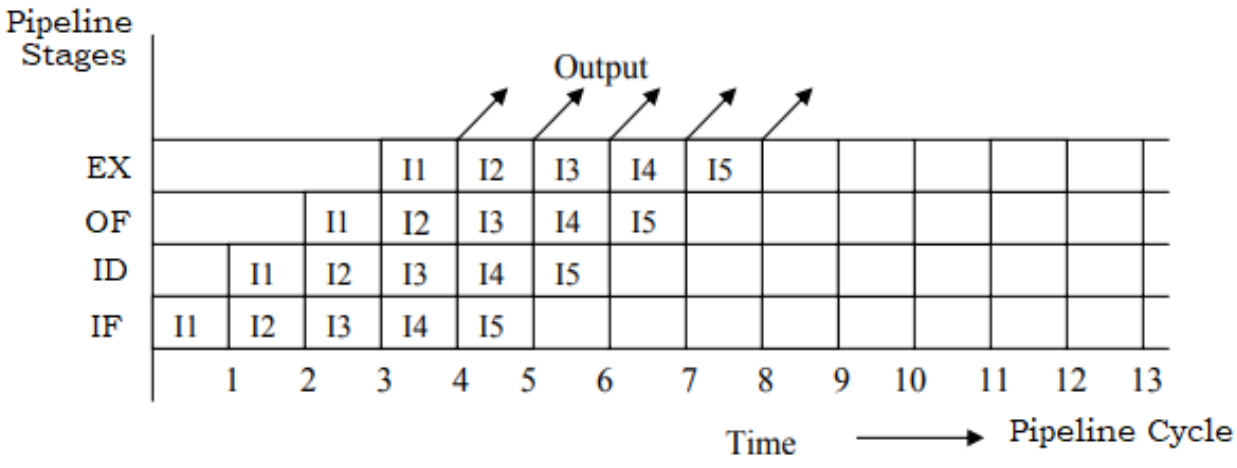


Figure Space-time diagram for Non-pipelined Processor



Space-time diagram for Overlapped Instruction pipelined Processor

- ✓ ***An instruction cycle consists of multiple pipeline cycles***.
- ✓ A ***pipeline can be set equal to the delay of the slowest stage***.
- ✓ The flow of data from stage to stage is triggered by a common clock of the pipeline.
- ✓ Interface latches are used between adjacent segments to hold the Intermediate results.
- ✓ For the non-pipelined computers, 4 pipeline cycles are needed to complete one instruction. But for pipelined computers, the result is it is produced from the adjacent pipeline cycles.
- ✓ Some main issues in designing a pipeline computer include ***job sequencing, collision prevention, congestion control, etc.***
- ✓ Pipeline computer are more of suitable for **vector processing**.

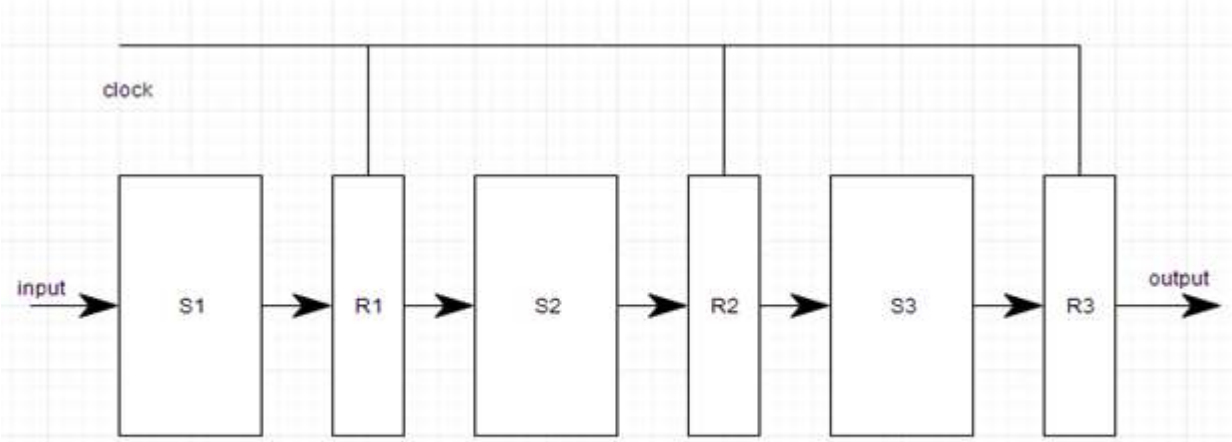


Figure:General structure of a 4 segment pipeline

ARRAY PROCESSORS: -

- ✓ An array processor is a synchronous parallel computer with multiple ALU's called ***processing elements*** (PEs) that can be operated in parallel.
- ✓ The PEs are synchronous to perform same functions at the same time.
- ✓ They are ***passive devices***.
- ✓ An appropriate data routing mechanism must be established among the PEs.
- ✓ Array processors are ***more suitable for vector processing.***

✓ A typical array processor is shown in the figure.

