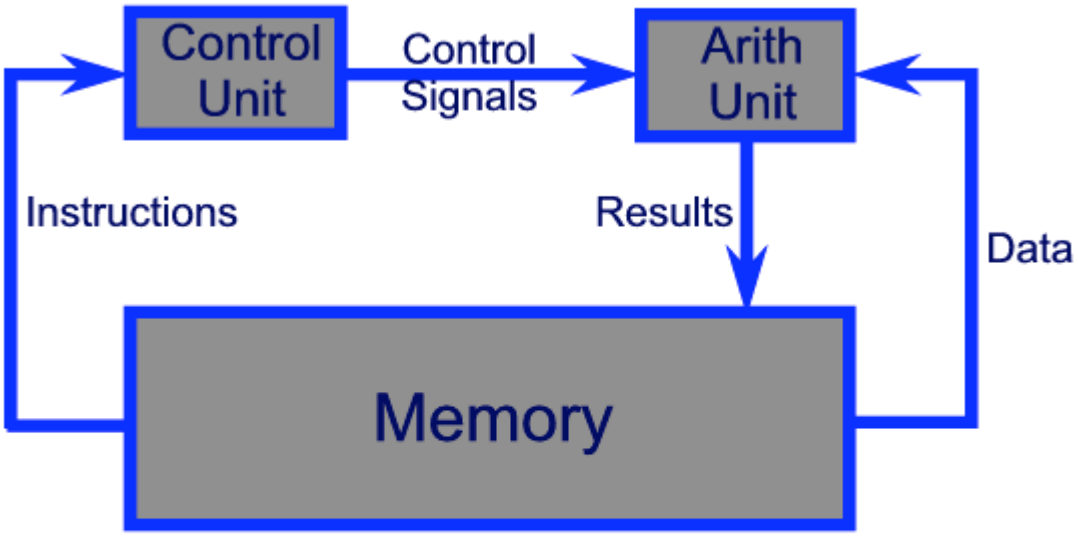
STARTING POINT:

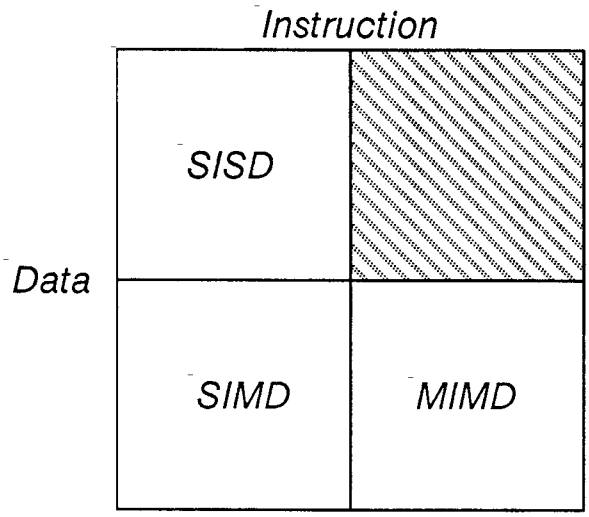


SISD (von Neumann) Architecture

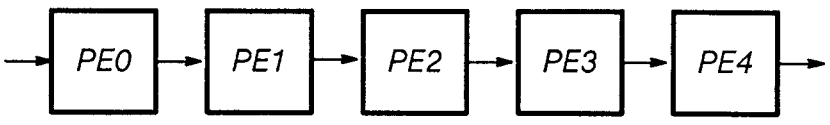
GOALS OF PARALLEL COMPUTING:

- High Performance – exploitation of inherent parallelism
- Fault Tolerance – exploitation of inherent redundancy
- Natural Solution to “Distributed Problems”
- Graceful Degradation and Growth

