Seventh Semester B.E. Degree Examination, Dec.2018/Jan.2019

Advanced Computer Architecture

Time: 3 hrs. Max. Marks: 80

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. List the performance factors and system attributes. Explain how performance factors are influenced by system attributes. **(08 Marks)**

b. Explain the architecture of vector super computer with neat diagram. **(08 Marks)**

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:

$$T = Ic \times CPI \times \tau \qquad (1.1)$$

- The execution of an instruction requires going through a cycle of events involving the instruction fetch, decode, operand(s) 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 cycles 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;

$$T = I_c \times (p + m \times k) \times \tau \qquad (1.2)$$

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,
r 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 1.2.
- The instruction-set architecture affects the program length (Ic) and processor cycle needed (p). The compiler technology affects the values of I_c, p and the memory reference count (m).
- The CPU implementation and control determine the total processor time (*p*. τ) needed.
- Finally, the memory technology and hierarchy design affect the memory access latency (*k*. τ). The above CPU time can be used as a basis in estimating the execution rate of a processor.

Table 1.2 Performance Factors Versus System Attributes

System Attributes	Performance Factors				
	Instr. Count,	Average Cycles per Instruction, CPI			Processor Cycle Time, τ
		Processor Cycles per Instruction, p	Memory References per Instruction, *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

1.3.1 Vector Supercomputers

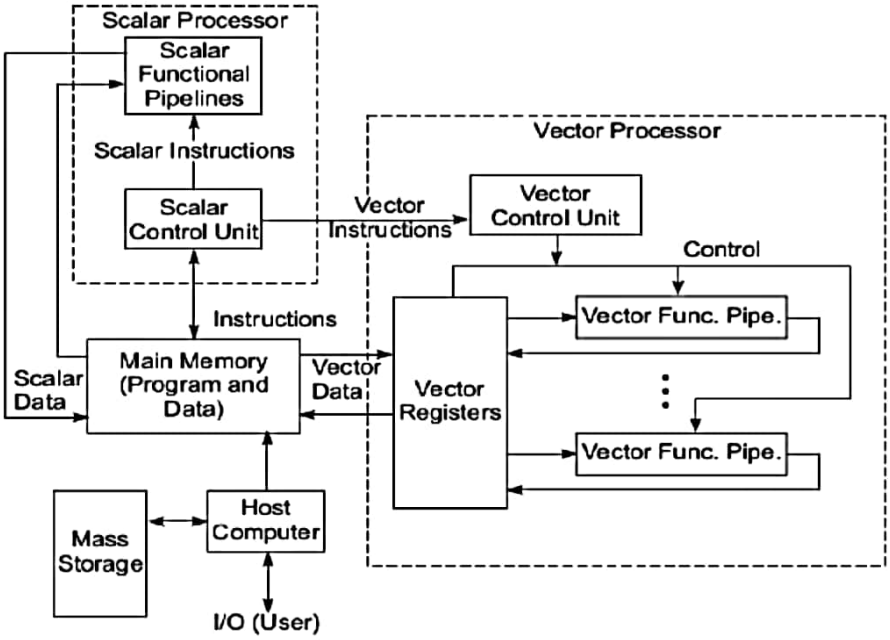


Fig. 1.11 The architecture of a vector supercomputer

- A vector computer is often built on top of a scalar processor.
- As shown in Fig. 1.11, the vector processor is attached to the scalar processor as an optional feature.
- Program and data are first loaded into the main memory through a host computer.
- All instructions are first decoded by the scalar control unit.
- If the decoded instruction is a scalar operation or a program control operation, it will be directly executed by the scalar processor using the scalar functional pipelines.
- If the instruction is decoded as a vector operation, it will be sent to the vector control unit.
- This control unit will supervise the flow of vector data between the main memory and vector functional pipelines.
- The vector data flow is coordinated by the control unit. A number of vector functional pipelines may be built into a vector processor.