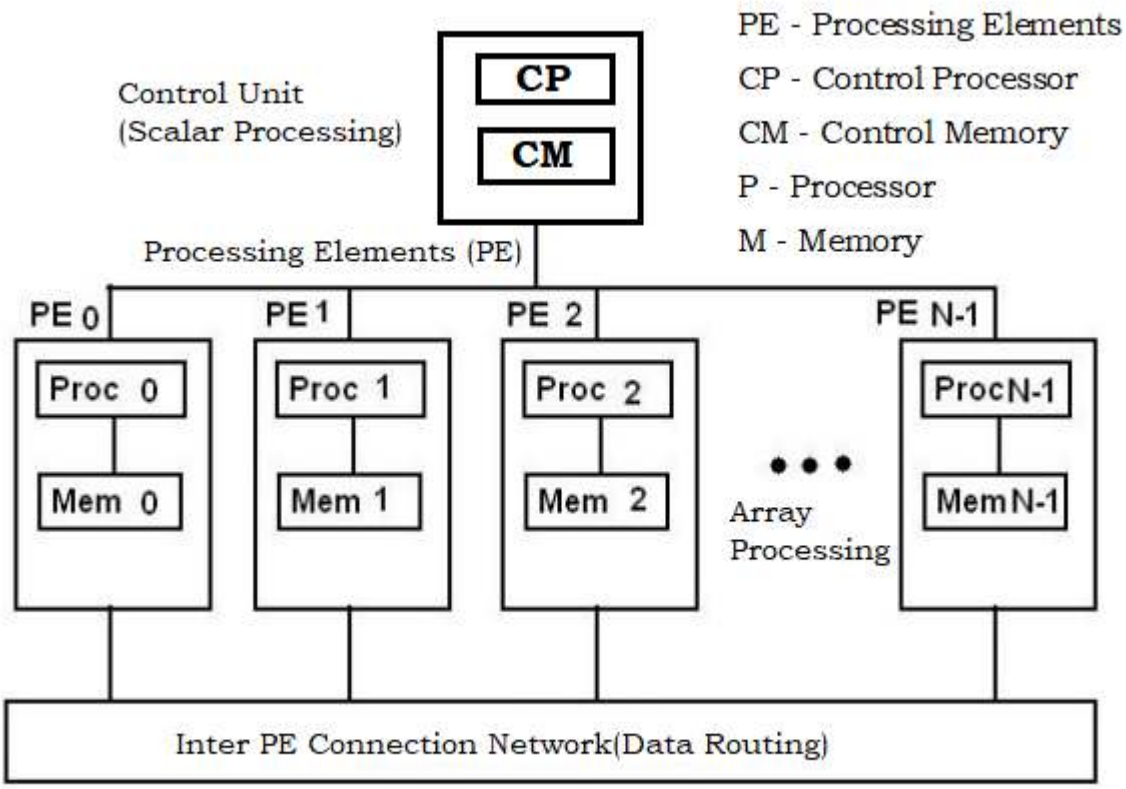


Figure: Functional Structure of an array processor with concurrent scalar processing in the control unit

MULTIPROCESSOR SYSTEMS: -

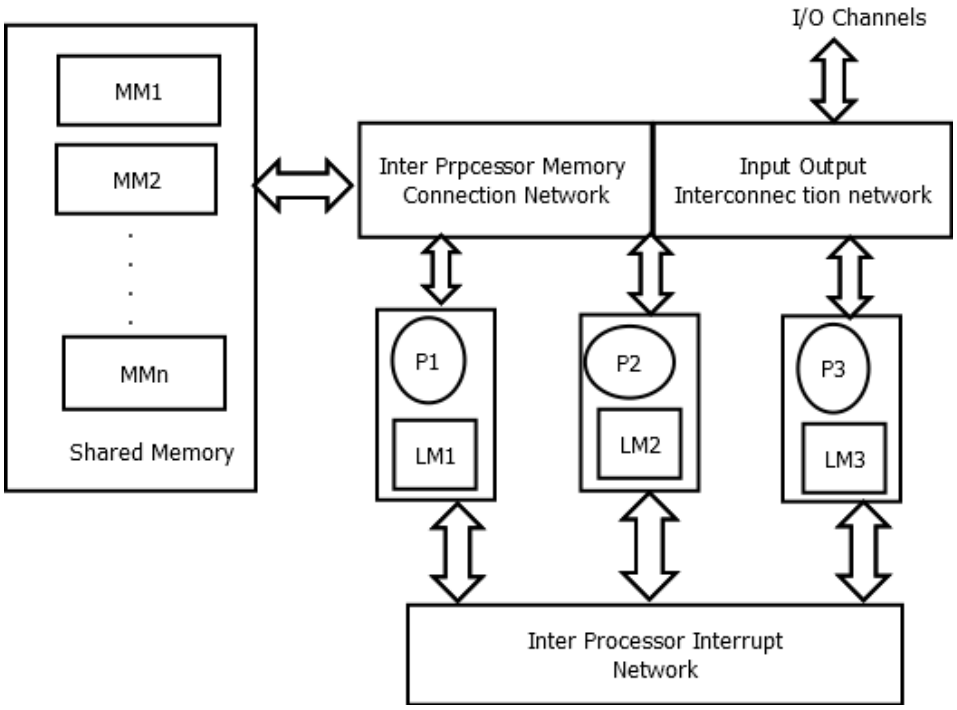
✓ A multiple processor system contains two or more processor of approximately comparable capabilities.

✓ All processors share access to common set of memory modules, I/O channels and peripheral devices.

✓ The entire system must be controlled by ***a single integrated operating system*** providing interaction between processors and their programs at the various levels.

