Computer Architecture

V semester

Chapter 9: Introduction to Parallel Processing

6 hours

Compiled by :

Ankit Bhattarai, Assistant Professor
Email: ankitbhattarai@cosmoscollege.edu.np

Topics to be discussed

- Multiprocessor
- Organization of Multiprocessor
- Cache Coherence
- Vector Processor
- Array Processor
- Multithreading

Introduction

Parallelism in Uniprocessor system

- A computer system achieves parallelism when it performs two or more tasks simultaneously. A system that processes two different instructions simultaneously could be considered to perform parallel processing. If a system that performs different operation on same instruction is not considered as parallel processing.
- Parallelism in uniprocessor is achieved by instruction pipelining**. By overlapping the fetching, decoding, executions of instructions, a RISC or CISC processors with an instruction pipeline exhibits parallelism.**
- Arithmetic pipelining is another example of parallelism in uniprocessor. IOP is another example of parallelism in uniprocessor where IOP performs data transfer whereas CPU executes instruction. DMA when operating in transparent mode supports parallelism.

Multiprocessor

- A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment. Multiprocessor can mean either central processing unit (CPU) or an input-output processor (IOP).
- For multiprocessor system IOP must have computational facility like CPU in a single CPU system. Multiprocessor are classified as (MIMD) multiple instruction stream, multiple data stream.
- A multiprocessor system is controlled by one operating system that provides interaction between processor and all the components of the system co-operate in the solution of a problem.

Multiprocessor

- Multiprocessing improves the reliability of the system so that a failure error in one part has a limited effect on the rest of the system. If a fault occurs in one process the other can be assigned to perform that task.

- A multiprocessor system supports parallel and distributed computing techniques.

- Multiprocessor system supports pipelining where specific task is divided into several subtasks that must be performed in a sequence.

Multiprocessor

- Multiprocessor system supports Vector Computing that involves the usage of vector processors, wherein operations such as ‘multiplication’ are divided into many steps and are then applied to a stream of operands (“vectors”).

- Multiprocessor system are classified on the way their memory is organized. Shared memory: tightly coupled system; local memory: loosely couple system.

- Multiprocessor can improve performance by decomposing a program into parallel executable tasks. It can be achieved in two ways: → User can explicitly define that certain task of program be executed in parallel, →Provide a compiler with multiprocessor software that can detect parallelism in user’s program.

Organization of multiprocessor system

Flynn's Taxonomy: There are many ways to organize the processors and memory within the microprocessor system, and different ways to classify these systems, including Flynn's classification, a common accepted taxonomy of computer organization based on the flow of instructions and data within the computer.

Flynn's classification is based on instruction and data processing. A computer is classified by whether it processes a single instruction at a time or multiple instructions simultaneously, whether it operates on one or multiple data sets. It's four categories are as follows:

1. SISD: Single instruction single data
2. SIMD: Single instruction multiple data
3. MISD: Multiple instruction single data
4. MIMD: Multiple instruction multiple data

Organization of multiprocessor system

A taxonomy first introduced by Flynn [FLYN72] is still the most common way of categorizing systems with parallel processing capability. Flynn proposed the following categories of computer systems:

i. **Single instruction, single data (SISD) stream:** A single processor executes a single instruction stream to operate on data stored in a single memory. *Uniprocessors* fall into this category.

ii. **Single instruction, multiple data (SIMD) stream:** A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis. Each processing element has an associated data memory, so that instructions are executed on different sets of data by different processors. *Vector and array processors* fall into this category.

Organization of multiprocessor system

iii. Multiple instruction, single data (MISD) stream: A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence.

This structure is not commercially implemented.

iv. Multiple instruction, multiple data (MIMD) stream: A set of processors simultaneously execute different instruction sequences on different data sets.

SMPs (Symmetric microprocessor), clusters, and NUMA (Non-uniform memory access) systems fit into this category.

System Topologies/ Interconnection structures

- The topology of a multiprocessor system refers to the pattern of connections between its processors.
- Various factors, typically involving a cost performance trade-off, determine which topology a computer designer will select for a multiprocessor system.