Vector Processor Models

- Figure 1.11 shows a **register-to-register** architecture.
- Vector registers are used to hold the vector operands, intermediate and final vector results.
- The vector functional pipelines retrieve operands from and put results into the vector registers.
- All vector registers are programmable in user instructions.
- Each vector register is equipped with a component counter which keeps track of the component registers used in successive pipeline cycles.
- The length of each vector register is usually fixed, say, sixty-four 64-bit component registers in a vector register in a Cray Series supercomputer.
- Other machines, like the Fujitsu VP2000 Series, use reconfigurable vector registers to dynamically

2 a. What are the conditions of parallelism? Explain the types of data dependence. **(06 Marks)**
b. What are the metrics affecting scalability of a computer system? **(06 Marks)**
c. What are the important characteristics of parallel algorithms? **(04 Marks)**

2.1 Condition of parallelism

2.2.1 Data and Resource Dependence

- The ability to execute several program segments in parallel requires each segment to be independent of the other segments. We use a dependence graph to describe the relations.
- The nodes of a dependence graph correspond to the program statement (instructions), and directed edges with different labels are used to represent the ordered relations among the statements.
- The analysis of dependence graphs shows where opportunity exists for parallelization and vectorization.

Control Dependence:

- This refers to the situation where the order of the execution of statements cannot be determined before run time.
- For example all condition statement, where the flow of statement depends on the output.
- Different paths taken after a conditional branch may depend on the data hence we need to eliminate this data dependence among the instructions.
- This dependence also exists between operations performed in successive iterations of looping procedure. Control dependence often prohibits parallelism from being exploited.

Control-independent example:

```
for (i=0; i<n; i++)
{
        a[i] = c[i];
        if (a[i] < 0) a[i] = 1;
}
```

Resource dependence:

- Data and control dependencies are based on the independence of the work to be done.
- Resource independence is concerned with conflicts in using shared resources, such as registers, integer and floating point ALUs, etc. ALU conflicts are called ALU dependence.
- Memory (storage) conflicts are called storage dependence.

Bernstein's Conditions

Bernstein's conditions are a set of conditions which must exist if two processes can execute in parallel.

Notation

- I_i is the set of all input variables for a process P_i. I_i is also called the read set or domain of Pi. Oi is the set of all output variables for a process Pi . Oi is also called write set.
- If P*1* and P*2* can execute in parallel (which is written as P*1* || P*2*), then:

$$I_1 \cap O_2 = \emptyset$$
$$I_2 \cap O_1 = \emptyset$$
$$O_1 \cap O_2 = \emptyset$$

Bernsteins Condition 2 : In terms of data dependencies, Bernstein's conditions imply that two processes can execute in parallel if they are flow-independent, antiindependent, and outputindependent.

The parallelism relation || is commutative (Pi || Pj implies Pj || Pi), but not transitive (Pi || Pj and Pj || Pk does not imply Pi || Pk) . Therefore, || is not an equivalence relation. Intersection of the input sets is allowed.

Data dependence:

The ordering relationship between statements is indicated by the data dependence. Five type of data dependence are defined below:

1. **Flow dependence:** A statement S2 is flow dependent on S1 if an execution path exists from s1 to S2 and if at least one output (variables assigned) of S1feeds in as input (operands to be used) to S2 also called RAW hazard and denoted as $S_1 \rightarrow S_2$

2. **Antidependence:** Statement S2 is antidependent on the statement S1 if S2 follows S1 in the program order and if the output of S2 overlaps the input to S1 also called RAW hazard and denoted as $S_1 \mapsto S_2$

3. **Output dependence:** Two statements are output dependent if they produce (write) the same output variable. Also called WAW hazard and denoted as $S_1 \leftrightarrow S_2$

4. **I/O dependence:** Read and write are I/O statements. I/O dependence occurs not because the same variable is involved but because the same file referenced by both I/O statement.

5. **Unknown dependence:** The dependence relation between two statements cannot be determined in the following situations:
   - The subscript of a variable is itself subscribed (indirect addressing)
   - The subscript does not contain the loop index variable.
   - A variable appears more than once with subscripts having different coefficients of the loop variable.
   - The subscript is non linear in the loop index variable.

2b)

Scalability - The attributes of a computer system which allow it to be gracefully and linearly scaled up or down in size, to handle smaller or larger workloads, or to obtain proportional decreases or increase in speed on a given application. The applications run on a scalable machine may not scale well. Good scalability requires the algorithm *and* the machine to have the right properties

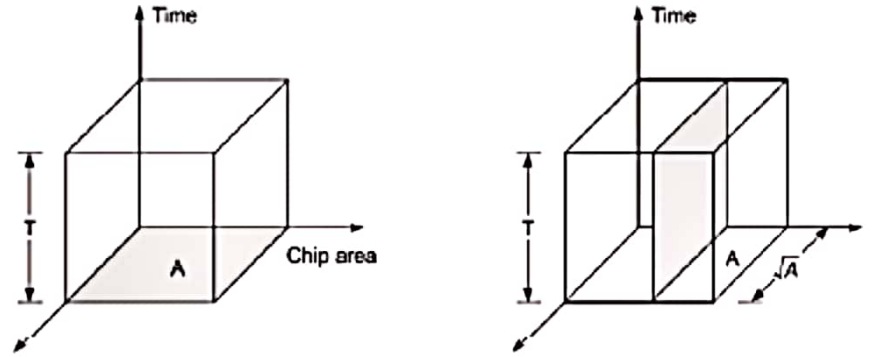
Thus in general there are five performance factors (*Ic*, p, m, k, t) which are influenced by four system attributes:

- instruction-set architecture (affects *Ic* and *p*)
- compiler technology (affects *Ic* and *p* and *m*)
- CPU implementation and control (affects *p* *t*) cache and memory hierarchy (affects memory access latency, *k* *t*)
- Total CPU time can be used as a basis in estimating the execution rate of a processor.

2c)

1.4.2 VLSI Model

Parallel computers rely on the use of VLSI chips to fabricate the major components such as processor arrays memory arrays and large scale switching networks. The rapid advent of very large scale intergrated (VSLI) technology now computer architects are trying to implement parallel algorithms directly in hardware. An AT2 model is an example for two dimension VLSI chips



(a) Memory-limited bound on chip area A and I/O-limited bound on chip history represented by the volume AT

(b) Communication-limited bound on the bisection $\sqrt{A}T$

Fig. 1.15 The AT^2 complexity model of two-dimensional VLSI chips

Seventh Semester B.E. Degree Examination, Dec.2019/Jan.2020
Advanced Computer Architecture

Time: 3 hrs. Max. Marks: 80

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. With a neat diagram explain the elements of modern computer system. **(08 Marks)**

b. Explain Flynn's classification of computer architecture. **(08 Marks)**

OR

2 a. Define data dependency. Explain different functions of data dependency with the help of dependency graph. **(08 Marks)**

b. A 4 MHz processor was used to execute a benchmark program with the following instruction mix and clock cycle counts.

Instruction type	Instruction count	Cycles/instruction
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. **(08 Marks)**

1a

1.1.2 Elements of Modern Computers

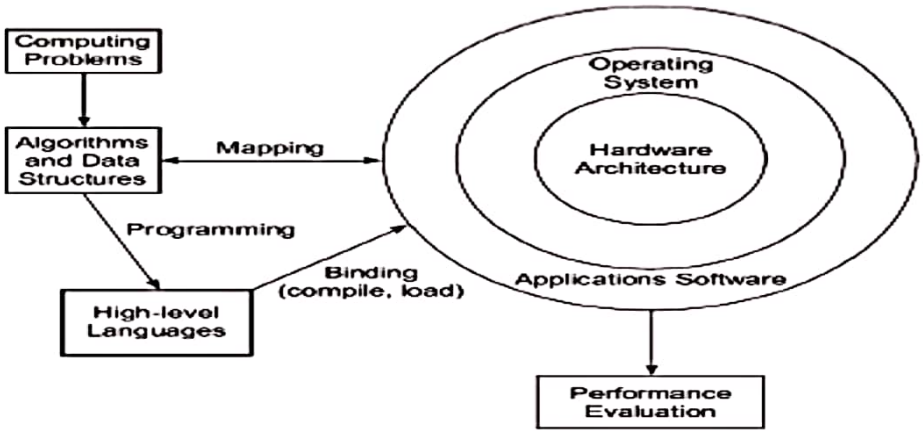


Fig. 1.1 Elements of a modern computer system

- **Computing Problems**

- The use of a computer is driven by real-life problems demanding fast and accurate solutions. Depending on the nature of the problems, the solutions may require different computing resources.
- For numerical problems in science and technology, the solutions demand complex mathematical formulations and tedious integer or floating-point computations.
- For alpha numerical problems in business and government, the solutions demand accurate transactions, large database management, and information retrieval operations.
- For artificial intelligence (AI) problems, the solutions demand logic inferences and symbolic manipulations.
- These computing problems have been labeled *numerical computing, transaction processing,* and *logical reasoning*.
- Some complex problems may demand a combination of these processing modes.

- **Hardware Resources**

- A modern computer system demonstrates its power through coordinated efforts by hardware resources, an operating system, and application software.
- Processors, memory, and peripheral devices form the hardware core of a computer system.
- Special hardware interfaces are often built into I/O devices, such as terminals, workstations, optical page scanners, magnetic ink character recognizers, modems, file servers, voice data entry, printers, and plotters.
- These peripherals are connected to mainframe computers directly or through local or wide-area networks.

- **System Software Support**

- Software support is needed for the development of efficient programs in high-level languages. The source code written in a HLL must be first translated into object code by an optimizing compiler.
- The *compiler* assigns variables to registers or to memory words and reserves functional units for operators.

- **Compiler Support**

There are three compiler upgrade approaches:

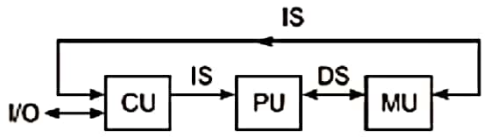
- **Preprocessor:** A preprocessor uses a sequential compiler and a low-level library of the target computer to implement high-level parallel constructs.
- **Precompiler:** The precompiler approach requires some program flow analysis, dependence checking, and limited optimizations toward parallelism detection.
- **Parallelizing Compiler:** This approach demands a fully developed parallelizing or vectorizing compiler which can automatically detect parallelism in source code and transform sequential codes into parallel constructs.

Flynn's Classification

Michael Flynn (1972) introduced a classification of various computer architectures based on notions of instruction and data streams.

1. **SISD (Single Instruction stream over a Single Data stream) computers**
2. **SIMD (Single Instruction stream over Multiple Data streams) machines**
3. **MIMD (Multiple Instruction streams over Multiple Data streams) machines.**
4. **MISD (Multiple Instruction streams and a Single Data stream) machines**

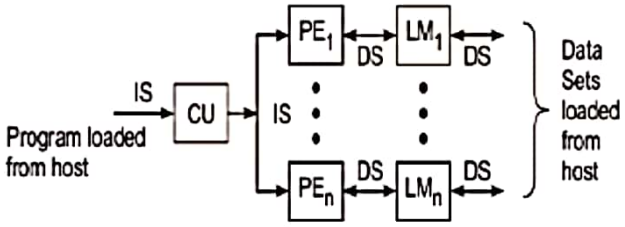
1. SISD (Single Instruction stream over a Single Data stream) computers



(a) SISD uniprocessor architecture

- **Conventional sequential machiunes are called SISD computers.**
- **They are also called scalar processor i.e., one instruction at a time and each instruction have only one set of operands.**
- **Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle**
- **Single data: only one data stream is being used as input during any one clock cycle**
- **Deterministic execution**

2. SIMD (Single Instruction stream over Multiple Data streams) machines

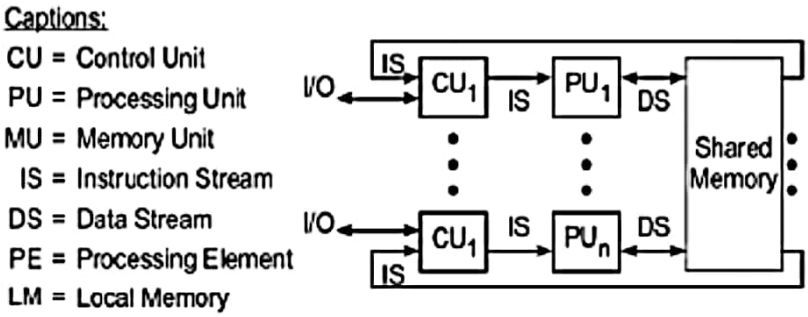


(b) SIMD architecture (with distributed memory)

A type of parallel computer

- **Single instruction: All processing units execute the same instruction issued by the control unit at any given clock cycle.**
- **Multiple data: Each processing unit can operate on a different data element. The processors are connected to shared memory or interconnection network providing multiple data to processing unit.**
- **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.**
- **Thus single instruction is executed by different processing unit on different set of data.**
- **Best suited for specialized problems characterized by a high degree of regularity, such as image**

3. MIMD (Multiple Instruction streams over Multiple Data streams) machines.



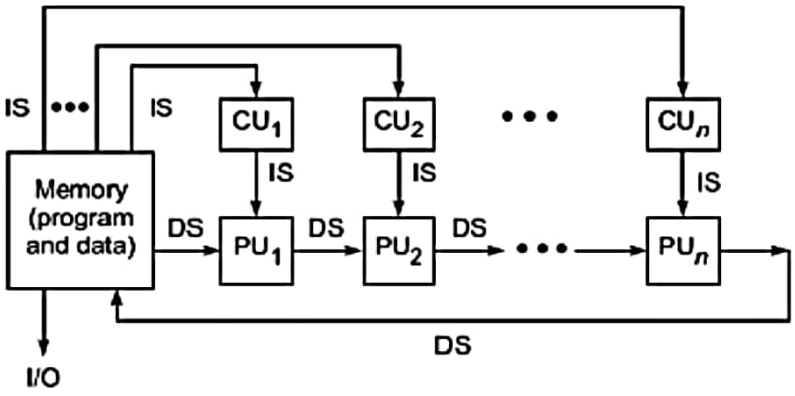
(c) MIMD architecture (with shared memory)

- A single data stream is fed into multiple processing units.
- Each processing unit operates on the data independently via independent instruction streams.
- A single data stream is forwarded to different processing unit which are connected to different control unit and execute instruction given to it by control unit to which it is attached.
- Thus in these computers same data flow through a linear array of processors executing different instruction streams.
- This architecture is also known as Systolic Arrays for pipelined execution of specific instructions.

  Some conceivable uses might be:

  1. multiple frequency filters operating on a single signal stream
  2. multiple cryptography algorithms attempting to crack a single coded message.

4. MISD (Multiple Instruction streams and a Single Data stream) machines



(d) MISD architecture (the systolic array)

Fig. 1.3 Flynn's classification of computer architectures (Derived from Michael Flynn,

- **Multiple Instructions:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream, multiple data stream is provided by shared memory.
- Can be categorized as loosely coupled or tightly coupled depending on sharing of data and control.
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- There are multiple processors each processing different tasks.
- Examples: most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.

Seventh Semester B.E. Degree Examination, June/July 2019

Advanced Computer Architectures

Time: 3 hrs. Max. Marks: 80

Note: ***Answer any FIVE full questions, choosing ONE full question from each module.***

Module-1

1 a. Explain the evolution of computer architecture. **(08 Marks)**

b. Explain with diagram the operational model of SIMD super computer. **(08 Marks)**

OR

2 a. Explain the Bernstein's conditions for parallelism. Detect the parallelism in the following code using Bernstein's conditions. (Assume no pipeline).

$P_1 : C = D \times E$; $P_2 : M = G + C$; $P_3 : A = B + C$; $P_4 : C = L + M$; $P_5 : G \div E$. **(08 Marks)**

b. With a diagram, explain the operation of tagged token data flow computer. **(08 Marks)**

1.1.3 Evolution of Computer Architecture

- The study of computer architecture involves both hardware organization and programming/software requirements.
- As seen by an assembly language programmer, computer architecture is abstracted by its instruction set, which includes opcode (operation codes), addressing modes, registers, virtual memory, etc.
- From the hardware implementation point of view, the abstract machine is organized with CPUs, caches, buses, microcode, pipelines, physical memory, etc.

- Therefore, the study of architecture covers both instruction-set architectures and machine implementation organizations.

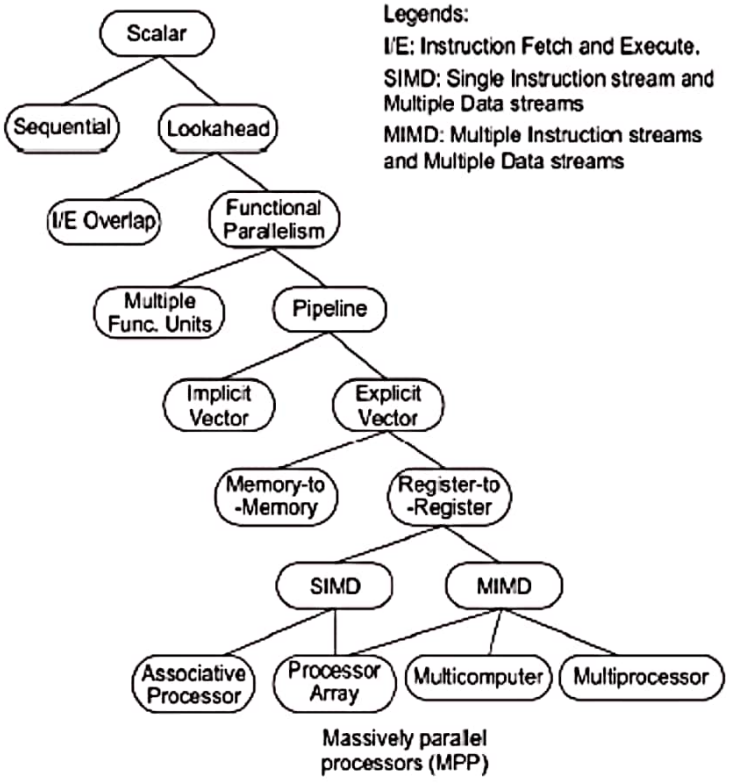


Fig. 1.2 Tree showing architectural evolution from sequential scalar computers to vector processors and parallel computers

1b

1.3.2 SIMD Supercomputers

SIMD computers have a single instruction stream over multiple data streams.

An operational model of an SIMD computer is specified by a 5-tuple:

$$M = (N, C, I, M, R)$$

where

1. N is the number of *processing elements* (PEs) in the machine. For example, the Illiac IV had 64 PEs and the Connection Machine CM-2 had 65,536 PEs.
2. C is the set of instructions directly executed by the *control unit* (CU), including scalar and program flow control instructions.
3. I s the set of instructions broadcast by the CU to all PEs for parallel execution. These include arithmetic, logic, data routing, masking, and other local operations executed by each active PE over data within that PE.
4. M is the set of masking schemes, where each mask partitions the set of PEs into enabled and disabled subsets.

R is the set of data-routing functions, specifying various patterns to be set up in the interconnection network for inter-PE communications.

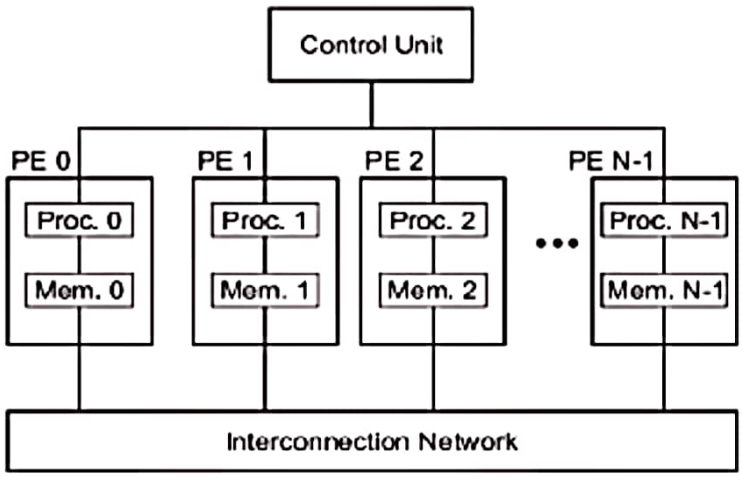


Fig. 1.12 Operational model of SIMD computers

2b)tagged token data flow computer

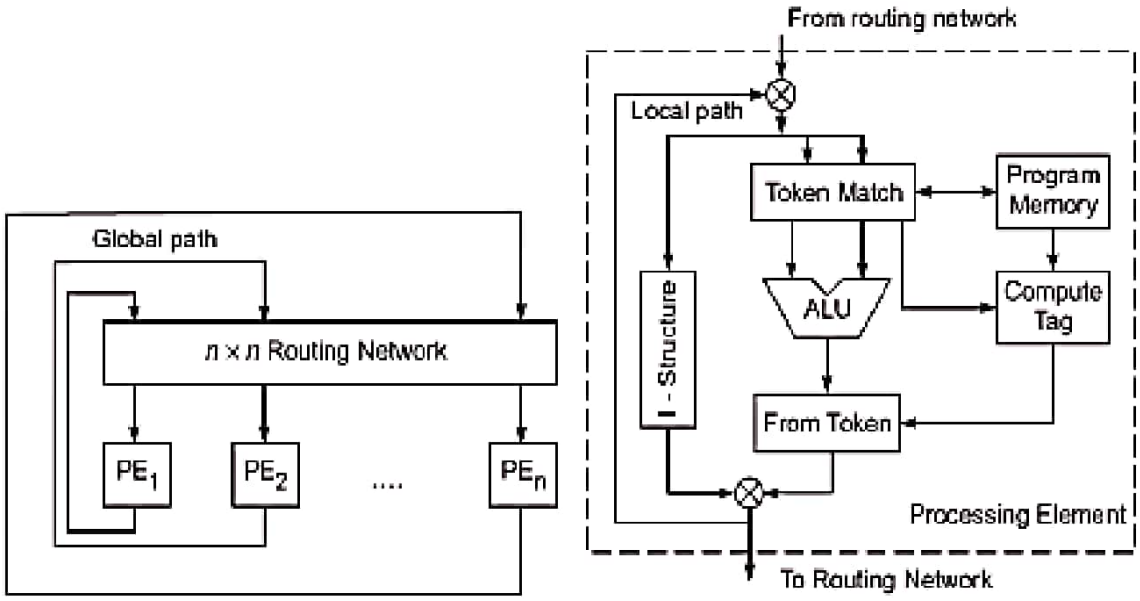


Fig. 2.12 The MIT tagged-token dataflow computer (adapted from Arvind and Iannucci, 1986 with permission)

A Dataflow Architecture

- The Arvind machine (MIT) has N PEs and an N-by-N interconnection network.
- Each PE has a token-matching mechanism that dispatches only instructions with data tokens available.
- Each datum is tagged with
  - address of instruction to which it belongs
  - context in which the instruction is being executed
- Tagged tokens enter PE through local path (pipelined), and can also be communicated to other PEs through the routing network.
- Instruction address(es) effectively replace the program counter in a control flow machine.
- Context identifier effectively replaces the frame base register in a control flow machine.
- Since the dataflow machine matches the data tags from one instruction with successors, synchronized instruction execution is implicit.
- An *I-structure* in each PE is provided to eliminate excessive copying of data structures.
- Each word of the I-structure has a **two-bit tag indicating whether the value is empty, full or has pending read requests.**
- This is a retreat from the pure dataflow approach.
- Special compiler technology needed for dataflow machines.