✓ Inter processor communication can be done through the shared memories or through the interrupt network.

✓ A basic multiprocessor organisation is shown in figure.



- ✓ Multiprocessor hardware system organisation is determined by the interconnection structure to be used between the memories and the processors.
  - Three of them are time shared common bus, cross-bar switch network and multiport memories.

ARCHITECTURAL CLASSIFICATION SCHEMES

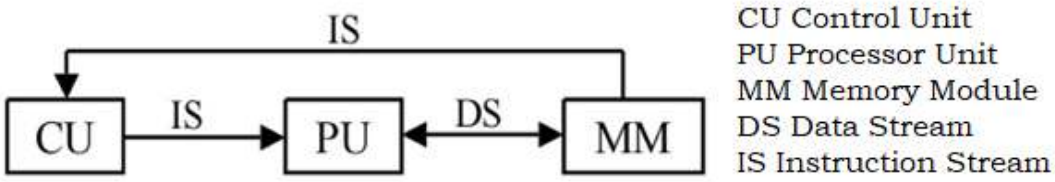
Flynn's classification.

- ✓ Flynn's classification is ***based on the multiplicity of instruction streams and data streams in the computer system.***
  - An instruction stream is a sequence of instructions and data stream is a sequence of data.
- ✓ According to Flynn's classification, digital computers may be classified into four categories
  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)
- ✓ Both instructions and the data fetched from the memory modules.

- ✓ Instructions are decoded by the CU, which sends the decoded instructions stream to the processor unit for the execution.
- ✓ Data stream flows between the processor and the memory bidirectionally.

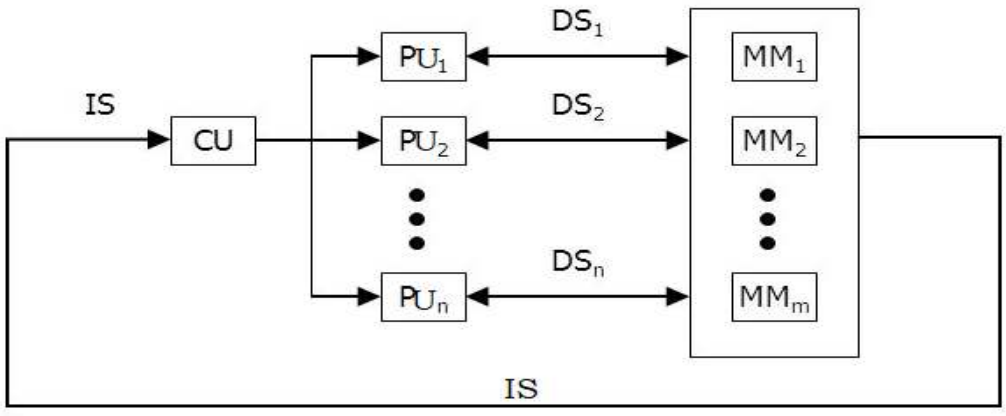
1. <u>SISD (Single Instruction Stream Single Data Stream):-</u>

- ✓ Most of the serial computers use this organisation.
- ✓ Instructions are executed sequentially but maybe overlapped in their execution stages (pipelining).
- ✓ An SISD computers may have more than one functional unit.
- ✓ All the functional units are under the supervision of the one CU.
- ✓ Example: pipeline computers.



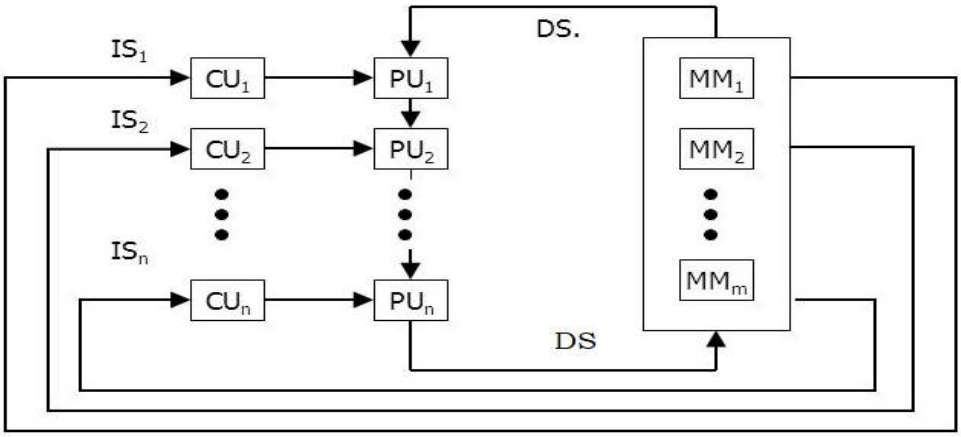
2. <u>SIMD (Single Instruction Stream Multiple Data Stream):-</u>

- ✓ Here multiple processing element (PEs) are supervised by the same CU.
- ✓ All the PEs receive the same instruction broadcast from the CU ; but operate on different datasets from the distinct data streams.
- ✓ The shared memory may contain multiple modules.
- ✓ SIMD machines are further divided into word slice and bit slice modules.
- ✓ Example: Array processors.