Some topologies metrics are:

1. **Diameter**: maximum distance between two processors in the computer system.
2. **Bandwidth**: It is the capacity of the communications link multiplied by the number of such links in the system. In best case scenario all links are active at once.

System Topologies

a. **Shared Bus**: processor communicate with each other exclusively via this bus; it can handle only one data transmission at a time.

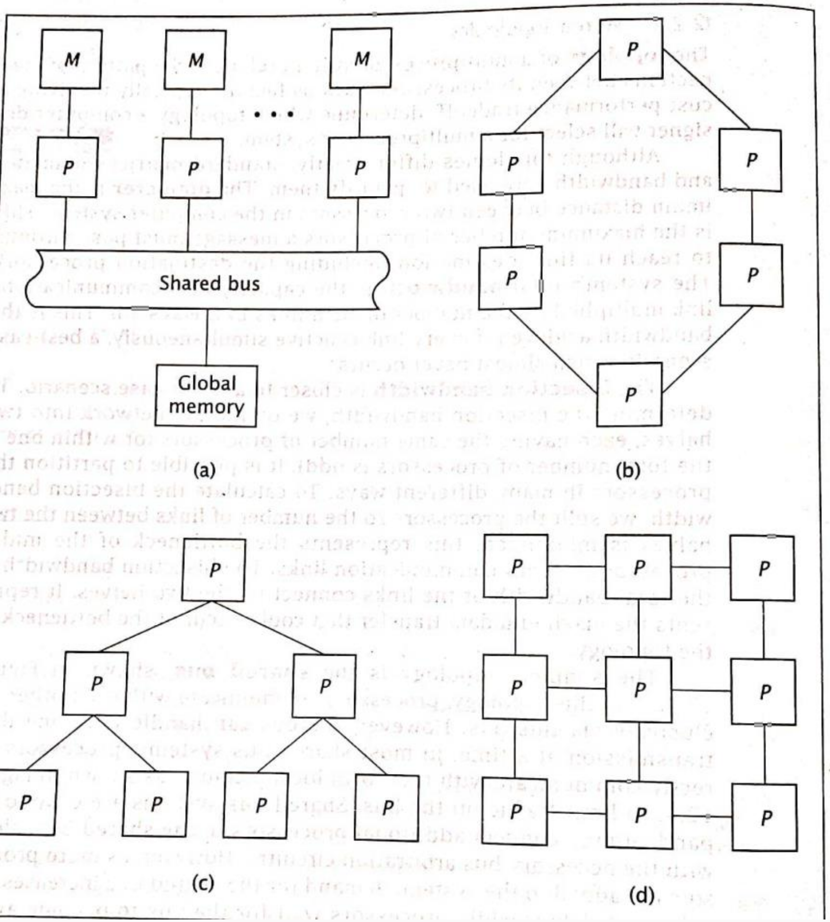
b. **Ring topology**: uses direct dedicated connections between processors instead of a shared bus; a piece of data may have to travel through several processors to reach its final destination; all processors must have two communication links as opposed to the one used in a shared bus system.

c. **Tree topology**: Like the ring, it uses direct connections between processors; each processor has three connections; tracing through this topology we can see that there is only one unique path between any pair of processors.

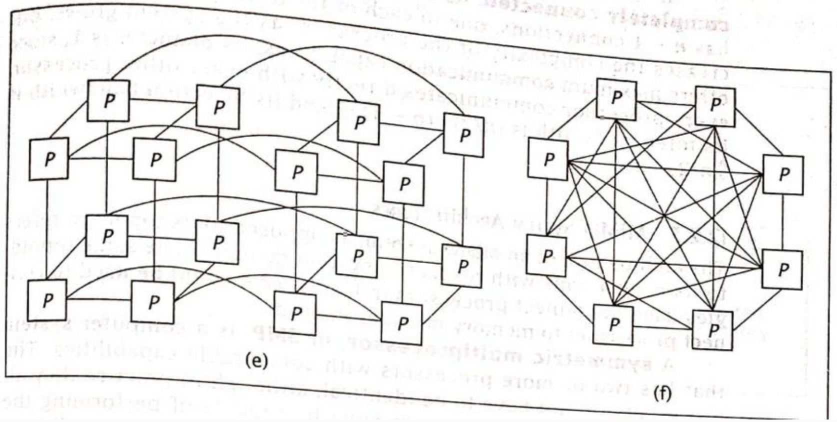
d. **Mesh topology**: Every processor connects to the processors above and below it, and its right and left.