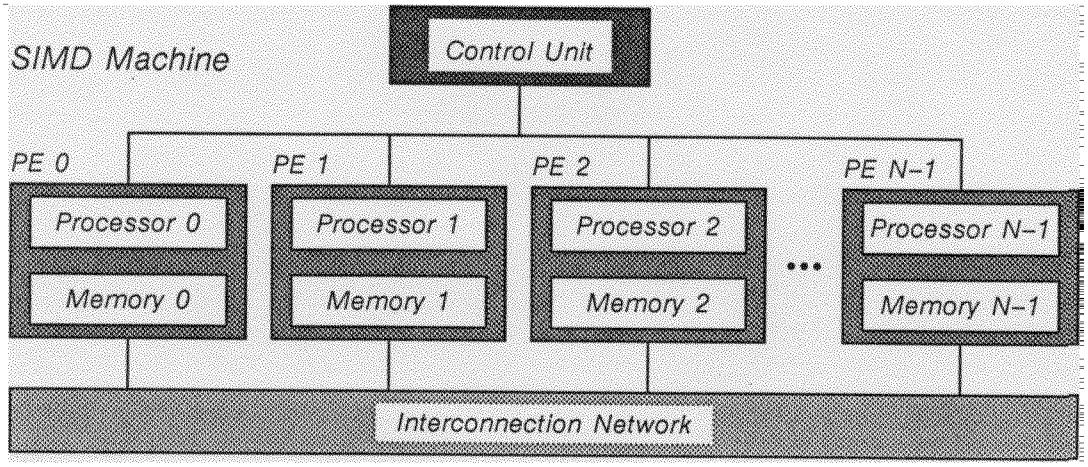
TYPES OF COMPUTER ARCHITECTURES:



NOTE: *Sometimes the designation MISD is given incorrectly to the pipeline of processors:*

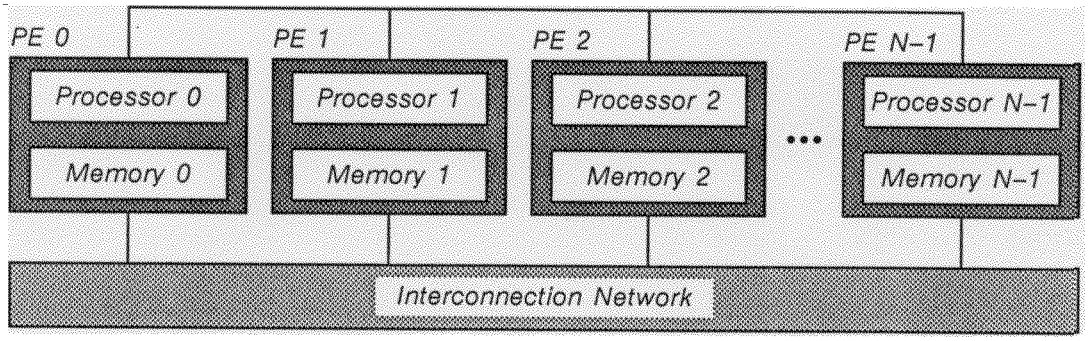


SIMD ARCHITECTURE:



- Single instruction stream, multiple data stream
- PE: Processing element (i.e. processor-memory pair)
- All PEs execute synchronously the same instructions using private data
- Instructions are broadcast globally by a single control unit
- Single control thread, single program
- Machine examples: AMT DAP, CLIP-4, CM-2, MasPar MP-1, MPP

MIMD ARCHITECTURE:



- Multiple instruction stream, multiple data stream
- PE: Processing element (i.e. processor-memory pair)
- Each PE executes asynchronously its own instructions using private data
- Multiple control threads, multiple collaborating programs
- Machine examples: BBN Butterfly, Cedar, CM-5, IBM RP3, Intel Cube, Ncube, NYU Ultracomputer

SIMD vs. MIMD: **SIMD Advantages:**

1. **Ease of programming and debugging:**
   SIMD: Single program, PEs operate synchronously
   MIMD: Multiple communicating programs,
   PEs operate asynchronously