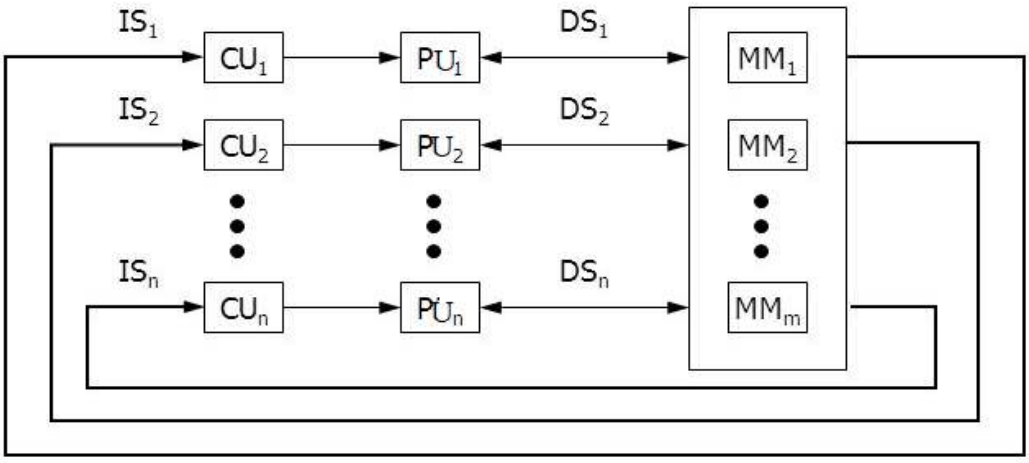
3. MISD (Multiple Instruction Stream Single Data Stream):-

- ✓ There are 'n' processor units, each receive distinct instruction operating over the same data stream and its derivations.
- ✓ The result of one processor are the Inputs of the next to processor.
- ✓ No real embodiments of this class exist.



4. MIMD (Multiple Instruction Stream Multiple Data Stream):-

- MIMD organisation refers to the computer system capable of processing several programs at the same time.
- MIMD computers fall into
  - **Tightly coupled system** - If the degree of interaction among the processor is high.
  - **Loosely coupled system** - If the degree of the interaction among the processor is low.
- Example: Multiprocessor System



INTRODUCTION TO PIPELINING

- ✓ Pipelining is a technique of ***decomposing a sequential process into sub process with each sub process being executed and concurrently with each other.***
- ✓ Each sub process is being executed in ***dedicated segment*** and each segment performs partial processing.
- ✓ The result obtained from 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 behaviour of the pipeline can be shown with the space-time diagram.

Segment:	1	2	3	4	5	6	7	8	9	Clock cycle →
1	T1	T2	T3	T4	T5	T6				
2		T1	T2	T3	T4	T5	T6			
3			T1	T2	T3	T4	T5	T6		
4				T1	T2	T3	T4	T5	T6	

Space-time diagram for pipeline.

- ✓ The horizontal axis displays time in clock cycle and the vertical axis gives the segment number.
- ✓ The diagram shows 6 tasks T_1 to T_6 executed in 4 segments. Initially task T_1 is in S_1. After the first clock, T_2 is in S_1 while T_1 in S_2. Continuing in this manner, T_1 is completed after 4^{th} cycle. Once the pipeline is full, it takes only 1 clock period to get output.
- ✓ There are two areas of the computer design, Where the pipelining is applicable.