Figure 12.4

MIMD system topologies: (a) shared bus, (b) ring, (c) tree, (d) mesh, (e) hypercube, and (f) completely connected



(continued)



System Topologies

e**. Hypercube**: It is a multidimensional mesh; it has n processors each with lg n connections; each processor connects to all other processors whose binary values differ by only one bit.

f. **Completely connected**: It is the most extreme connections; every processor has n-1 connections, one to each of the other processors. This increases the complexity of the processors as the system grows, but offers maximum communication capabilities. Its diameter is 1, since every processor communicates directly with every other processor.

MIMD System Architectures

It refers to its connections with respect to the system memory.

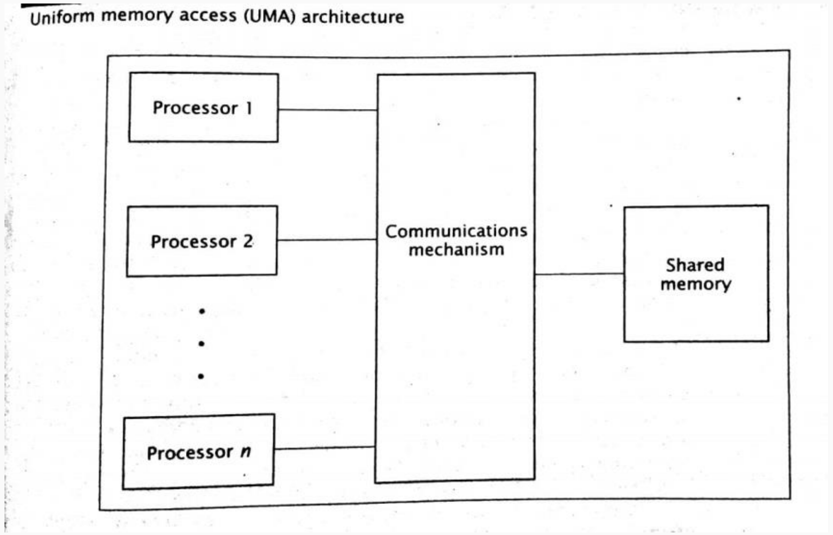
Symmetric multiprocessor (SMP):

- It is a computer system that has two or more processors with comparable capabilities.
- All processors must be capable of performing the same functions; this is the symmetry of SMPs.
- The processors all have access to the same I/O devices and memory modules. An integrated operating system controls the entire computer system.

MIMD System Architectures

1. **Uniform memory access (UMA) architecture**

- UMA gives all CPUs equal access to all locations in shared memory.
- They interact with shared memory via communication mechanism, which may be as simple as a shared bus or as complex as a multistage interconnection network.
- Each processor may have its own cache memory and not directly accessible by the other processors.