2. **Overlap loop control with operations:**
   SIMD: Control unit does increment and compare,
   while PEs "compute"
   MIMD: Each PE does both

3. **Overlap operations on shared data:**
   SIMD: Control unit overlaps operations needed by all PEs
   (say, common array addresses)
   MIMD: Each PE does all the work

4. **Low PE-to-PE communications overhead:**
   SIMD: automatic synchronization of all "send" and "receive"
   operations
   MIMD: Explicit synchronization and identification protocol
   is needed

5. **Low synchronization overhead:**
   SIMD: Implicit in program
   MIMD: Explicit data structures and operations needed
   (semaphores, rendezvous, etc.)

6. **Lower program memory requirements:**
   SIMD: One copy of the program is stored
   MIMD: Each PE stores its own program

7. **Low instruction decoder cost:**
   SIMD: One decoder in control unit
   MIMD: One decoder in each PE

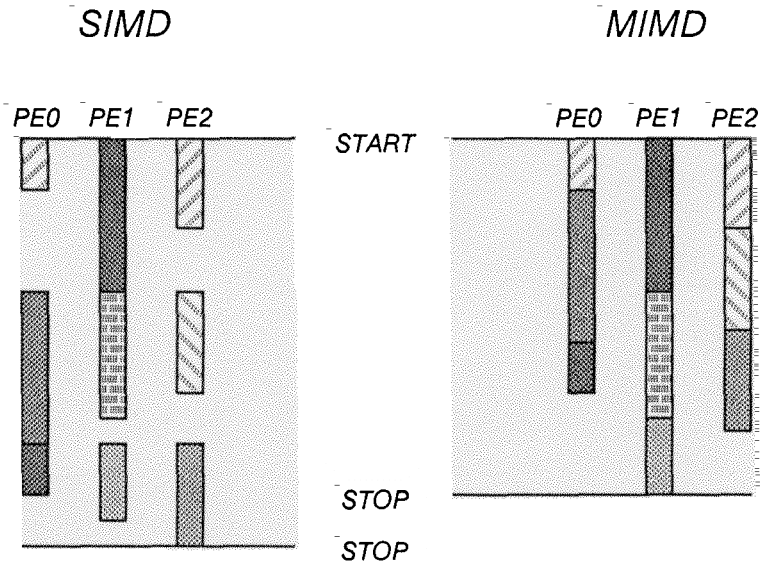
SIMD vs. MIMD: **MIMD Advantages:**

1. **Greater flexibility**: no constraints on operations that can be performed concurrently

2. **Conditional statements executed more efficiently:**
   MIMD: Each PE executes as if uniprocessor
   SIMD: Serialized execution of "then" and "else" clauses

3. **Efficient execution of variable-time instructions:**
   MIMD: Total execution time equals the maximum execution time on a given processor
   SIMD: Total execution time equals the sum of maximal execution times through all processors

EXAMPLE: Execution time of a program in which instruction execution times are data dependent:



SIMD EXAMPLES:

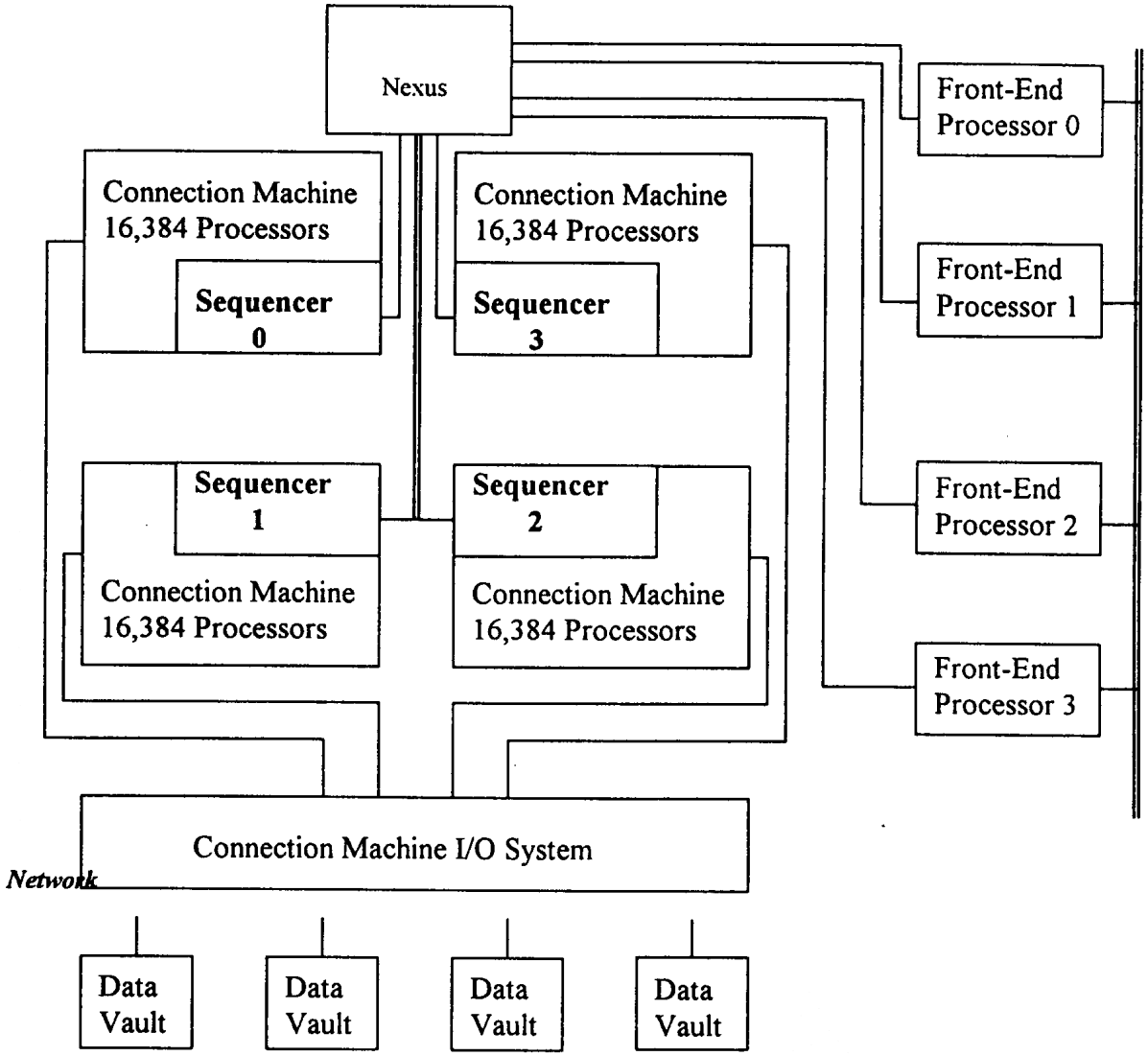


Figure 1.12. General architecture of SIMD CM-2.