1. Arithmetic pipeline

2. Instruction pipeline

1. Arithmetic pipeline: -

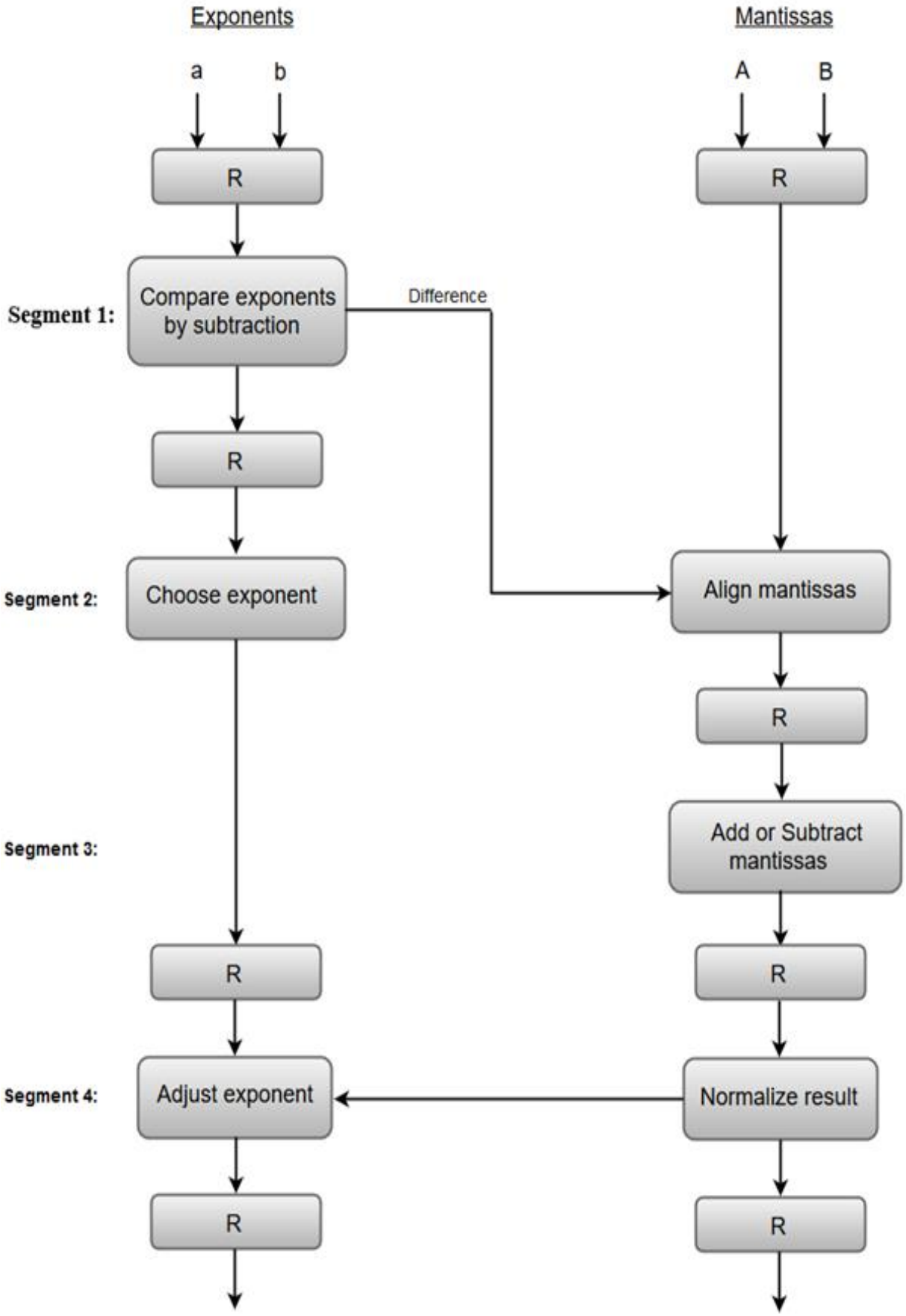
- ✓ Arithmetic pipeline is used to implement ***floating point operations***.
- ✓ Consider an example of a pipeline unit for floating point addition and subtraction.
- ✓ The input to the floating-point adder pipeline are 2 normalized floating-point binary numbers.

$$X=A \times 2^a$$ **(A and B are mantissas, a and b are exponents)**

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

- ✓ The floating-point addition and subtraction can be performed in four segments as,
    1. Compare the exponents
    2. Align the mantissas
    3. Add or subtract the mantissas
    4. Normalize the result

Pipeline organization for floating point addition and subtraction:



Consider 2 floating point numbers, X=950.4, Y=82

Normalized Form

$X=0.9504 \times 10^3$

$Y=0.82 \times 10^2$

Segment 1:

- ✓ Exponents are compared by subtraction.
- ✓ Here **a=3, b=2 a-b=3-2=1**

Segment 2:

- ✓ Choose exponent and align mantissa, here exponent is 3.
- ✓ This segment shifts the mantissa of Y to the right to obtain

$X=0.9504 \times 10^3$

$Y=0.082 \times 10^3$

- ✓ This aligns the two mantissas under the same exponent.

Segment 3:

- ✓ This segment adds the two mantissas to get the sum.

$X=0.9504 \times 10^3$

$Y=0.082 \times 10^3$

1.0324×10^3

Segment 4:

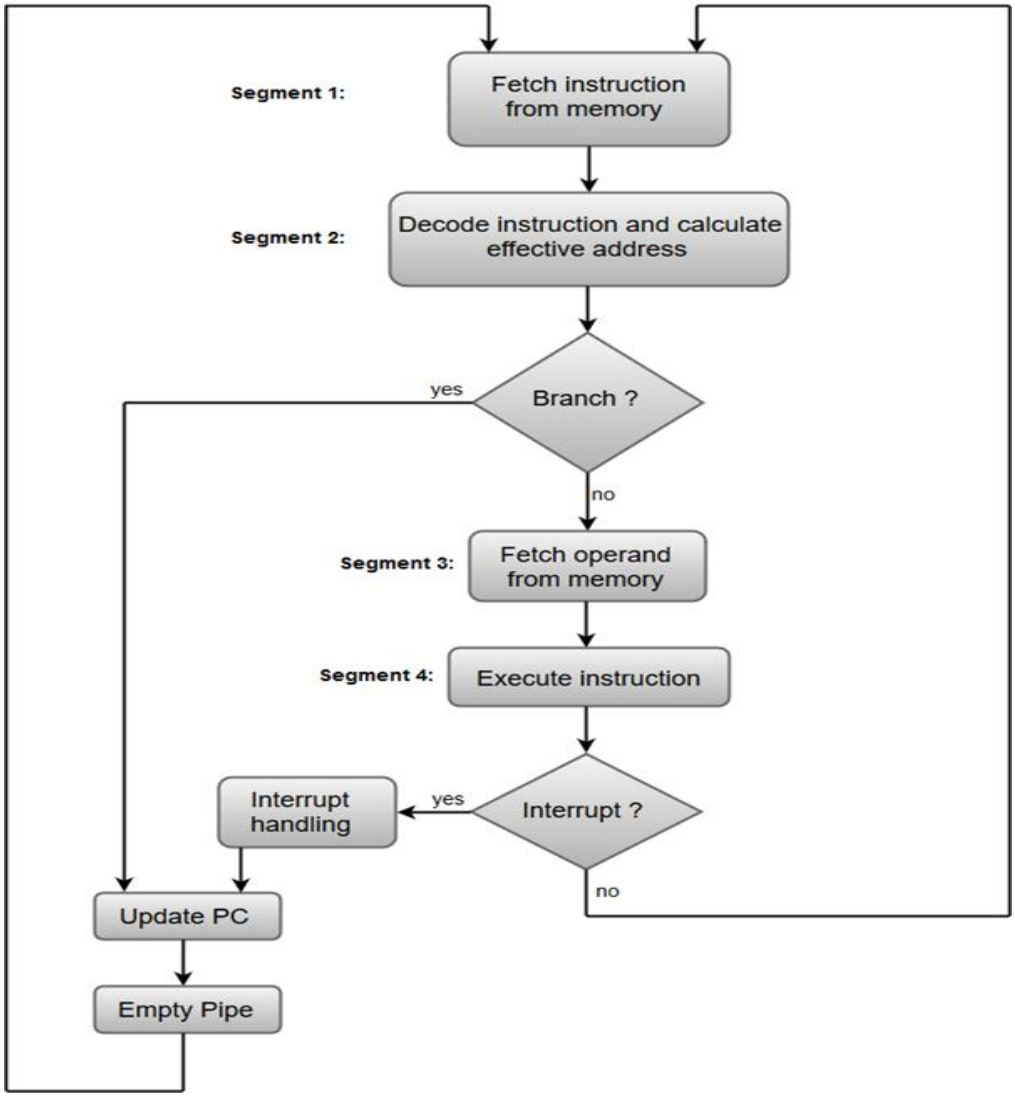
- ✓ The sum is adjusted to get the normalized result.
- ✓ This is done by shifting the mantissa once to the right and incrementing the exponent by 1 to obtain the normalized sum.