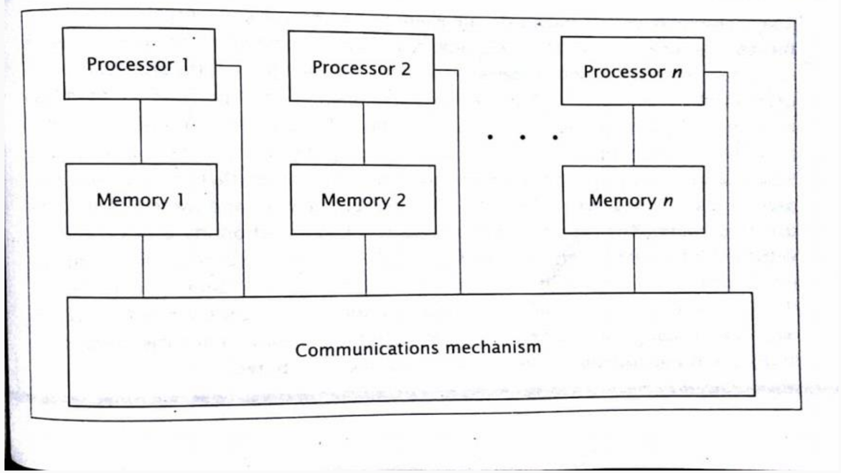


MIMD System Architectures

2. **Non-Uniform memory access (NUMA) architecture**

- It do not allow uniform access to all shared memory locations.
- Each processor can access the memory module closest to it, its local shared memory, more quickly than the other modules; hence, the memory access times are non uniform.
- NUMA machines are more scalable.

Nonuniform memory access (NUMA) architecture

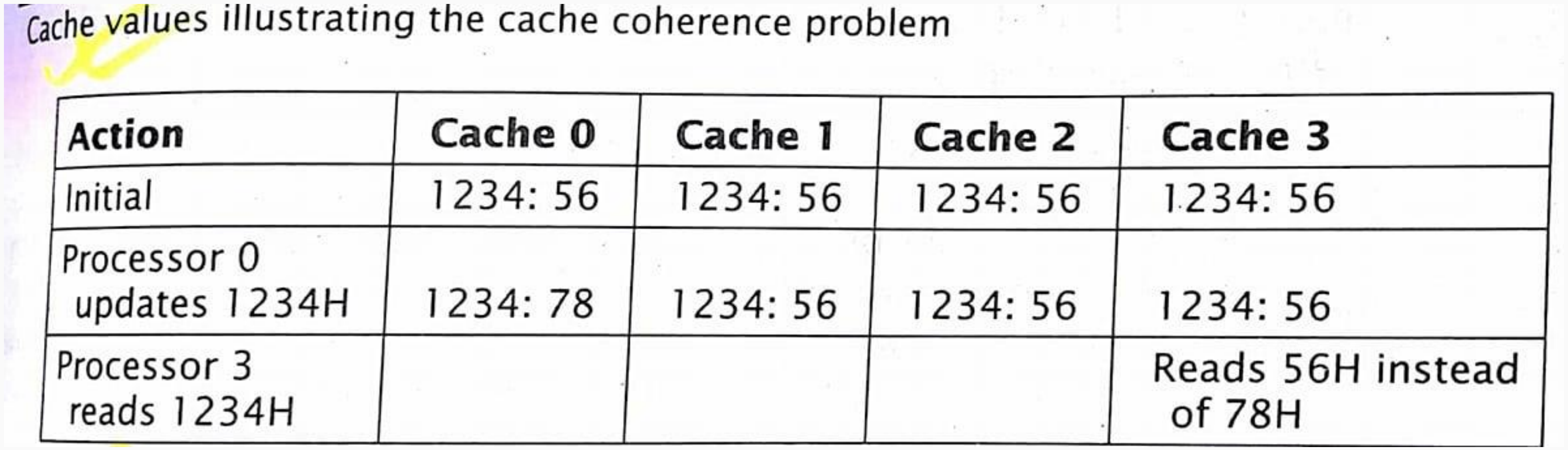


Cache Coherence

- Unlike uniprocessor systems, however, multiprocessors have individual caches for each processor.
- This can lead to problems when two or more caches hold the value of the same memory location simultaneously.
- As one processor stores a value to that location in its cache, the other cache will have an invalid value in its location.
- Using a write-through cache will not resolve this problem, since it would update main memory but not the other caches.
- In addition, the extra writes to main memory would decrease system performance.
- This is the **cache coherence** problem.

Contd…

- To illustrate this cache coherence problem, consider a multiprocessor system with four processors, each of which has a write-back cache.



Cache values illustrating the cache coherence problem

Action	Cache 0	Cache 1	Cache 2	Cache 3
Initial	1234: 56	1234: 56	1234: 56	1234: 56
Processor 0 updates 1234H	1234: 78	1234: 56	1234: 56	1234: 56
Processor 3 reads 1234H				Reads 56H instead of 78H

Contd…

- It is possible to resolve the cache coherence problem during program compilation.
- The compiler can mark all shared data as **non-cacheable**, thus forcing all accesses to this data to be from shared memory.
- Although, this resolves the problem, it lowers the cache hit ratio and reduces overall system performance.
- To reduce these effects and improve the hit ratio, a compiler may mark data as non-cacheable only at specified, critical parts of code.

Contd…

- Hardware solution scheme is the **cache directory.**
- A directory controller is integrated with the main memory; it maintains a cache directory in main memory, which contains information on the contents of local caches.
- All cache writes are also sent to the directory controller so that it can update the cache directory.
- When the processor writes data to its cache, the directory controller checks to see which other caches also have that data.
- It invalidates that data in those caches by marking its locations as empty.

Contd…

- One popular solution to the cache coherence problem is called **snooping**.
- In snooping, each cache(sometimes called snoopy cache) monitors memory activity on the system bus.
- Whenever it encounters a memory access(by another processor) to a location that it currently holds, it takes appropriate action.

  => If the request is memory read, and the contents in its cache are the same as those in main memory, it simply notes that another cache also contains this data.

  => If the request is memory read, and the contents in its cache are different than those in main memory, the processor must have written data to the cache that has not yet been written back to main memory.
  The cache intercepts the memory read request, sending its data to both main memory and the cache which requested the data.

  => If the request is the memory write, it simply marks its own data as invalid, essentially removing it from the cache.

Introduction to Vector Processing

- A vector processor is an ensemble of hardware resources, including vector registers, functional pipelines, processing elements and register counters for performing register operations.
- Vector processing occurs when ***arithmetic or logical operations are applied to vectors***. It is distinguished from scalar processing which operates on one or one pair of data. The conversion from scalar code to vector code is called vectorization.
- Both pipelined processors and SIMD computers can perform vector operations.
- Vector processing reduces software overhead incurred in the maintenance of looping control, reduces memory access conflicts and above all matches nicely with pipelining and segmentation concept to generate one result per each clock cycle continuously.

Introduction to Vector Processing

- Computers with vector processing capabilities are in demand in specialized applications.

e.g.

✓ Long-range weather forecasting
✓ Petroleum explorations
✓ Seismic data analysis
✓ Medical diagnosis
✓ Artificial intelligence and expert systems
✓ Image processing
✓ Mapping the human genome

- **Vector Processor (computer)**

➢ Ability to process vectors, and related data structures such as matrices and multi-dimensional arrays, much faster than conventional computers

➢ Vector Processors may also be pipelined

Introduction to Vector Processing

- To examine the difference between a conventional scalar processor and a vector processor, consider the following Fortran DO loop:

```
      DO  20  I = 1, 100
20    C(I) = B(I) + A(I)
```

Conventional computer

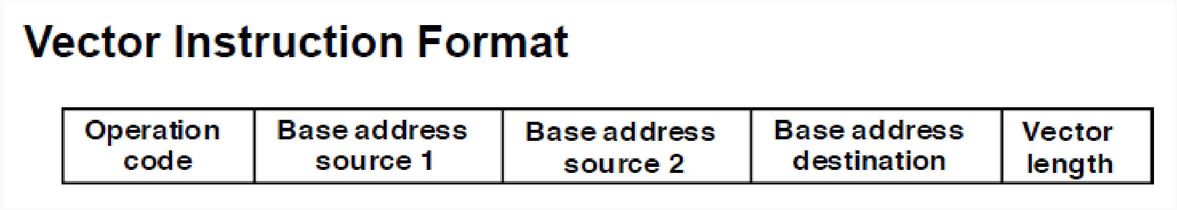
```
      Initialize I = 0
20    Read A(I)
      Read B(I)
      Store C(I) = A(I) + B(I)
      Increment I = i + 1
      If I ≤ 100 goto 20
```

- A computer capable of vector processing eliminates the overhead associated with the time it takes to fetch and execute the instructions in the program loop.

Vector computer

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

Introduction to Vector Processing



Vector Instruction Format

Operation code	Base address source 1	Base address source 2	Base address destination	Vector length

- A possible instruction format for a vector instruction is shown in above figure. This assumes that the vector operands reside in *memory*.

Memory Interleaving

- *Pipeline* and *vector processors* often require simultaneous access to memory from two or more sources.
- An instruction pipeline may require the fetching of an instruction and an operand at the same time from two different segments.
- An arithmetic pipeline usually requires two or more operands to enter the pipeline at the same time.

Introduction to Vector Processing

Memory Interleaving

- Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a common memory address and data buses.
- A memory module is a memory array together with its own address and data registers.

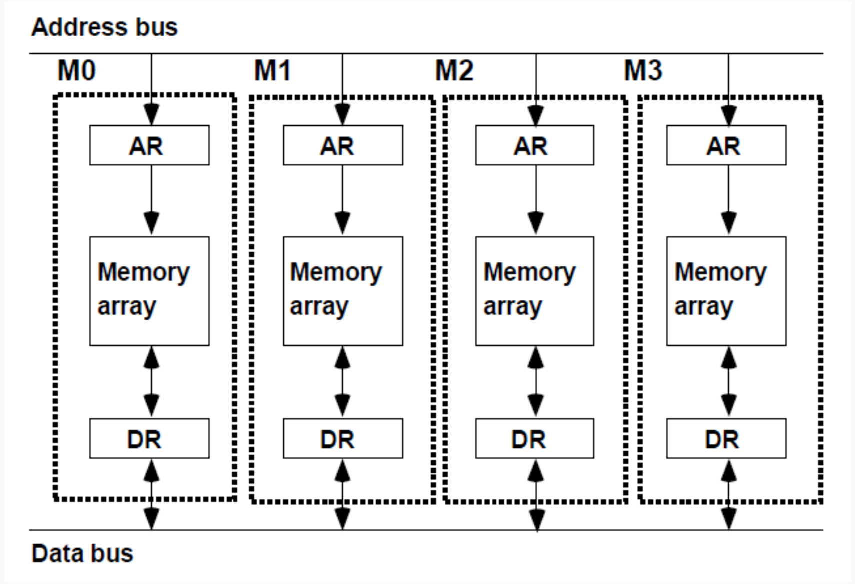


Figure 1: Multiple module memory organization

Introduction to Vector Processing

- The advantage of a modular memory is that it allows the use of a technique called *interleaving*.
- In an interleaved memory, different sets of addresses are assigned to different memory modules.
- By staggering the memory access, the effective memory cycle time can be *reduced by a factor close to the number of modules*

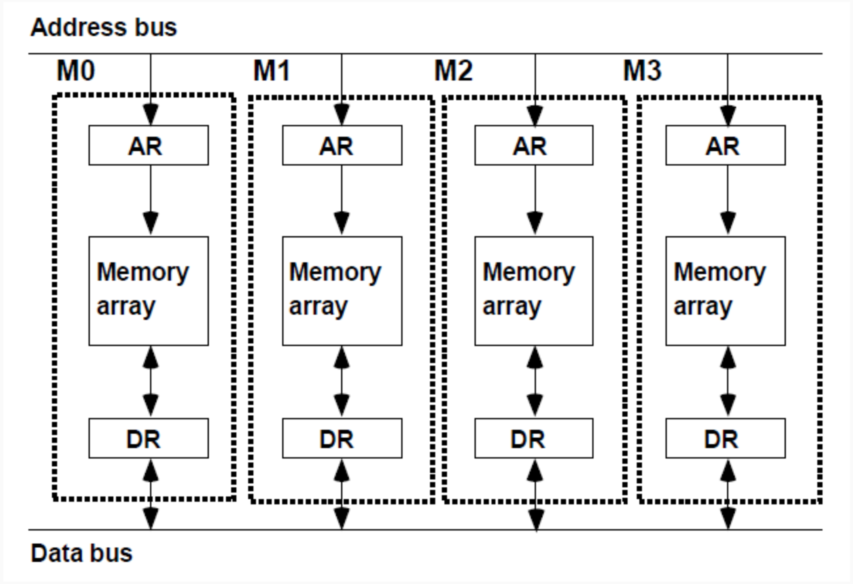


Figure 1: Multiple module memory organization

Introduction to Array Processing

- An array processor is a processor that performs computations on large arrays of data. The term is used to refer to two different types of processors.

1. **Attached array processor:**

✓ Is an auxiliary processor.

✓ It is intended to improve the performance of the host computer in specific numerical computation tasks.

2. **SIMD array processor:**

✓ Has a single-instruction multiple-data organization.

✓ It manipulates vector instructions by means of multiple functional units responding to a common instruction

Introduction to Array Processing

1. **Attached array processor:**

- ✓ An attached array processor is a processor which is attached to a general purpose computer and its purpose is to enhance and improve the performance of that computer in numerical computational tasks.
- ✓ It achieves high performance by means of parallel processing with multiple functional units.

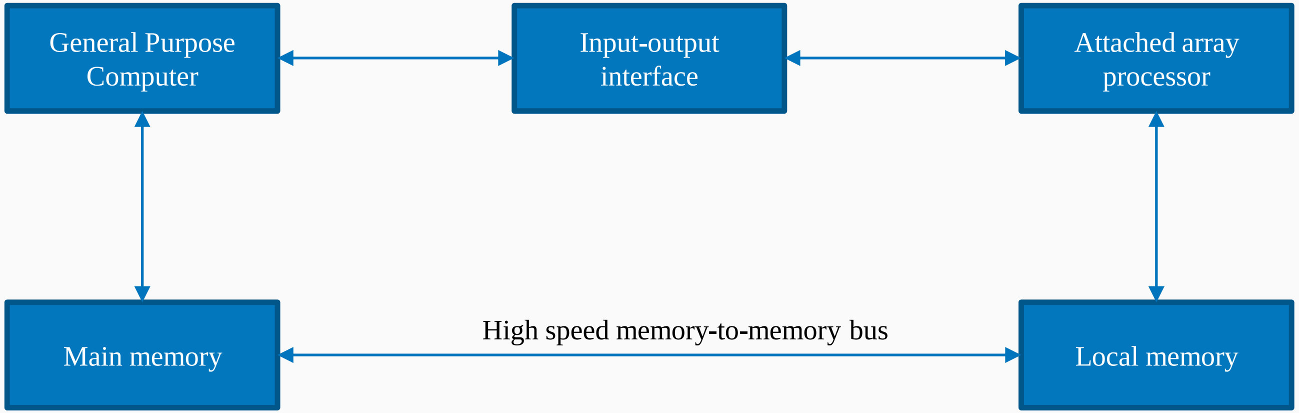


Figure 2. Attached array processor with host computer

Introduction to Array Processing

2. **SIMD array processor:**

✓ An SIMD array processor is a computer with multiple processing units operating in parallel.

✓ A general block diagram of an array processor is shown in Figure 3.

- It contains a set of identical processing elements (PEs), each having a local memory *M*.
- Each PE includes an ALU, a floating-point arithmetic unit, and working registers.
- Vector instructions are broadcast to all PEs simultaneously.

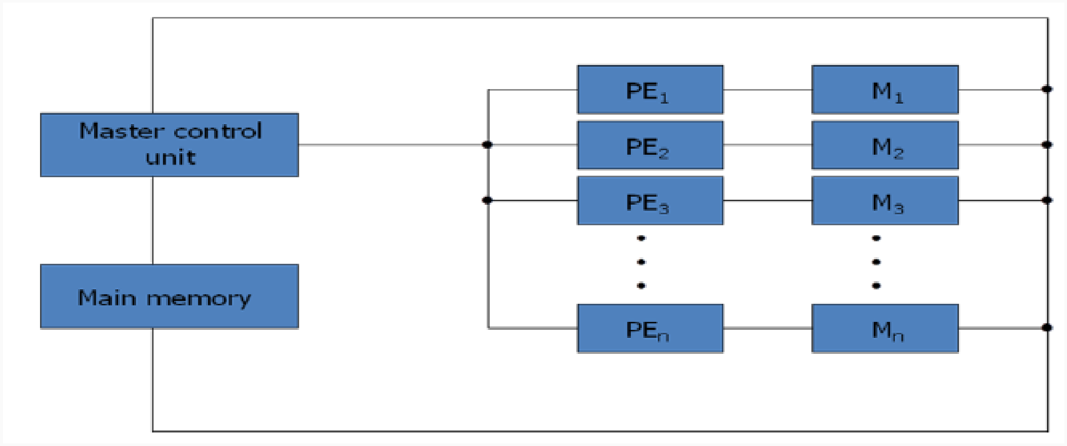


Figure 3: SIMD array processor organization

Why use Array Processor

- Array processors increases the overall instruction processing speed.
- As most of the Array processors operates asynchronously from the host CPU, hence it improves the overall capacity of the system.
- Array Processors has its own local memory, hence providing extra memory for systems with low memory.

Multithreading: Basics

- **Thread**

✓ Instruction stream with state (registers and memory)

✓ Register state is also called “thread context”.

- Threads could be part of the same process (program) or from different programs
- Threads in the same program share the same address space (shared memory model)

- Traditionally, the processor keeps track of the context of a single thread

Introduction to Multithreaded Architecture

- Multithreading is the ability of a central processing unit (CPU) (or a single core in a multi-core processor) to provide multiple threads of execution concurrently, supported by the operating system.
- This approach differs from multiprocessing. In a multithreaded application, the threads share the resources of a single or multiple cores, which include the computing units, the CPU caches, and the translation lookaside buffer (TLB).
- Multithreading aims to increase utilization of a single core by using thread-level parallelism, as well as instruction-level parallelism

Introduction to Multithreaded Architecture

- Multithreading allows a high degree of instruction-parallelism without increasing circuit complexity or power consumption.
- Implicit multithreading refers to the concurrent execution of multiple threads extracted form a single sequential program. These implicit threads can be defined either statically by the compiler or dynamically by hardware.
- For explicit multithreading processor must provide a separate program counter for each thread of execution to be executed concurrently.

Introduction to Multithreaded Architecture

- The designs differ in the amount and type of additional hardware used to support concurrent thread execution. There are four principal approaches to multithreading: -

✓ **Interleaved multithreading:**

- This is also known as fine-grained multithreading.
- The processor deals with two or more thread contexts at a time, switching from one thread to another at each clock cycle.
- If a thread is blocked due to data dependencies or memory latencies, that thread is skipped and a ready thread is executed.

Introduction to Multithreaded Architecture

- ✓ **Blocked multithreading:**
- ▪ This is also known as coarse-grained multithreading. The instructions of a thread are executed successively until an event occurs that may cause delay, such as cache miss. This event switches thread to another.

- ✓ **Simultaneous multithreading:**
- ▪ Instructions are simultaneously issued from multiple threads to the execution units of a superscalar processor. This combines the wide superscalar instruction issue capability with the use of multiple threads contexts

Introduction to Multithreaded Architecture

- ✓ **Chip multiprocessing:**
- ▪ Here the entire processor is replicated on a single chip and each processor handles separate threads. This is referred to as multicore.

Advantages

- takes advantage of the unused computing resources, which may lead to faster overall execution, as these resources would have been idle if only a single thread were executed.

Disadvantages

- Multiple threads can interfere with each other when sharing hardware resources such as caches or translation lookaside buffers (TLBs). As a result, execution times of a single thread are not improved and can be degraded.

THANK YOU

Any Queries ?

ankitbhattarai@cosmoscollege.edu.np