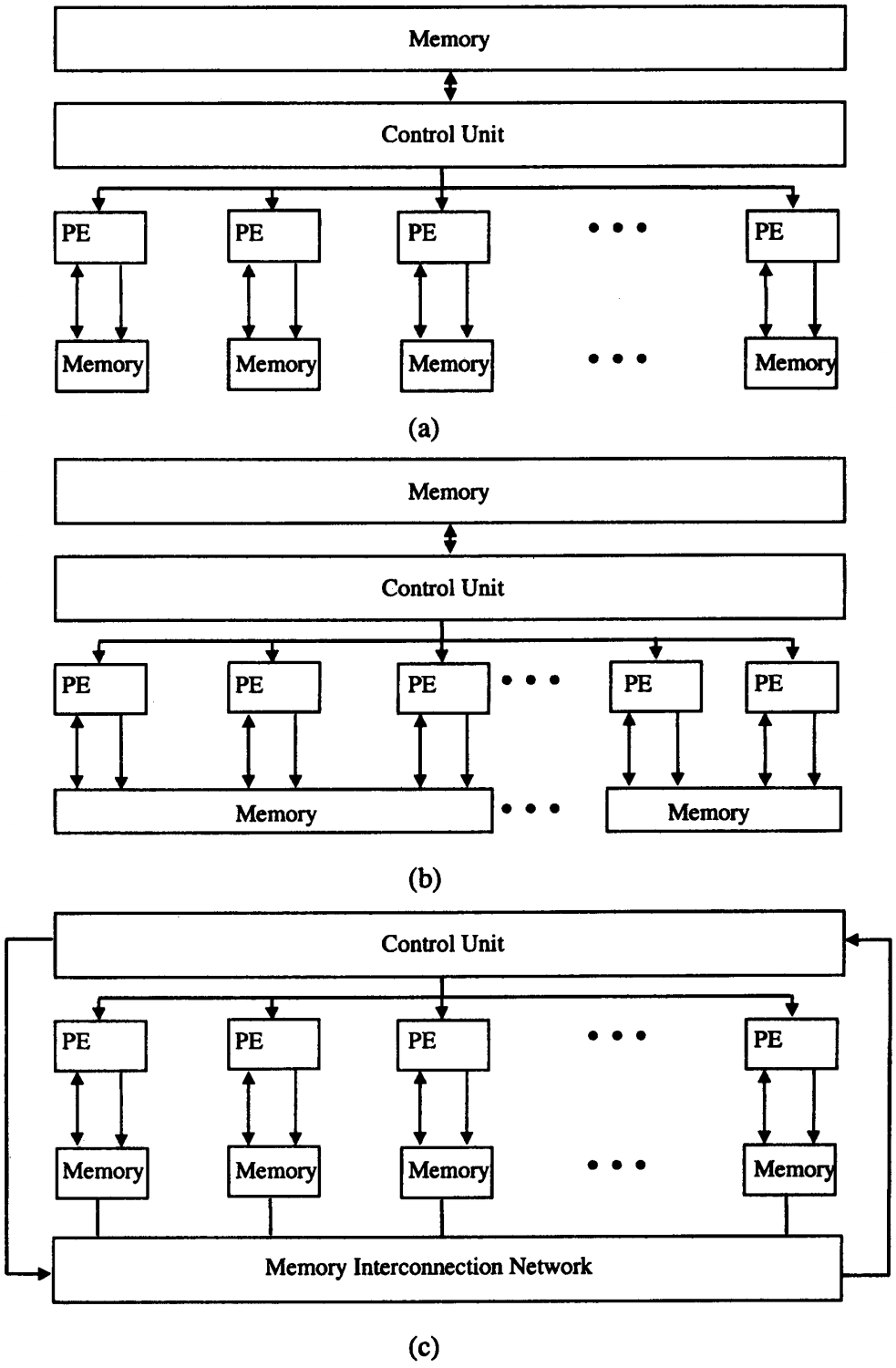
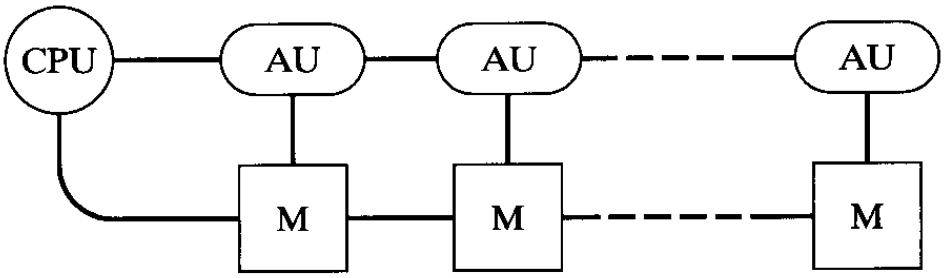