$Z=0.10324 \times 10^4$

2. Instruction Pipeline: -

- ✓ An instruction pipeline reads consecutive instructions from memory, while previous instructions are being executed in other segments.
- ✓ i.e., instruction ***fetch and execute*** phases to overlap and perform simultaneous operation.

- ✓ One special case is that, ***the branch instructions*** - in that case the pipeline must be emptied and all the instructions after the branch instruction must be discarded.
- ✓ ***Example: Four segment instruction pipeline.***
- ✓ In a 4-segment pipeline, the computer needs to process each instruction with the following sequence of the steps.
    1. **Fetch the instruction from memory.**
    2. **Decode the instruction.**
    3. **Fetch the operand from memory.**
    4. **Execute the instruction.**
- ✓ Figure shows how **instruction cycle** in the CPU can be processed with a 4-segment pipeline.



- ✓ Thus, up to **four** sub operations in the instruction cycle can overlap and up to four different instructions can be processed at the same time.
- ✓ In the case of branch instruction, the pending operations in the last two segments are completed and all the information stored in the instruction buffer is deleted.
- ✓ The pipeline then restarts from the new address stored in the PC.

[Note: include the space-time diagram]

- ✓ In general, there are **three major difficulties** that cause the instruction pipeline to deviate from its normal operation.
    1. **Resource conflict** - caused by access to memory by two segments at the same time.
    2. **Data dependencies** - conflict arises when an instruction, depends on the result of the previous instruction, but the result is not yet available.
    3. **Branch Difficulties** - Arises from the branch and other instruction that changes the value of the PC.

VECTOR PROCESSING: -

- ✓ The basic idea of vector processing is that **Same operation on different data.**
- ✓ The computational problems that required a vast number of computations that will take conventional computer, days or even weeks to complete follows the method of vector processing.
- ✓ **Applications of vector processing:** Example - long range weather forecasting, medical diagnosis, aero dynamics and space flight simulations, artificial intelligence and expert systems, image processing etc.
- ✓ **Vector processing uses mainly in two situations.**
    1. **Vector Operation**
    2. **Matrix Multiplication**

VECTOR OPERATION: -

- ✓ A vector is an ***ordered set of one-dimensional array of data items.***
- ✓ A vector **V** of length **n** is represented as row vector by,

V= [V1, V2, V3.....Vn]

- ✓ Consider the following code,

```
for (I=0; I<100; I++)
        C [ I ] = A [ I ] + B [ I ]
```

- ✓ These statements are used for adding two vectors A & B of length 100 to produce a vector C. This is implemented by the following sequence of operations.

Initialize I=0

LOOP: Read A[I]

Read B[I]

ADD A[I], B[I]

Store C[I]

Increment I; I=I+1

If I<100 go to LOOP

Else Stop.

- ✓ In conventional method of processing, the above steps may be executed 100 times.
- ✓ Vector processing method allows the operations to be specified with a single vector instruction of the form,

C (1:100) = A (1:100) + B (1:100)

- ✓ The vector instruction includes the initial address of the operands, the length of the vectors and operation to be performed all in one composite instruction.

✓ A possible instruction format of a vector instruction is shown in figure**:**

Operation Code	Base Address Source 1	Base Address Source 2	Base Address destination	Vector length
ADD	starting Address of A	starting Address of B	starting Address of C	100

✓ This is essentially a three-address instruction with three fields specifying the base address of the operands and an additional field that gives the length of the data item in the vector.

MATRIX MULTIPLICATION: -

✓ Matrix multiplication is one of the most computational operations performed in computer with vector processing.

✓ Consider the example, Multiplication of the 3X3 matrix A & B.

A X B = C

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

✓ The product matrix C is a 3X3 matrix whose elements are related to the elements of A & B by the inner product.

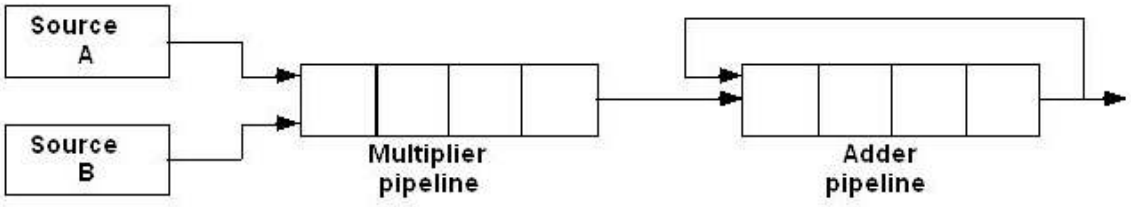
$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

✓ In general, the inner product consists of the sum of **'k'** products of the form

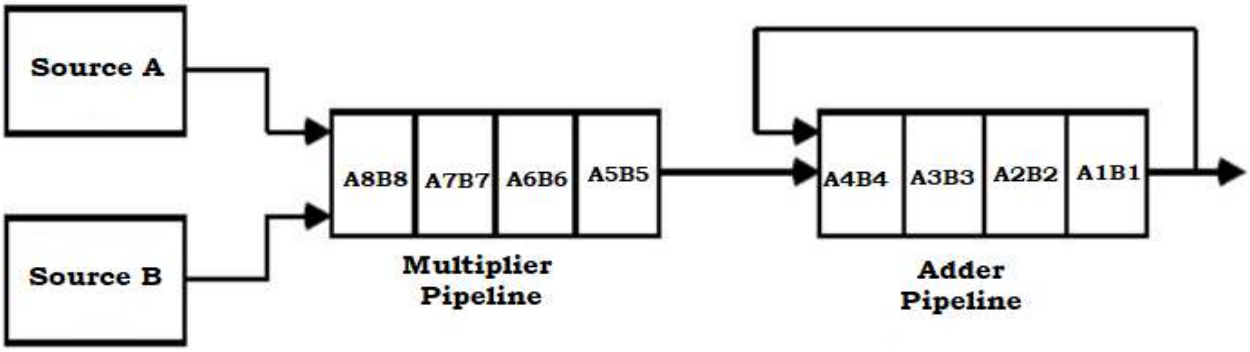
$$C=A_1B_1+A_2B_2+A_3B_3+\ldots\ldots\ldots\ldots\ldots\ldots+A_kB_k$$

✓ In a typical application **k** may be equal to 100 or 1000.

✓ The inner product calculation on a pipeline vector processor is shown in the figure.

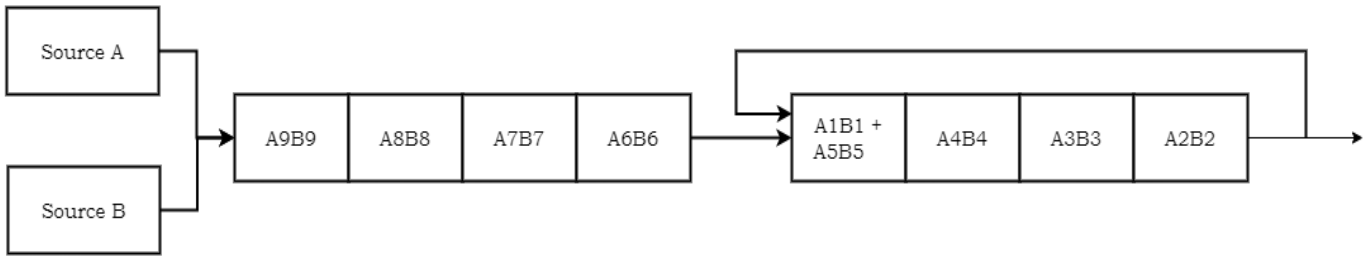


- ✓ The ***floating- point multiplier pipeline and floating-point adder pipeline*** are assumed to have four segments each.
- ✓ All segments registers in the multiplier and adder pipelines are ***initialized to zero.***
- ✓ Therefore, the output for the adder is zero for the first eight cycles until both pipes are full.
- ✓ ***After the first four cycles***, the products begin to be added to the output of the adder. ***At the end of the eighth cycle***, the first four products A_1B_1 through A_4B_4 are in 4- adder segments and next 4 products A_5B_5 through A_8B_8 are in 4- multiplier segments as shown in figure.

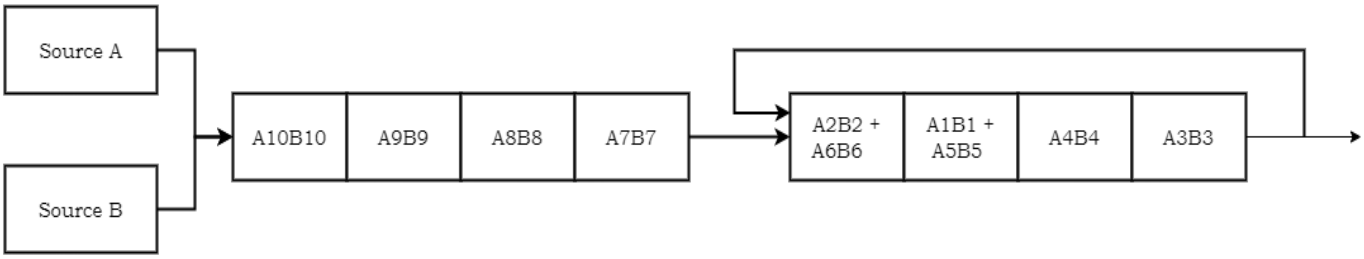


- ✓ At the beginning of the 9^{th} cycle, the output of the adder is A_1B_1 and the output of the multiplier is A_5B_5. Thus the 9^{th} cycle starts the addition $A_1B_1 + A_5B_5$ in the adder pipeline. The 10^{th} cycle starts the addition $A_2B2 + A_6B_6$ in the adder and so on.

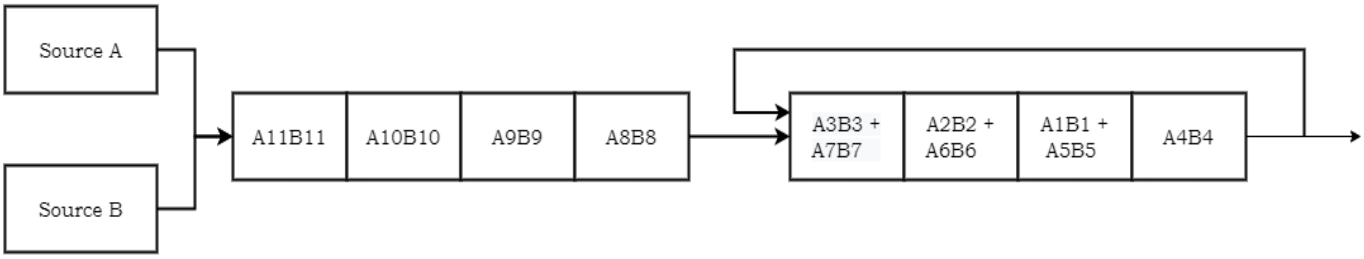
9th Clock Cycle



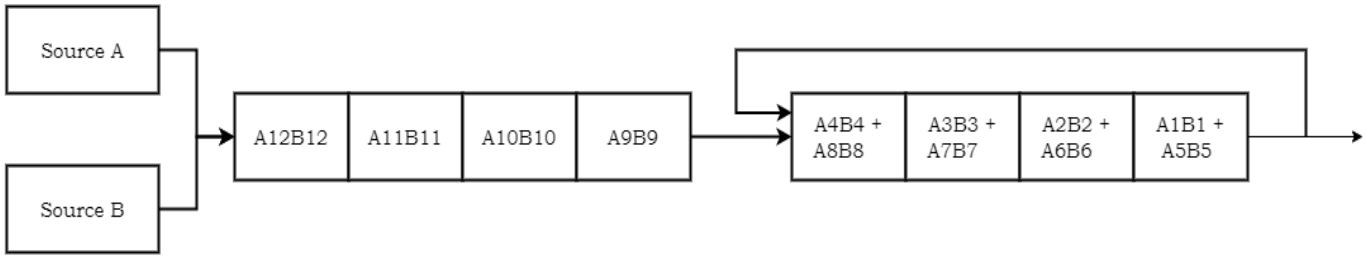
10th Clock Cycle



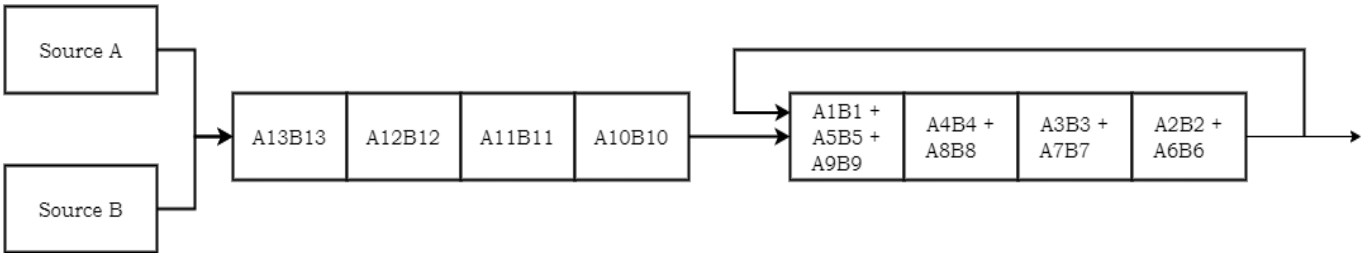
11th Clock Cycle



12th Clock Cycle



13th Clock Cycle



- ✓ This pattern breaks down the sum into four sections as follows.

$$C= A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} +.....+$$

$$A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} +.....+$$

$$A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} +.....+$$

$$A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} +.....$$

- ✓ When there are no more product terms to be added, the system inserts 4 zeros into the multiplier pipeline.
- ✓ The adder pipeline will then have one partial result, in each of its four segments corresponding to the four sums listed in the four rows in the above equation.
- ✓ The four partial results are then added to form the final sum.

.....END.....