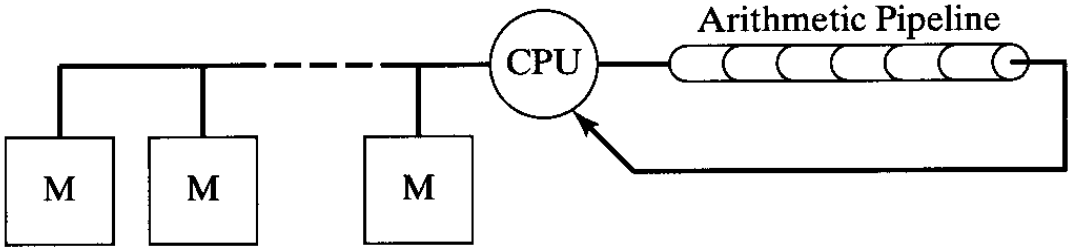
Figure 1.10. Three different SIMD processor array architectures. (a) Goodyear's Massively Parallel Processor. (b) Goodyear's STARAN. (c) ILLIAC IV.

NOTES OF TERMINOLOGY:

Some authors call pipelined machines SIMD machines, viz:



(a) True SIMD or vector computer

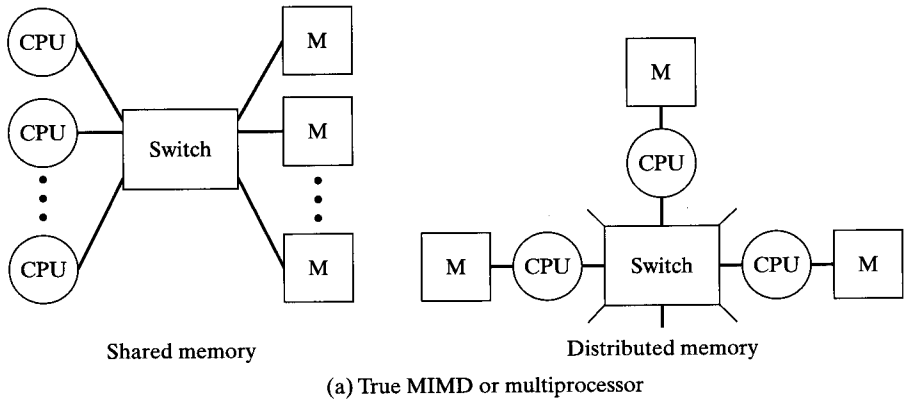


(b) Pipelined SIMD computer

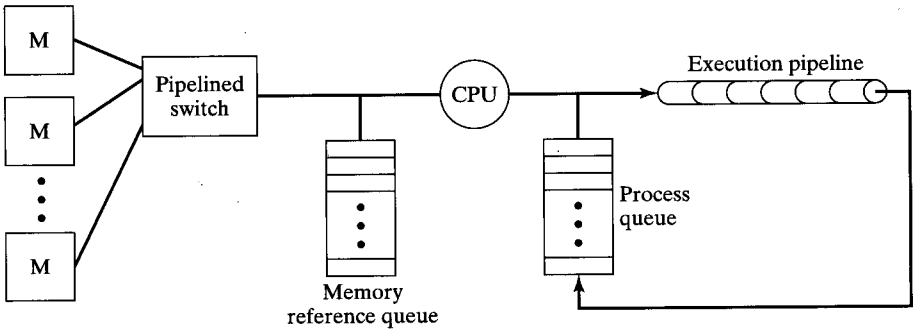
AU — Arithmetic unit CPU — Central processing unit M — Memory

MIMD EXAMPLES:

Various architectures are possible:



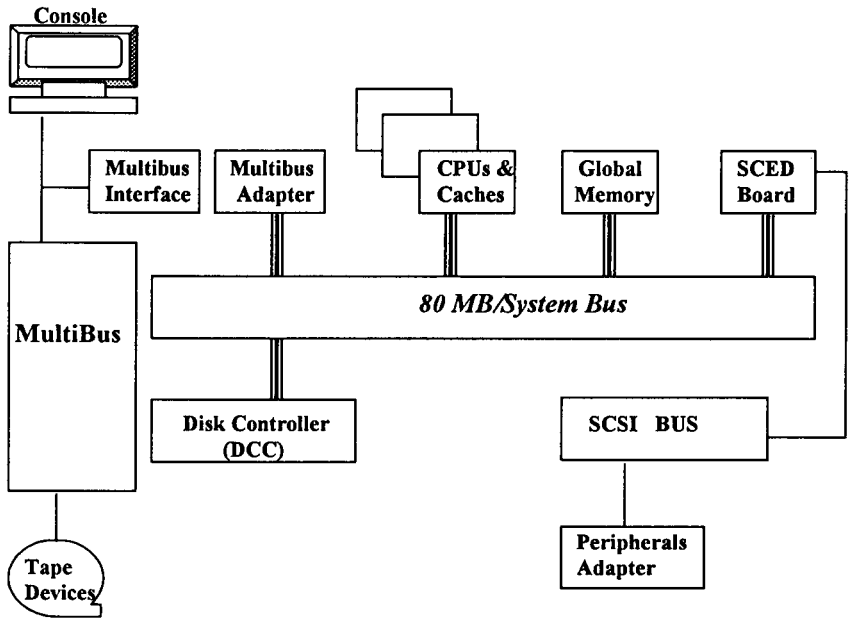
(a) True MIMD or multiprocessor



(b) Pipelined MIMD computer

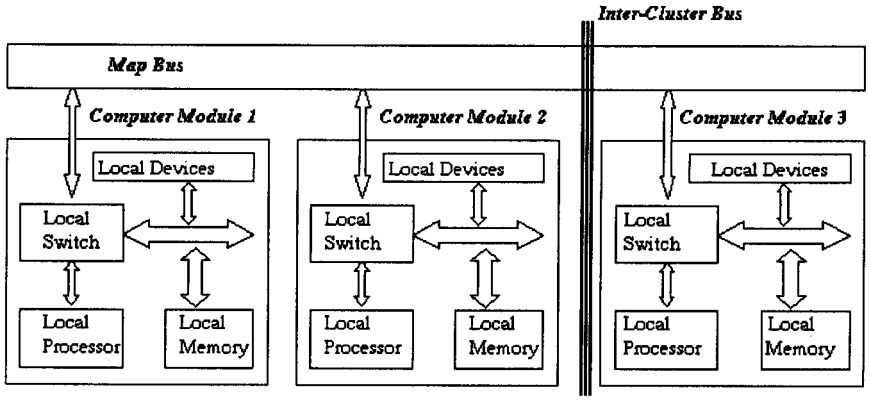
CPU — Central processing unit M — Memory

MIMD EXAMPLES (Cont'd.)



Sequent Symmetry multiprocessor architecture:

Shared memory machine with up to 30 CPUs based on Intel 80386/80387 chipset;
SCED = bus arbitrator, DCC = dual-channel disk controller



Carnegie-Mellon Multi-Mini-Processor, a.k.a. Cm* machine:

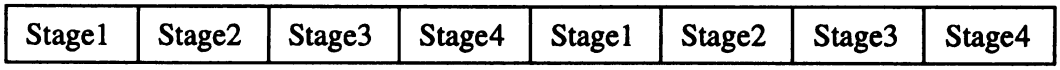
Processor-Memory pairs are called Computer Modules (CMs)
CMs are grouped into local clusters; Clusters are organized into a tree structure connected via Inter-Cluster buses.

PIPELINED PROCESSORS (sometimes called MISD machines)

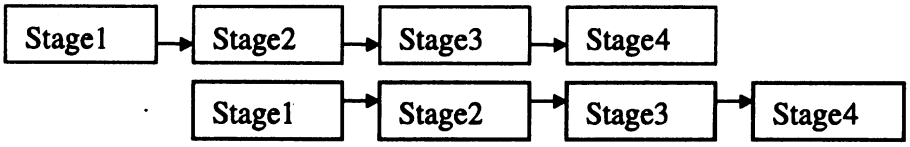
Principle of operation:

A process with 4 stages:

Serial execution of two processes consisting of 8 stages:



Pipelined execution of the same two processes consisting of 8 stages:



Total execution time of serial processing: $2*\sum_{i=1}^{4} S_i$

Total execution of pipelined processing: $1*\sum_{i=1}^{4} S_i+S_4$

where S_i is the execution time of stage i.

Assumptions: The computational process can be partitioned (i.e. segmented) into stages.

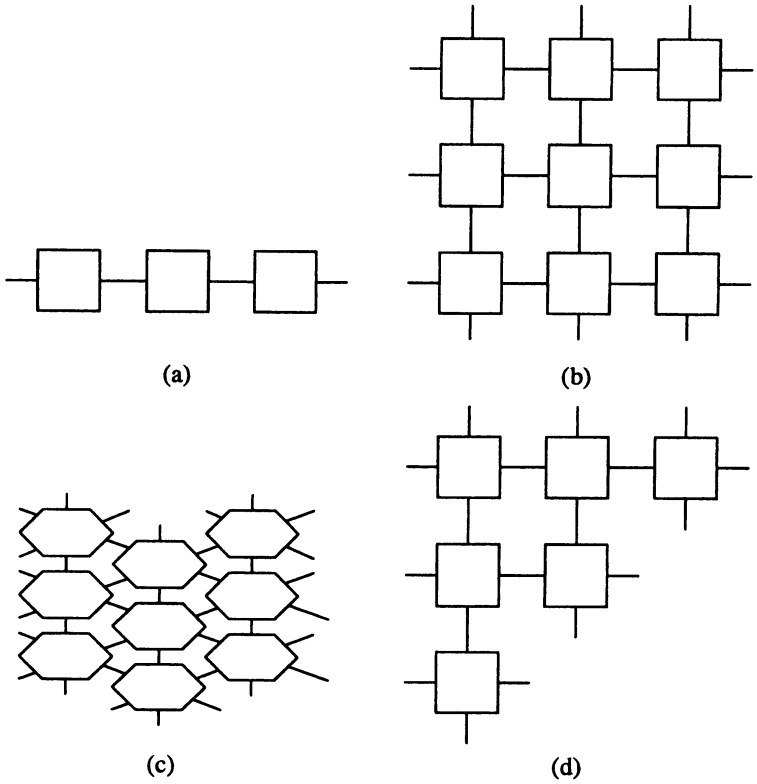
Pipelining can be exploited at various levels:

- **Instruction level**;
- **Subsystem level**: Pipelined arith. units (ADD, MUL, DIV, SORT) are found in many computers;
- **System level**: The pipeline segment need not be at the hardware level, but a software structure can form a pipeline.

SYSTOLIC ARRAYS

Generalization of the pipeline concept:

SYSTOLIC ARRAY = Network of locally connected functional units, operating synchronously with multidimensional pipelining, viz.:



a) Pipeline is a 1D systolic array
b) 2D systolic square array
c) 2D systolic hexagonal array
d) 2D systolic triangular array

Used when specific algorithms are mapped into fixed architectures to provide fast, massively-parallel computations. Offers good performance for special applications like image or signal processing. Otherwise of limited applicability and difficult to program.

SYSTOLIC ARRAY PROCESSOR ARCHITECTURE: