

# Parallel Processing

# Multiple Processor Organization

- Single instruction, single data stream - SISD
- Single instruction, multiple data stream - SIMD
- Multiple instruction, single data stream - MISD
- Multiple instruction, multiple data stream- MIMD
  - **Intruccion**: Perintah/operasi yang dieksekusi oleh processor (operasi matematika, manipulasi data, transfer data)
  - **Data stream**: aliran data, nilai atau informasi yang mengalir melalui system computer (seperti bilangan, karakter)

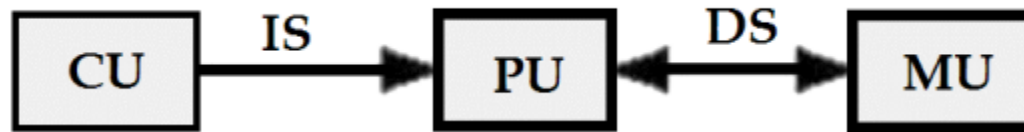


# Single Instruction, Single Data Stream - SISD

- Single processor
  - menggunakan satu prosesor utama untuk melakukan semua operasi pemrosesan.
- Single instruction stream
  - satu aliran instruksi dijalankan pada satu waktu.
  - Instruksi dieksekusi secara berurutan, satu per satu, untuk menyelesaikan tugas-tugas yang diberikan kepada CPU.
- Data stored in single memory
  - data yang diperlukan oleh CPU disimpan dalam satu memori tunggal.
- Uni-processor
  - hanya ada satu unit pemrosesan utama



# Parallel Organizations - SISD



CU : Control Unit  
MU : Memory Unit  
IS : Instruction Stream  
DS : Data Stream

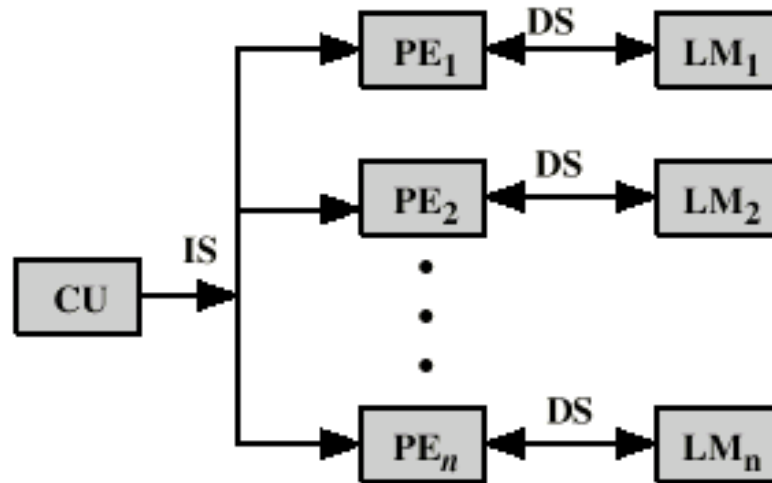


# Single Instruction, Multiple Data Stream - SIMD

- Single machine instruction
  - Instruksi yang sama diberikan kepada semua unit pemrosesan
- Number of processing elements
- Lockstep basis
  - menjalankan instruksi secara bersamaan dan secara teratur diatur untuk tetap dalam sinkronisasi.
- Each processing element has associated data memory
  - Setiap unit pemrosesan memiliki memori data yang terkait dengannya sendiri.
- Each instruction executed on different set of data by different processors
  - setiap prosesor mungkin bekerja pada data yang berbeda dalam rangkaian data yang diberikan.
- Vector and array processors
  - dirancang khusus untuk memproses larik data dalam operasi-instruksi yang sama secara paralel.
- Menggunakan banyak processor dengan intruksi yang sama, namun setiap processor mengolah data yang berbeda.



# Parallel Organizations - SIMD



CU : Control Unit  
MU : Memory Unit  
IS : Instruction Stream  
DS : Data Stream  
PE : Processing Element  
PU : Processing Unit  
LM : Local Memory

- Contoh: Penambahan dua vector:

- Intruksi Tunggal “+”

- Operasi penambahan dilakukan secara simultan menggunakan intruksi yang sama

A = [1, 2, 3, 4, 5]  
B = [6, 7, 8, 9, 10]

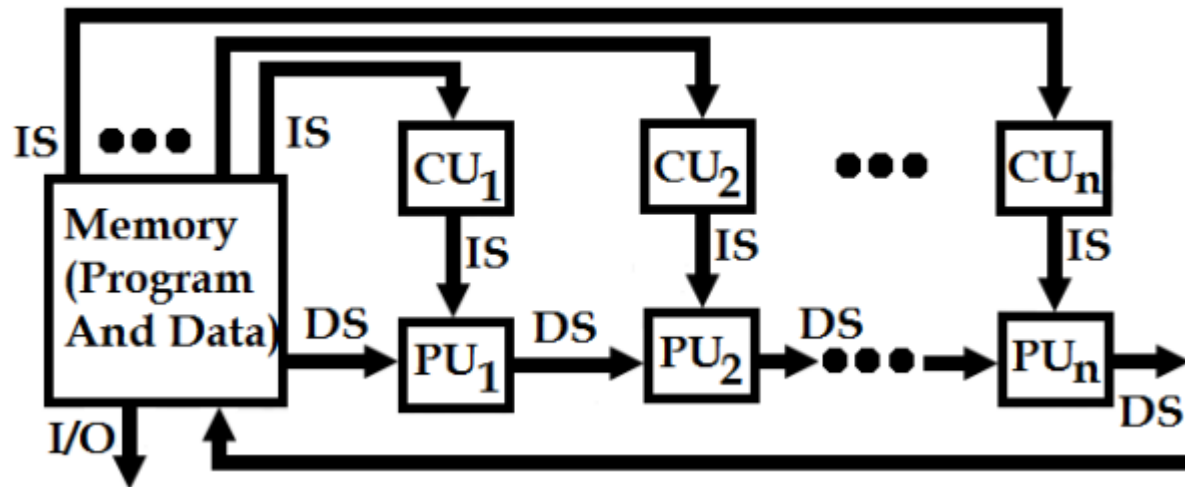


# Multiple Instruction, Single Data Stream - MISD

- Menggunakan banyak processor dengan setiap processor menggunakan intruksi yang berbeda namun mengolah data yang sama.
- Sequence of data
  - Aliran data Tunggal dikirim ke sekelompok processor
  - Setiap processor dapat menerima intruksi yang berbeda untuk diproses
- Transmitted to set of processors
  - Setiap prosessor menerima aliran data yang sama tetapi dapat menjalankan intruksi yg berbeda
- Each processor executes different instruction sequence
  - masing-masing prosesor dalam sistem MISD dapat menjalankan urutan instruksi yang berbeda.
- Never been implemented



# Multiple Instruction, Single Data Stream - MISD



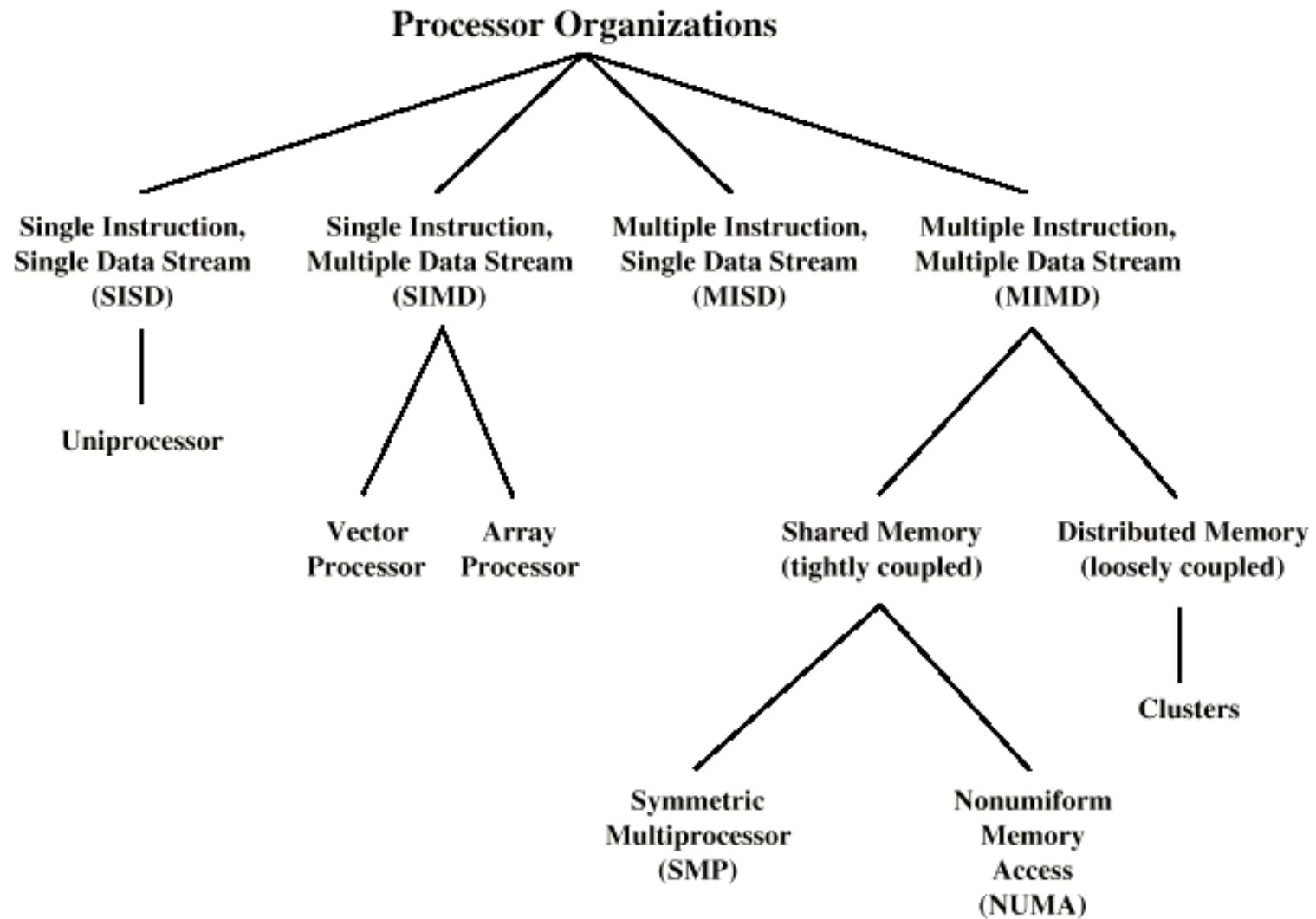


# Multiple Instruction, Multiple Data Stream- MIMD

- Set of processors
- Setiap prosesor dalam arsitektur MIMD dapat menjalankan urutan instruksi yang berbeda secara simultan.
- Different sets of data
  - setiap prosesor dalam sistem MIMD dapat memproses set data yang berbeda dari yang lain.
- SMPs, clusters and NUMA systems
  - SMPs (Symmetric Multiprocessing Systems) adalah sistem dengan beberapa prosesor yang terhubung ke memori umum dan diatur secara simetris.
  - Clusters adalah kumpulan komputer independen yang dihubungkan melalui jaringan dan beroperasi bersama-sama sebagai satu sistem.
  - NUMA (Non-Uniform Memory Access) systems adalah sistem dengan beberapa prosesor yang terhubung ke memori dengan akses yang berbeda-beda



# Taxonomy of Parallel Processor Architectures



# MIMD - Overview

- General purpose processors
- Each can process all instructions necessary
  - kemampuan untuk menjalankan semua instruksi yang diperlukan untuk menyelesaikan berbagai tugas komputasi.
- Dapat diklasifikasikan lebih lanjut berdasarkan metode komunikasi antar prosesor.



# Tightly Coupled - SMP

- Processors share memory
- Communicate via that shared memory
- Symmetric Multiprocessor (SMP)
  - Share single memory or pool
  - Shared bus to access memory
  - Memory access time to given area of memory is approximately the same for each processor



# Tightly Coupled - NUMA

- Nonuniform memory access (NUMA)
- Access times to different regions of memory may differ

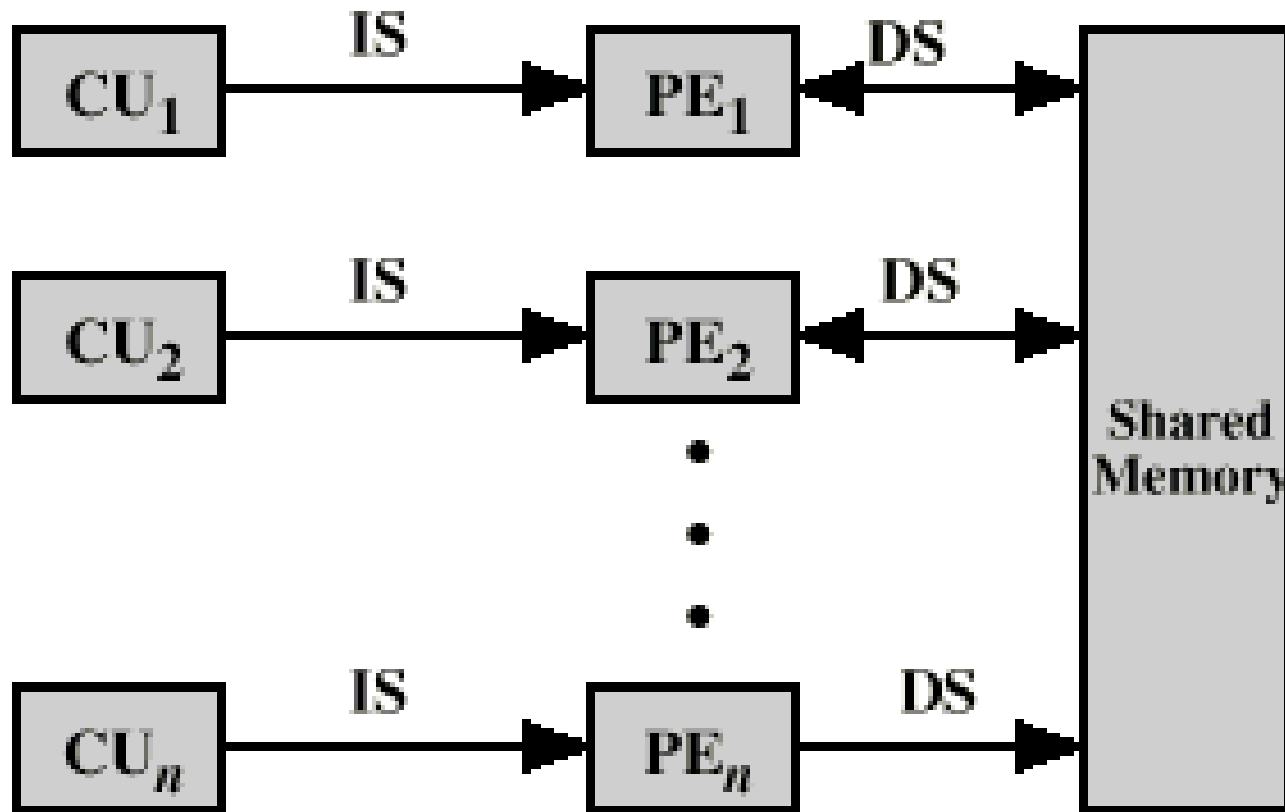


# Loosely Coupled - Clusters

- Collection of independent uniprocessors or SMPs
- Interconnected to form a cluster
- Communication via fixed path or network connections
  - Melalui bus terhubung langsung
  - LAN dan WAN

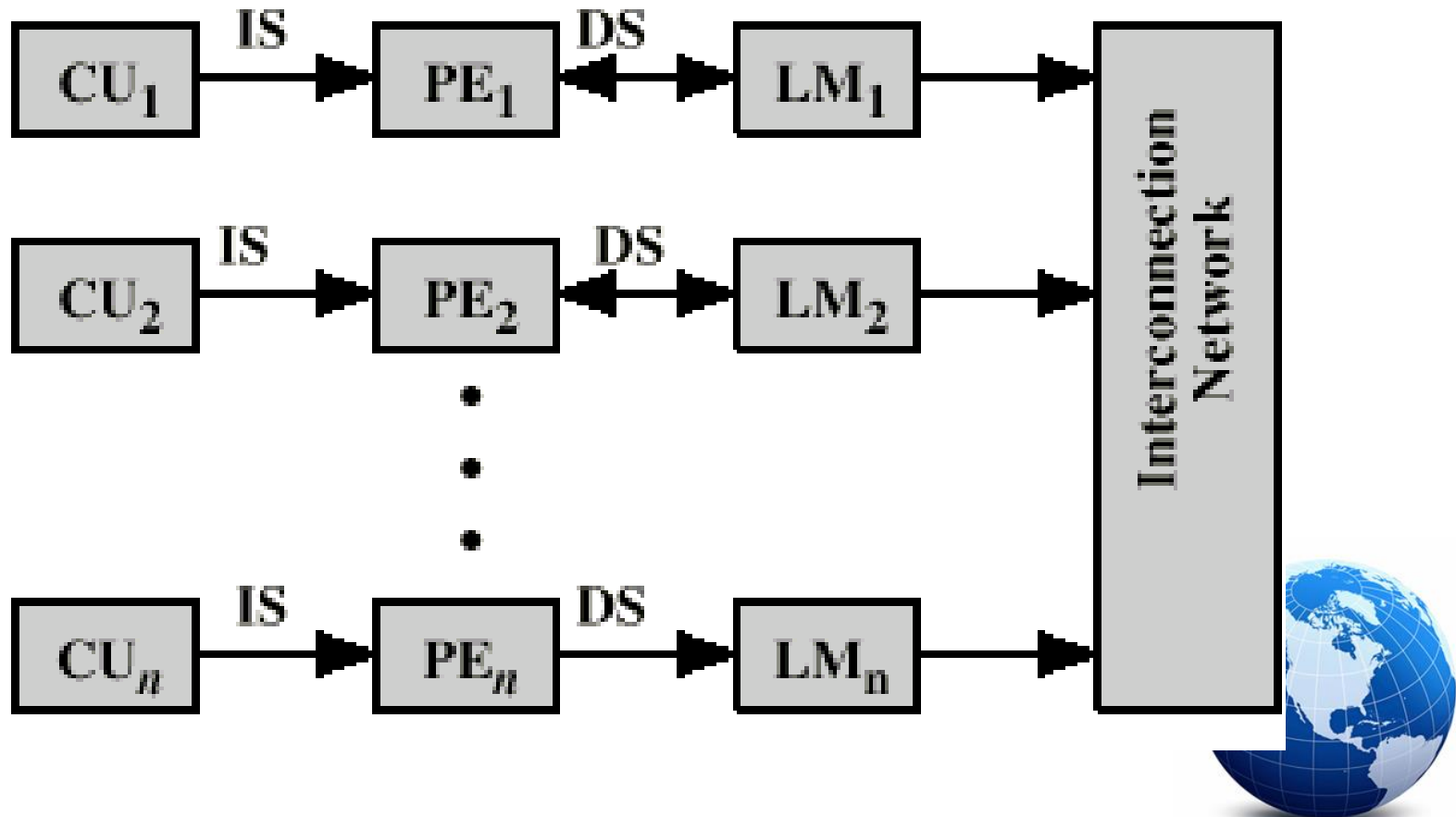


# Parallel Organizations - MIMD Shared Memory



# Parallel Organizations - MIMD

## Distributed Memory



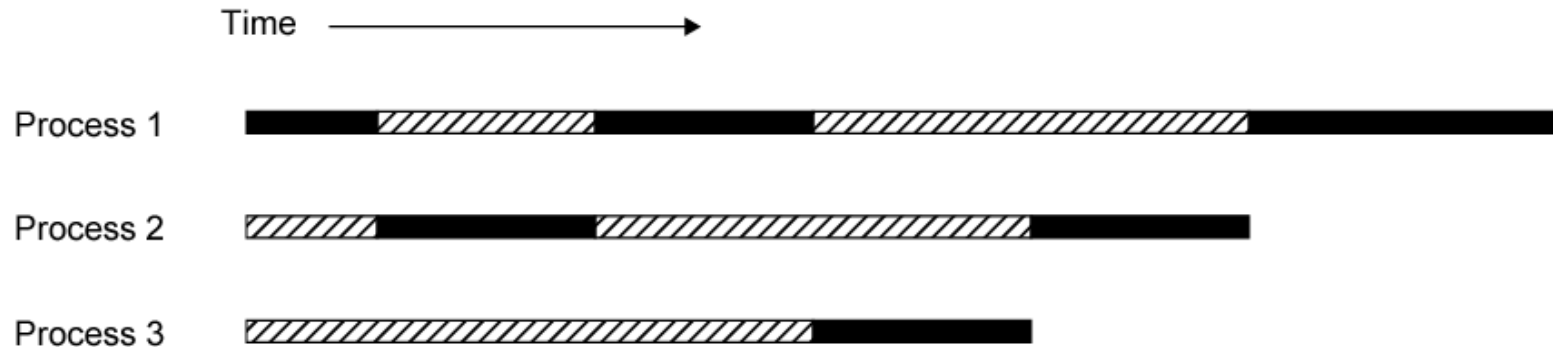


# Symmetric Multiprocessors (SMP)

- A stand alone computer with the following characteristics
  - Two or more similar processors of comparable capacity (serupa)
  - Processors share same memory and I/O
  - Processors are connected by a bus or other internal connection
  - Memory access time is approximately the same for each processor
  - All processors share access to I/O
    - Either through same channels or different channels giving paths to same devices
  - All processors can perform the same functions (hence symmetric)
  - System controlled by integrated operating system
    - providing interaction between processors
    - Interaction at job, task, file and data element levels



# Multiprogramming and Multiprocessing



(a) Interleaving (multiprogramming, one processor)



(b) Interleaving and overlapping (multiprocessing; multiple processors)

 Blocked       Running

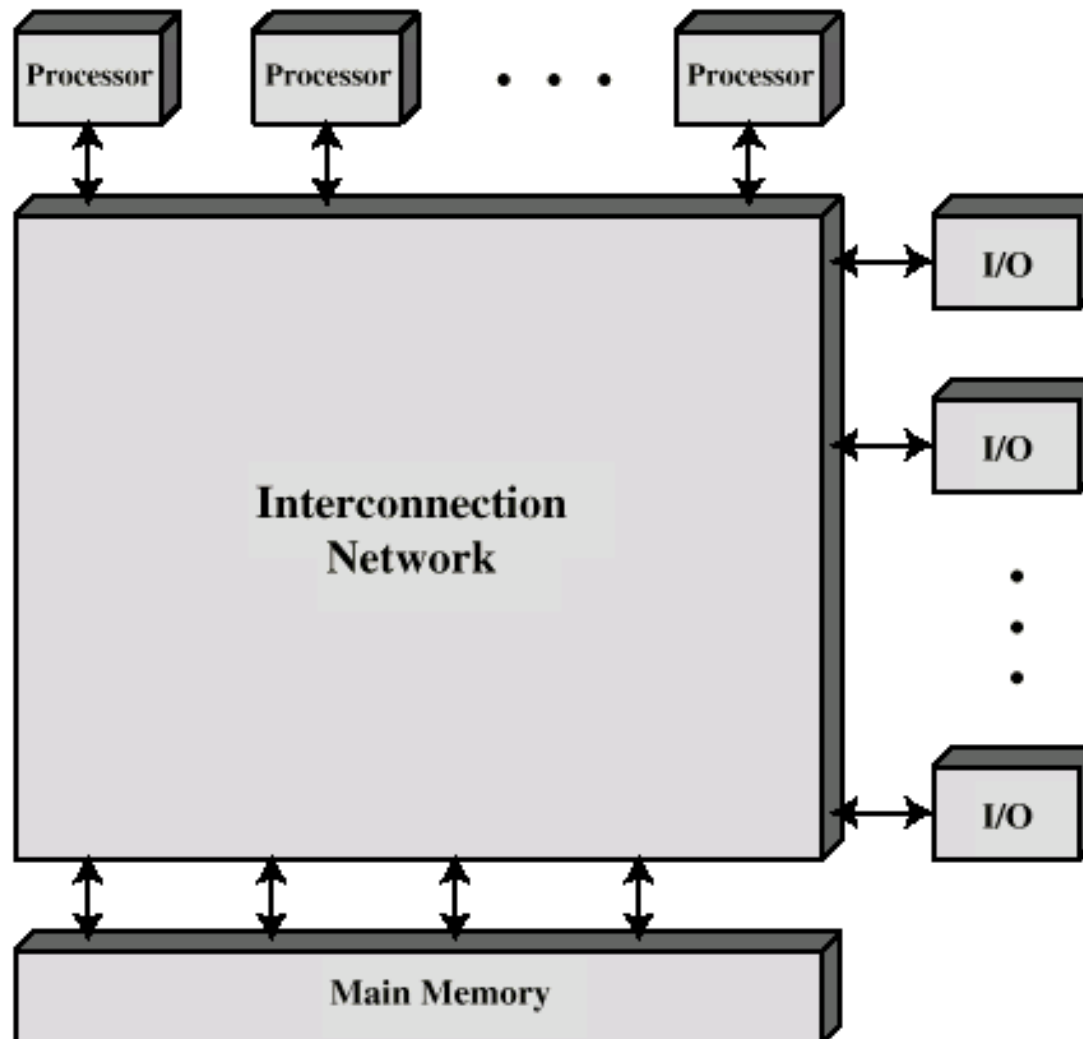


# SMP (Symmetric Multiprocessors ) Advantages

- Performance
  - Beberapa pekerjaan dapat diselesaikan secara parallel.
- Availability
  - Karena semua prosesor dapat menjalankan fungsi yang sama, kegagalan satu prosesor tidak menghentikan sistem
- Incremental growth
  - User can enhance performance by adding additional processors
- Scaling
  - Vendor dapat menawarkan berbagai produk berdasarkan jumlah prosesor



# Block Diagram of Tightly Coupled Multiprocessor



# Organization Classification-SMP

- Time shared or common bus
  - Penggunaan share waktu atau bus bersama
- Multiport memory
  - Memori memiliki beberapa jalur akses
- Central control unit
  - Pengaturan dan pengendalian terpusat

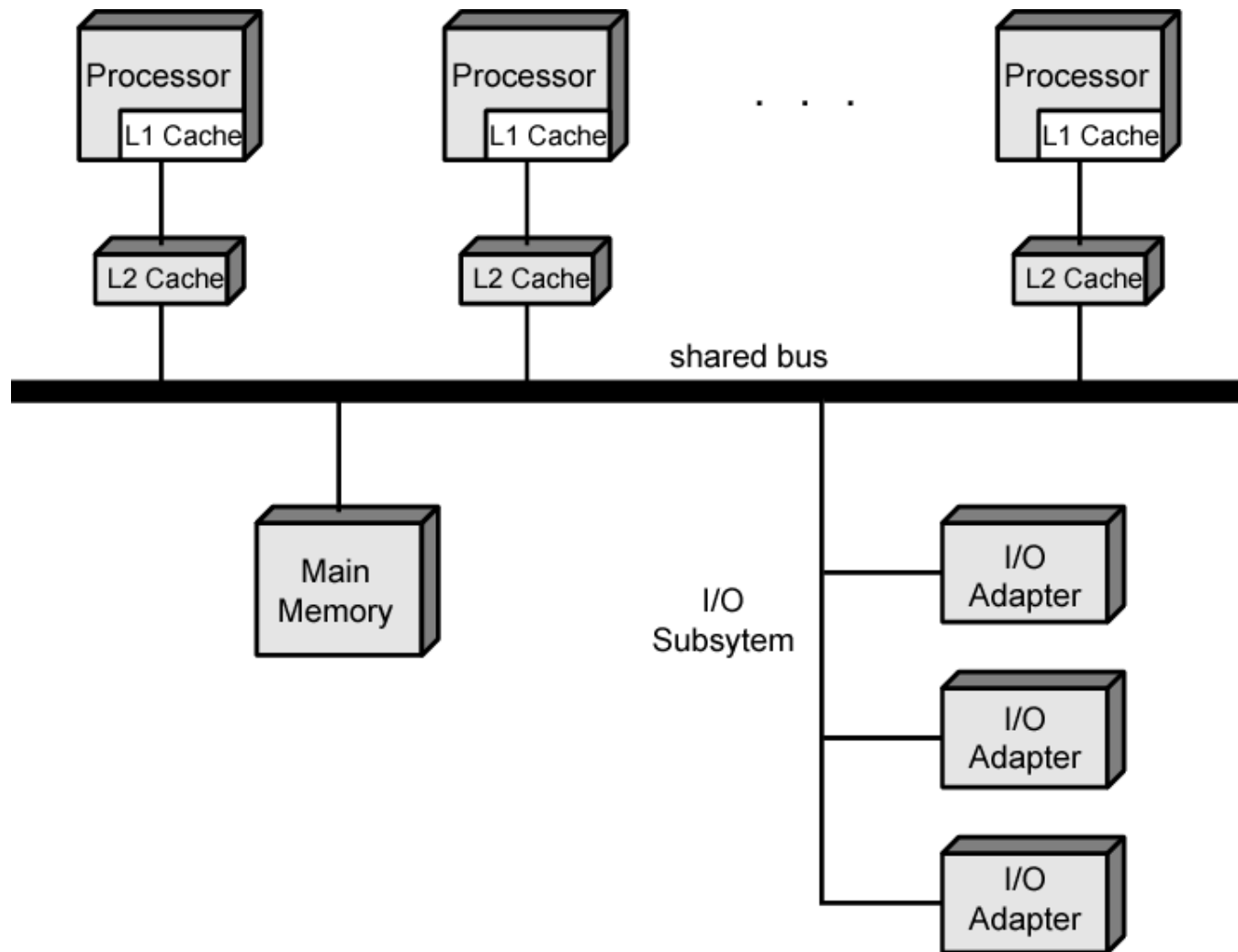


# Time Shared Bus

- Simplest form
- Structure and interface similar to single processor system
- Following features provided
  - Addressing - distinguish modules on bus
  - Arbitration - any module can be temporary master
  - Time sharing - if one module has the bus, **others must wait** and may have to suspend
- Now have multiple processors as well as multiple I/O modules



# Symmetric Multiprocessor Organization



# Time Share Bus - Advantages

- **Simplicity**
  - Sistem lebih sederhana karena berbagai komponen dapat berbagi jalur komunikasi yang sama.
- **Flexibility**
  - berbagai komponen dapat berbagi sumber daya secara fleksibel.
- **Reliability**
  - kontrol dan pengaturan dapat dilakukan secara lebih efisien, mengurangi kemungkinan konflik atau kegagalan komunikasi.





# Time Share Bus - Disadvantage

- Performance limited by bus cycle time
- Each processor should have local cache
  - Reduce number of bus accesses
- Leads to problems with cache coherence
  - Solved in hardware - see later



# Operating System Issues

- Simultaneous concurrent processes
  - kemampuan sistem operasi untuk mengelola dan menjalankan beberapa proses secara bersamaan
- Scheduling
  - sistem operasi untuk menentukan urutan eksekusi proses-proses yang bersaing untuk menggunakan sumber daya komputer.
- Synchronization
  - memastikan koordinasi yang tepat antara proses-proses yang berjalan secara bersamaan.
- Memory management
- Reliability and fault tolerance
  - menjaga kehandalan sistem dan memberikan toleransi terhadap kesalahan.

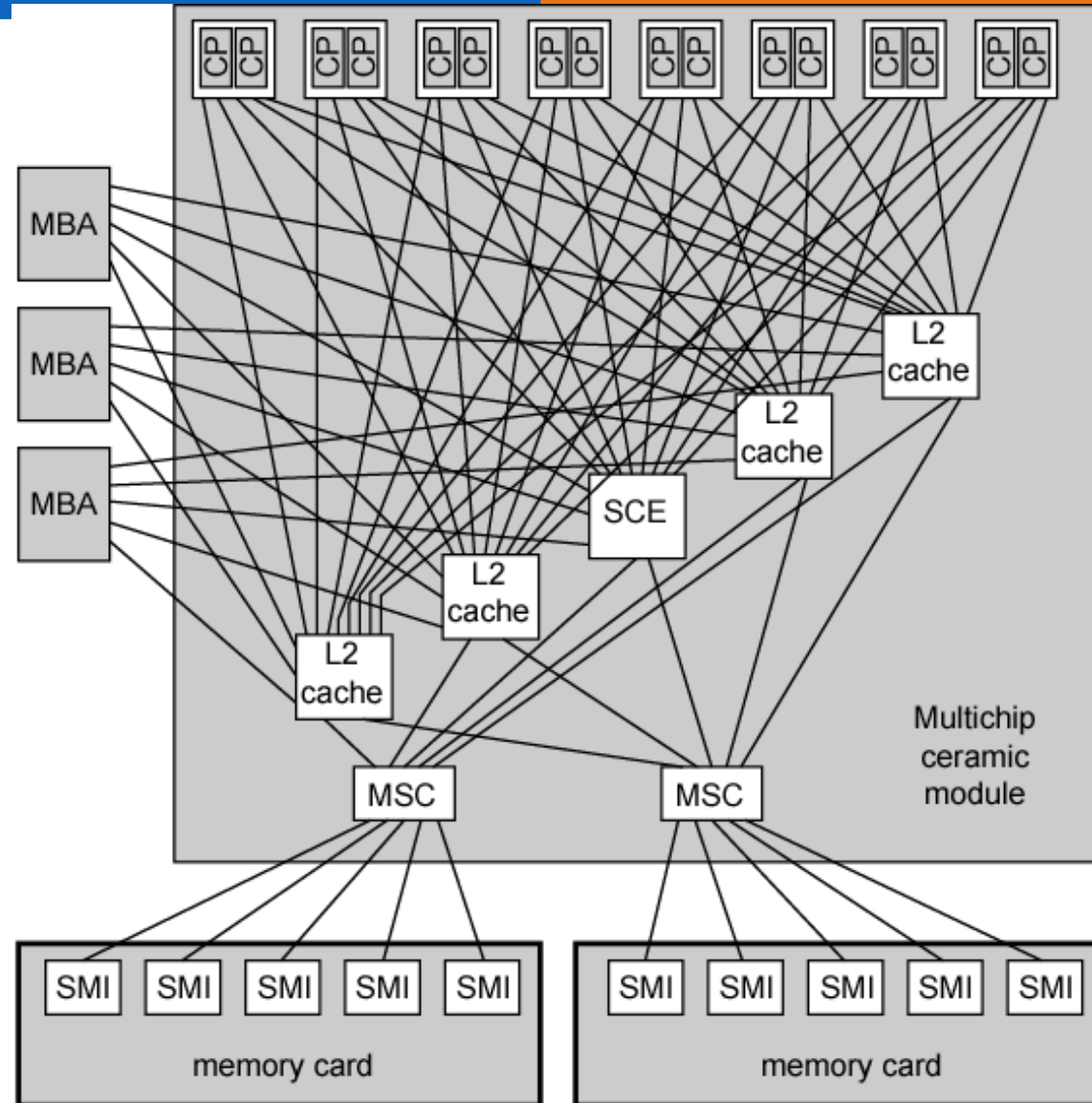


# A Mainframe SMP: IBM zSeries

- Uniprocessor with one main memory card to a high-end system with 48 processors and 8 memory cards
- Dual-core processor chip (dua CPU dalam satu chip)
  - Each includes two identical central processors (CPs)
  - CISC superscalar microprocessor
  - Mostly hardwired, some vertical microcode
  - 256-kB L1 instruction cache and a 256-kB L1 data cache
- L2 cache 32 MB
  - Clusters of five
  - Each cluster supports eight processors and access to entire main memory space
- System control element (SCE)
  - Arbitrates system communication
  - Maintains cache coherence
- Main store control (MSC)
  - Interconnect L2 caches and main memory
- Memory card
  - Each 32 GB, Maximum 8 , total of 256 GB
  - Interconnect to MSC via synchronous memory interfaces (SMIs)
- Memory bus adapter (MBA)
  - Interface to I/O channels, go directly to L2 cache



# IBM z990 Multiprocessor Structure



CP = central processor  
MBA = memory bus adapter  
MSC = main store control  
SCE = system control element  
SMI = synchronous memory interface

# Cache Coherence and MESI Protocol

- Cache Coherence: mekanisme sistem multiprosesor untuk memastikan bahwa salinan data yang disimpan di cache setiap prosesor tetap konsisten.
- MESI (Modified, Exclusive, Shared, dan Invalid) Protocol adalah protokol untuk mencapai cache coherence
- Problem - multiple copies of same data in different caches
- Can result in an inconsistent view of memory



# Software Solutions

- Compiler and operating system deal with problem
- Overhead transferred to compile time
  - Beban tambahan dapat dilakukan di compile time, tidak di waktu eksekusi
- Kompleksitas desain ditransfer dari hardware to software
- However, software cenderung pemanfaatan cache yang tidak efisien.
- Analyze code to determine safe periods for caching shared variables



# Hardware Solution

- Cache coherence protocols
  - semua cache yang terhubung ke memori utama memiliki salinan data yang konsisten.
- Dynamic recognition of potential problems
- Run time (pada saat program berjalan)
- More efficient use of cache
- Transparent to programmer
- Directory protocols
  - suatu metode yang digunakan untuk mengelola koherensi cache dalam sistem multiprosesor
- Snoopy protocols
  - setiap cache diawasi oleh "snoopy agent" yang memonitor transaksi memori dan mengambil tindakan yang sesuai untuk mempertahankan koherensi.



# Directory Protocols-Untuk mengelola koherensi cache

- Collect and maintain information about copies of data in cache
  - Protokol direktori mengumpulkan dan memelihara informasi tentang salinan data yang ada di dalam cache setiap prosesor dalam sistem multiprosesor.
- Directory stored in main memory
- Requests are checked against directory
- Appropriate transfers are performed
- Creates central bottleneck
- Effective in large scale systems with complex interconnection schemes





# Snoopy Protocols-Untuk mengelola koherensi cache

- Protokol snoopy mendistribusikan tanggung jawab koherensi cache di antara pengontrol cache (cache controllers) yang ada di setiap prosesor
- Cache recognizes that a line is shared
  - Setiap cache dalam sistem memantau status setiap baris (line) data dalam cache, apakah data tersebut dibagikan (shared) dengan prosesor lain atau tidak.
- Updates announced to other caches
  - Ketika sebuah prosesor memodifikasi data yang dibagikan, protokol snoopy memungkinkan cache controller untuk mengumumkan pembaruan tersebut kepada cache controller di prosesor lain yang juga menyimpan salinan data yang sama.
- Suited to bus based multiprocessor
  - Protokol snoopy lebih cocok digunakan dalam sistem multiprosesor yang berbasis bus, di mana semua prosesor terhubung ke bus yang sama untuk mengakses memori utama dan perangkat I/O.
- Increases bus traffic
  - Salah satu konsekuensi dari penggunaan protokol snoopy adalah peningkatan lalu lintas (traffic) pada bus sistem.



# Write Invalidate

- Multiple readers, one writer: beberapa prosesor untuk membaca data dari cache (multiple readers), tetapi hanya satu prosesor yang dapat menulis data ke cache pada suatu waktu (one writer).
- When a write is required, all other caches of the line are invalidated
  - Ketika suatu prosesor perlu menulis data ke cache, semua cache lain yang menyimpan salinan data yang sama (cache dari prosesor lain) akan dinonaktifkan (invalidated).
- Writing processor then has exclusive (cheap) access until line required by another processor
  - Setelah menulis data ke cache dan menginvalidasi cache-cache lain, prosesor yang menulis memiliki akses eksklusif ke data tersebut sampai dibutuhkan oleh prosesor lain.
- Used in Pentium II and PowerPC systems
- State of every line is marked as modified, exclusive, shared or invalid
- MESI



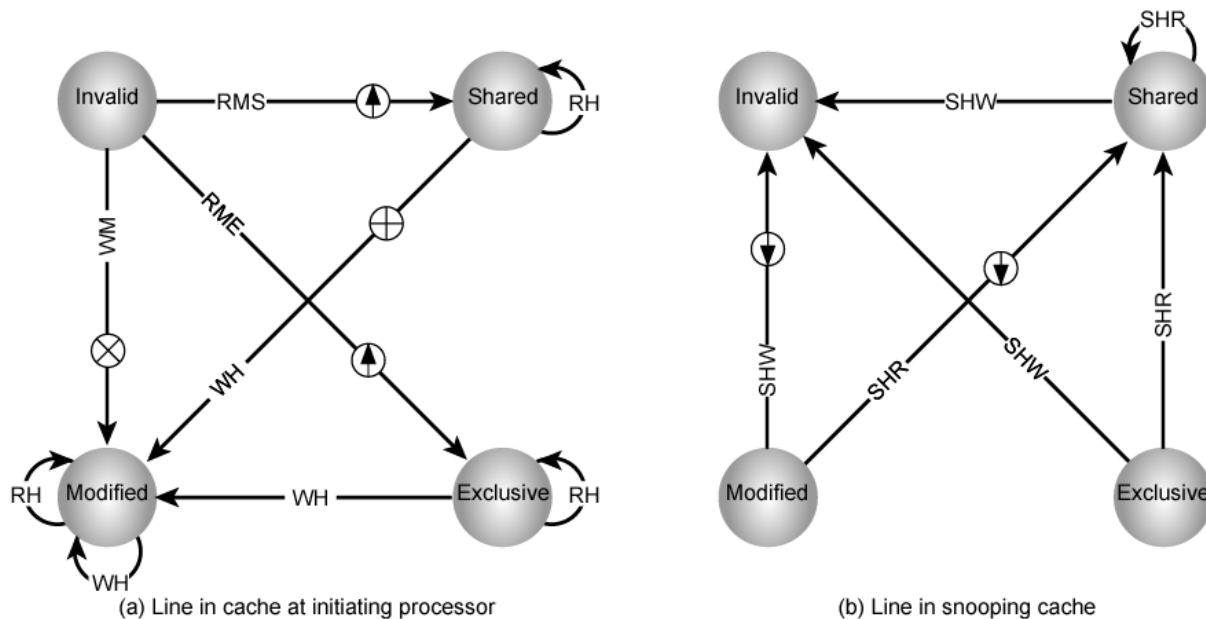
# Write Update





- Multiple readers and writers
  - beberapa prosesor untuk membaca dan menulis data ke cache secara bersamaan.
- Updated word is distributed to all other processors
  - setiap prosesor yang terlibat memiliki salinan terbaru dari data yang sama setelah pembaruan dilakukan.
- Some systems use an adaptive mixture of both solutions



# MESI State Transition Diagram

- Protokol MESI, yang menggambarkan bagaimana status sebuah blok cache berubah berdasarkan operasi yang dilakukan oleh prosesor.



|     |  |   |                            |
|-----|--|---|----------------------------|
| RH  | Read hit   |  | Dirty line copyback        |
| RMS | Read miss, shared                                |   |                            |
| RME | Read miss, exclusive                             |  | Invalidate transaction     |
| WH  | Write hit  |   |                            |
| WM  | Write miss                                       |   |                            |
| SHR | Snoop hit on read                                |  | Read-with-intent-to-modify |
| SHW | Snoop hit on write or read-with-intent-to-modify |  | Cache line fill            |



# Increasing Performance

- Processor performance can be measured by the rate at which it executes instructions
- MIPS (Million Instructions Per Second) rate =  $f * IPC$ 
  - $f$  processor clock frequency, in MHz
  - IPC is average instructions per cycle
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- May be reaching limit
  - Complexity
  - Power consumption



# Multithreading and Chip Multiprocessors

- Instruction stream divided into smaller streams (threads)
- Executed in parallel
- Wide variety of multithreading designs



# Definitions of Threads and Processes

- Thread dalam prosesor multithread mungkin sama atau mungkin tidak sama dengan thread perangkat lunak
- Process:
  - An instance of program running on computer
  - Resource ownership
    - Virtual address space to hold process image
  - Scheduling/execution
  - Process switch
- Thread: unit kerja yang dapat dikirim dalam proses
  - Includes processor context (which includes the program counter and stack pointer) and data area for stack
  - Thread executes sequentially
  - Interruptible: processor can turn to another thread
- Thread switch
  - Switching processor between threads within same process
  - Typically less costly than process switch



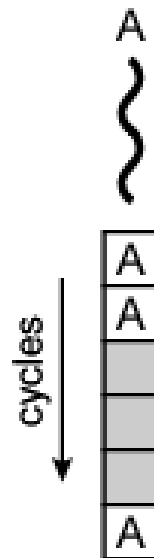
# Scalar Processor Approaches

- Single-threaded scalar
  - Simple pipeline
  - No multithreading
- Interleaved multithreaded scalar
  - Easiest multithreading to implement
  - Switch threads at each clock cycle
  - Pipeline stages kept close to fully occupied (berusaha penuh)
  - Hardware needs to switch thread context between cycles
- Blocked multithreaded scalar
  - Thread executed until latency event occurs
  - Would stop pipeline
  - Processor switches to another thread

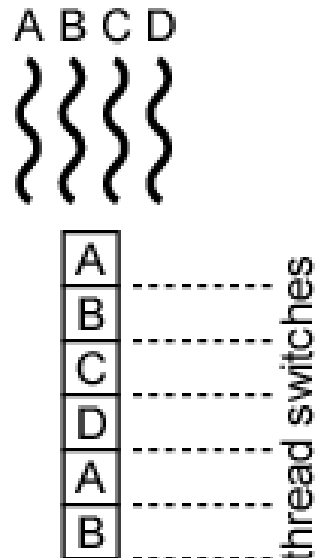




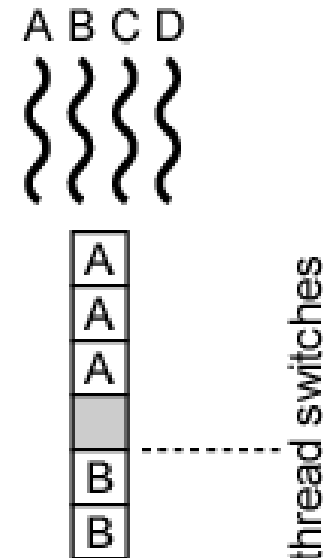
# Scalar Diagrams



(a) single-threaded scalar



(b) interleaved multithreading scalar



(c) blocked multithreading scalar

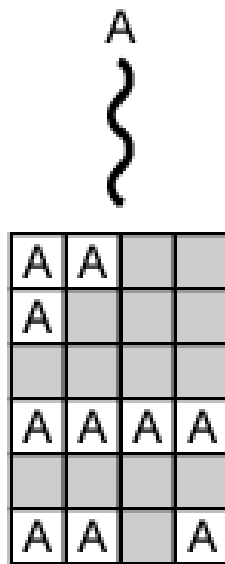


# Multiple Instruction Issue Processors (1)

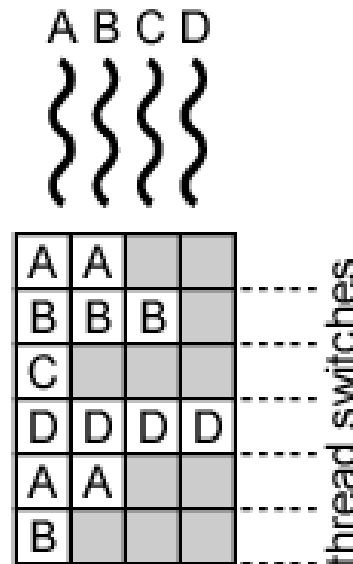
- Superscalar
  - No multithreading
- Interleaved multithreading superscalar:
  - Each cycle, as many instructions as possible issued from single thread
  - Delays due to thread switches eliminated
  - Number of instructions issued in cycle limited by dependencies
- Blocked multithreaded superscalar
  - Instructions from one thread
  - Blocked multithreading used



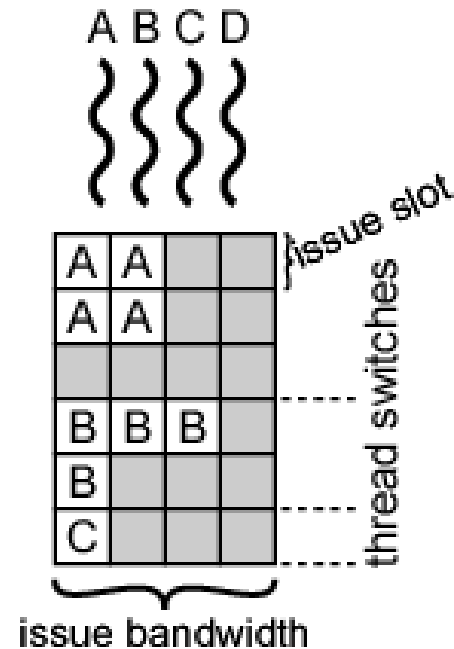
# Multiple Instruction Issue Diagram (1)



(d) superscalar



(e) interleaved  
multithreading  
superscalar



(f) blocked  
multithreading  
superscalar

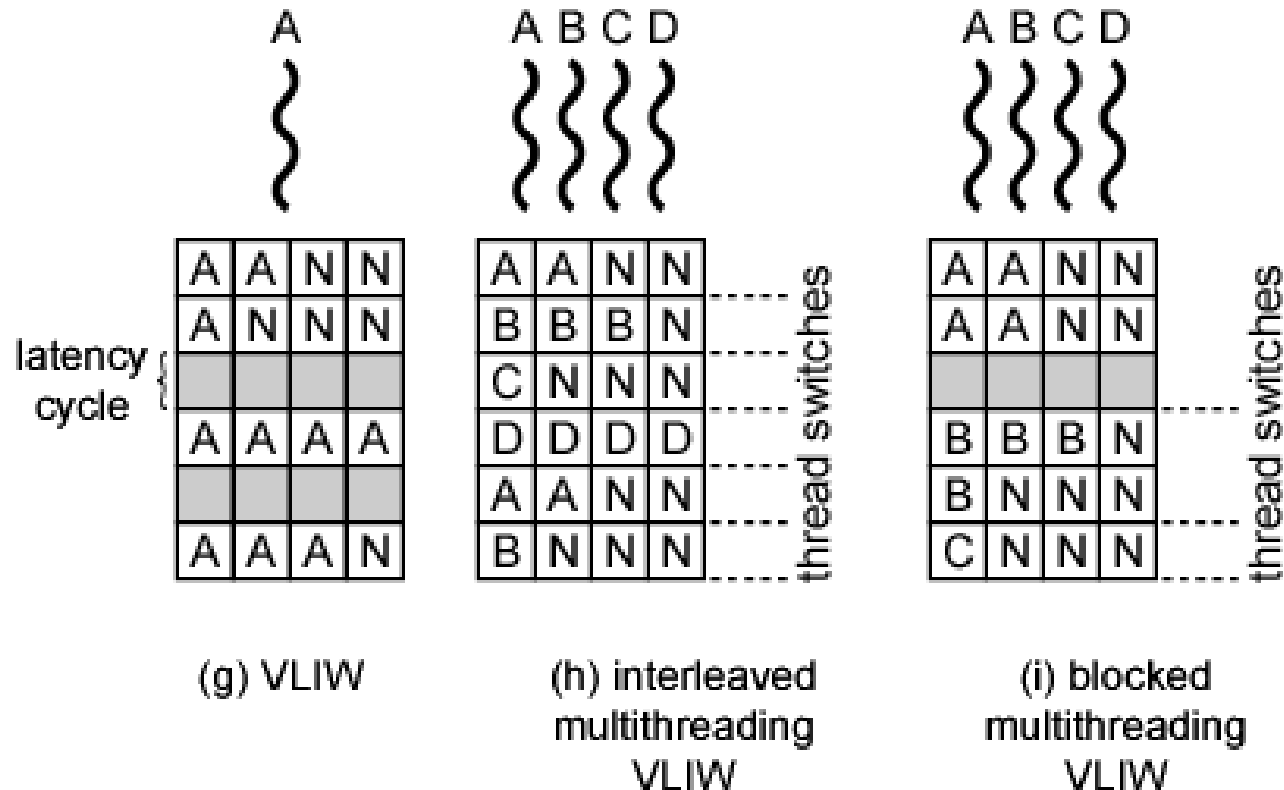


# Multiple Instruction Issue Processors (2)

- Very long instruction word (VLIW)
  - E.g. IA-64
  - Multiple instructions in single word
  - Typically constructed by compiler
  - Operations that may be executed in parallel in same word
  - May pad with no-ops
- Interleaved multithreading VLIW
  - Similar efficiencies to interleaved multithreading on superscalar architecture
- Blocked multithreaded VLIW
  - Similar efficiencies to blocked multithreading on superscalar architecture



# Multiple Instruction Issue Diagram (2)

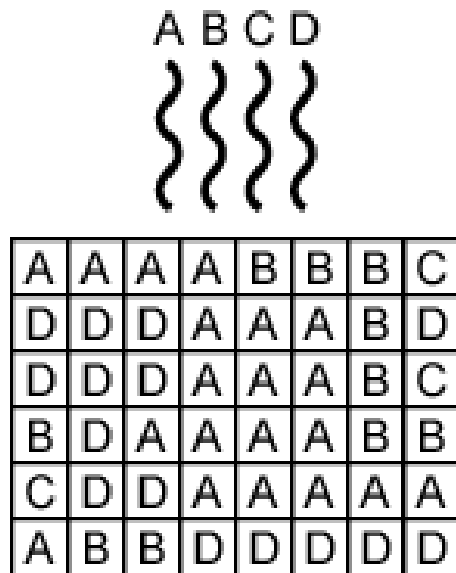


# Parallel, Simultaneous Execution of Multiple Threads

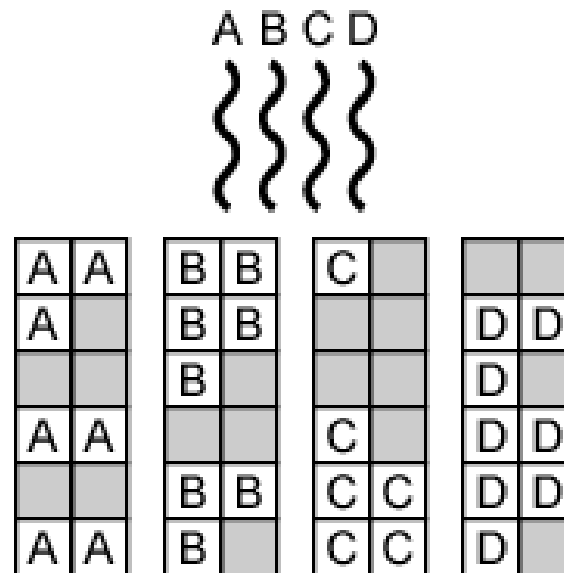
- Simultaneous multithreading
  - Issue multiple instructions at a time
  - One thread may fill all horizontal slots
  - Instructions from two or more threads may be issued
  - With enough threads, can issue maximum number of instructions on each cycle
- Chip multiprocessor
  - Multiple processors
  - Each has two-issue superscalar processor
  - Each processor is assigned thread
    - Can issue up to two instructions per cycle per thread



# Parallel Diagram



(j) simultaneous  
multithreading  
(SMT)



(k) chip multiprocessor



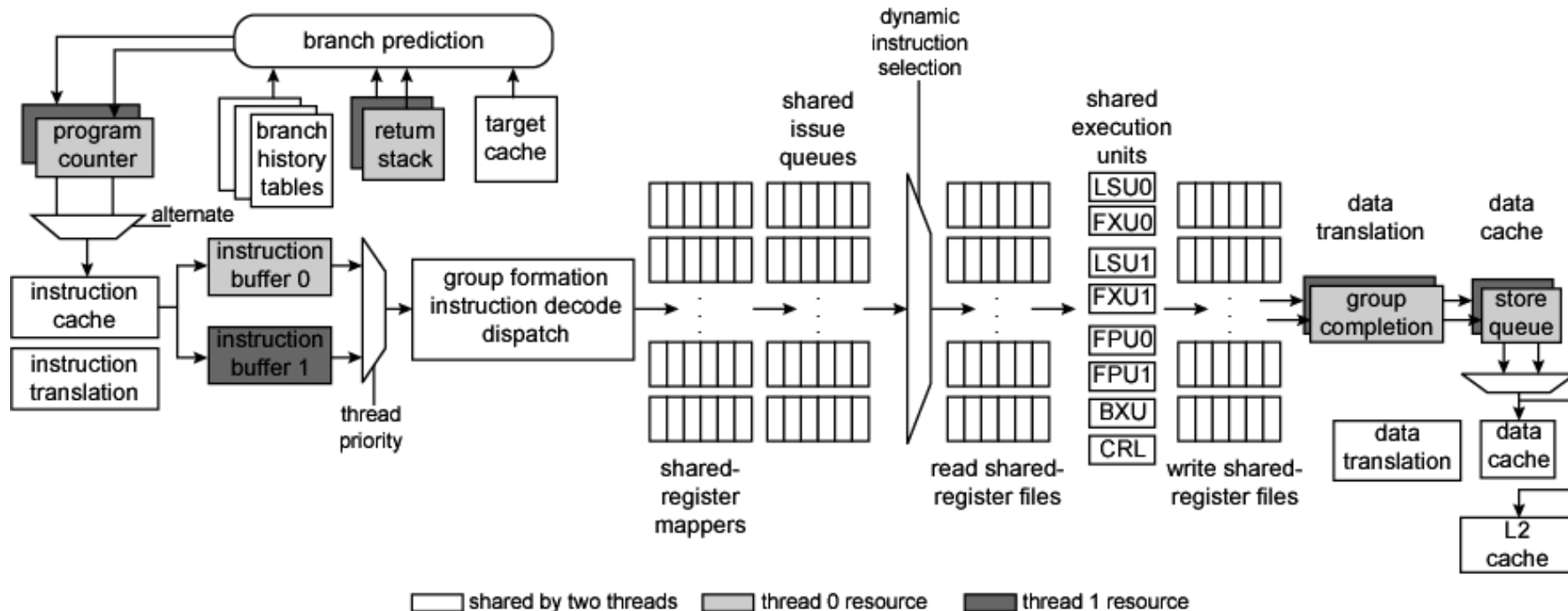
# Examples

- Some Pentium 4
  - Intel calls it hyperthreading
  - SMT with support for two threads
  - Single multithreaded processor, logically two processors
- IBM Power5
  - High-end PowerPC
  - Combines chip multiprocessing with SMT
  - Chip has two separate processors
  - Each supporting two threads concurrently using SMT





# Power5 Instruction Data Flow



BXU = branch execution unit and  
 CRL = condition register logical execution unit  
 FPU = floating-point execution unit  
 FXU = fixed-point execution unit  
 LSU = load/store unit

# Clusters

- Alternative to SMP
  - High performance
  - High availability
  - Server applications
- 
- A group of interconnected whole computers
  - Working together as unified resource
  - Illusion of being one machine
  - Each computer called a node

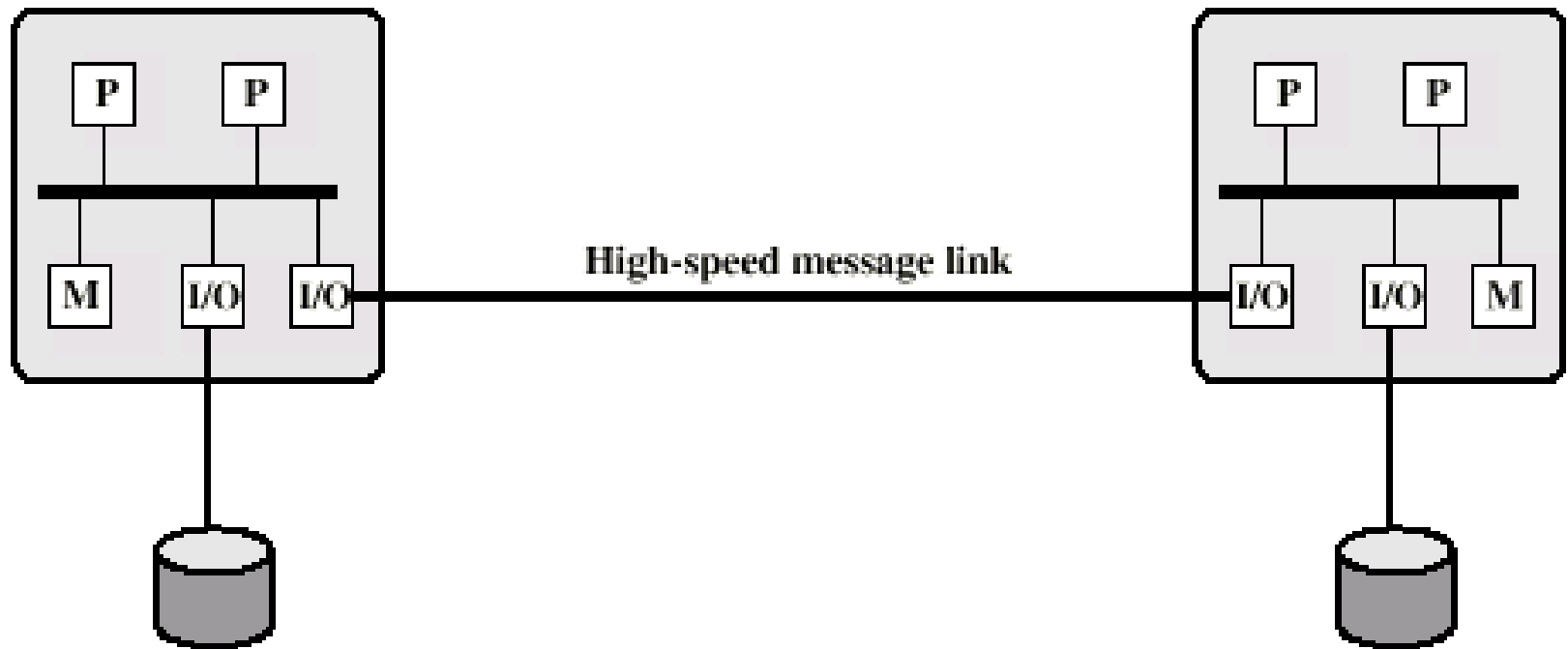


# Cluster Benefits

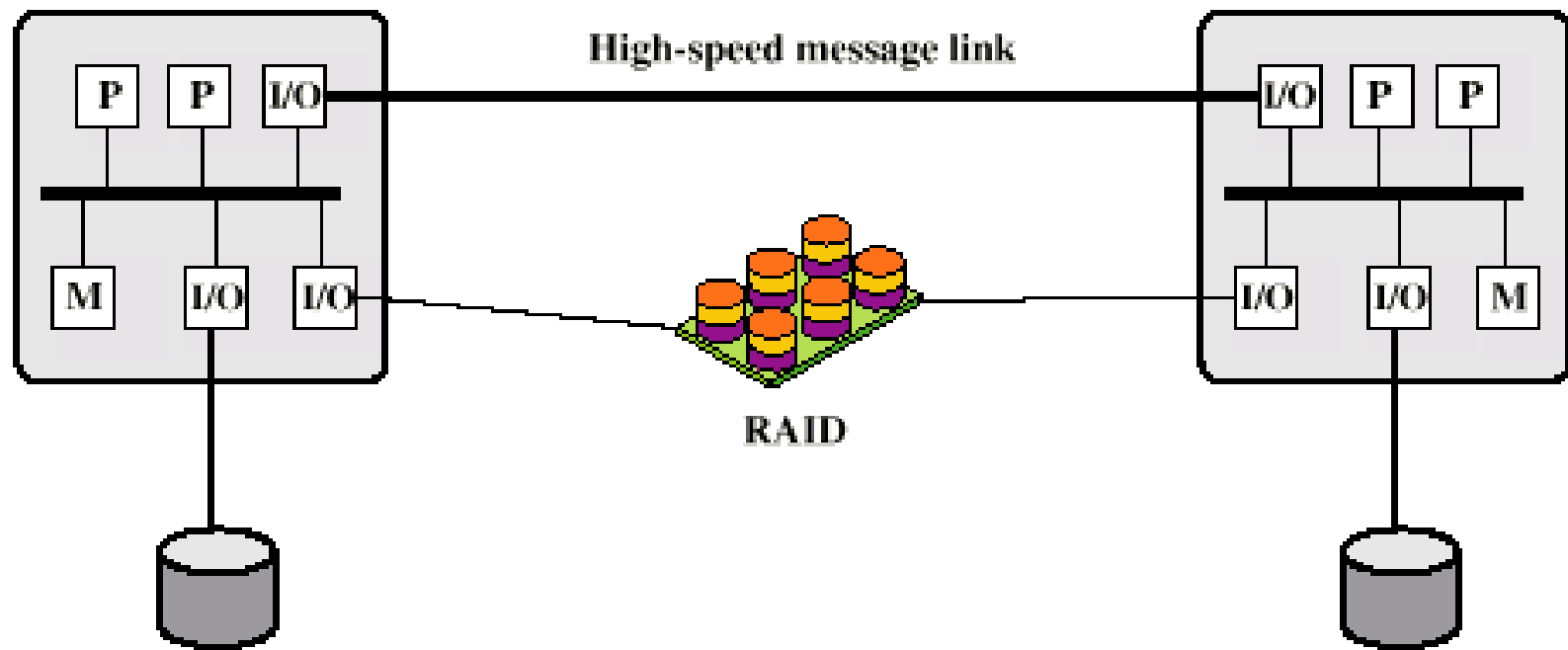
- Absolute scalability
- Incremental scalability
- High availability
- Superior price/performance



# Cluster Configurations - Standby Server, No Shared Disk



# Cluster Configurations - Shared Disk



# Operating Systems Design Issues

- Failure Management
  - High availability
  - Fault tolerant
  - Failover
    - Switching applications & data from failed system to alternative within cluster
  - Failback
    - Restoration of applications and data to original system
    - After problem is fixed
- Load balancing
  - Incremental scalability
  - Automatically include new computers in scheduling
  - Middleware needs to recognise that processes may switch between machines



# Parallelizing

- Single application executing in parallel on a number of machines in cluster
  - Compiler
    - Determines at compile time which parts can be executed in parallel
    - Split off for different computers
  - Application
    - Application written from scratch to be parallel
    - Message passing to move data between nodes
    - Hard to program
    - Best end result
  - Parametric computing
    - If a problem is repeated execution of algorithm on different sets of data
    - e.g. simulation using different scenarios
    - Needs effective tools to organize and run



# Cluster v. SMP

- Both provide multiprocessor support to high demand applications.
- Both available commercially
  - SMP for longer
- SMP:
  - Easier to manage and control
  - Closer to single processor systems
    - Scheduling is main difference
    - Less physical space
    - Lower power consumption
- Clustering:
  - Superior incremental & absolute scalability
  - Superior availability
    - Redundancy





# Nonuniform Memory Access (NUMA)

- Alternative to SMP & clustering
- Uniform memory access
  - All processors have access to all parts of memory
    - Using load & store
  - Access time to all regions of memory is the same
  - Access time to memory for different processors same
  - As used by SMP
- Nonuniform memory access
  - All processors have access to all parts of memory
    - Using load & store
  - Access time of processor differs depending on region of memory
  - Different processors access different regions of memory at different speeds
- Cache coherent NUMA
  - Cache coherence is maintained among the caches of the various processors
  - Significantly different from SMP and clusters

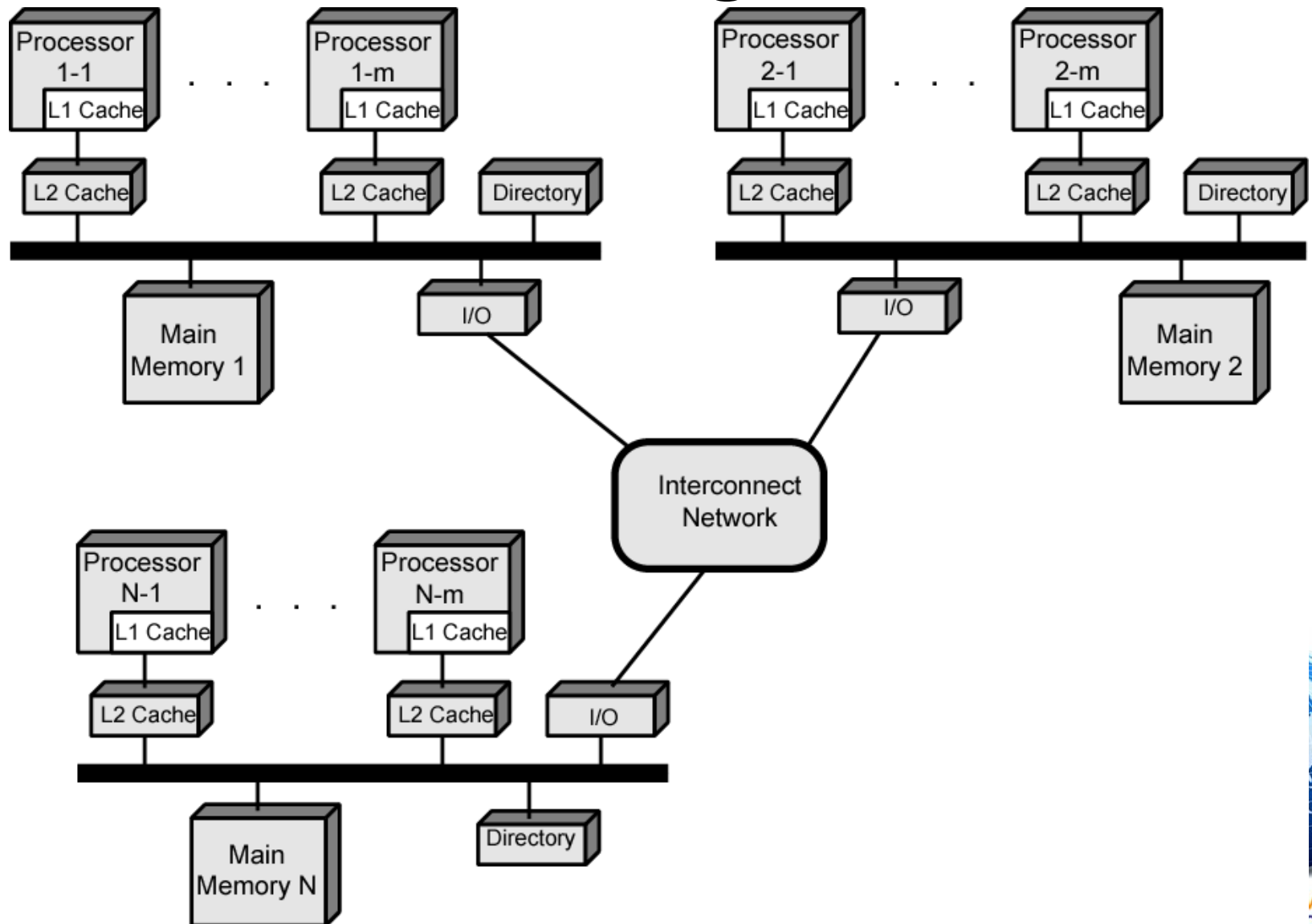


# Motivation

- SMP has practical limit to number of processors
  - Bus traffic limits to between 16 and 64 processors
- In clusters each node has own memory
  - Apps do not see large global memory
  - Coherence maintained by software not hardware
- NUMA retains SMP flavour while giving large scale multiprocessing
  - e.g. Silicon Graphics Origin NUMA 1024 MIPS R10000 processors
- Objective is to maintain transparent system wide memory while permitting multiprocessor nodes, each with own bus or internal interconnection system



# CC-NUMA Organization



# CC-NUMA Operation

- Each processor has own L1 and L2 cache
- Each node has own main memory
- Nodes connected by some networking facility
- Each processor sees single addressable memory space
- Memory request order:
  - L1 cache (local to processor)
  - L2 cache (local to processor)
  - Main memory (local to node)
  - Remote memory
    - Delivered to requesting (local to processor) cache
- Automatic and transparent



# Memory Access Sequence

- Each node maintains directory of location of portions of memory and cache status
- e.g. node 2 processor 3 (P2-3) requests location 798 which is in memory of node 1
  - P2-3 issues read request on snoopy bus of node 2
  - Directory on node 2 recognises location is on node 1
  - Node 2 directory requests node 1's directory
  - Node 1 directory requests contents of 798
  - Node 1 memory puts data on (node 1 local) bus
  - Node 1 directory gets data from (node 1 local) bus
  - Data transferred to node 2's directory
  - Node 2 directory puts data on (node 2 local) bus
  - Data picked up, put in P2-3's cache and delivered to processor



# Cache Coherence

- Node 1 directory keeps note that node 2 has copy of data
- If data modified in cache, this is broadcast to other nodes
- Local directories monitor and purge local cache if necessary
- Local directory monitors changes to local data in remote caches and marks memory invalid until writeback
- Local directory forces writeback if memory location requested by another processor



# NUMA Pros & Cons

- Effective performance at higher levels of parallelism than SMP
- No major software changes
- Performance can breakdown if too much access to remote memory
  - Can be avoided by:
    - L1 & L2 cache design reducing all memory access
      - Need good temporal locality of software
    - Good spatial locality of software
    - Virtual memory management moving pages to nodes that are using them most
- Not transparent
  - Page allocation, process allocation and load balancing changes needed
- Availability?



# Vector Computation

- Maths problems involving physical processes present different difficulties for computation
  - Aerodynamics, seismology, meteorology
  - Continuous field simulation
- High precision
- Repeated floating point calculations on large arrays of numbers
- Supercomputers handle these types of problem
  - Hundreds of millions of flops
  - \$10-15 million
  - Optimised for calculation rather than multitasking and I/O
  - Limited market
    - Research, government agencies, meteorology
- Array processor
  - Alternative to supercomputer
  - Configured as peripherals to mainframe & mini
  - Just run vector portion of problems





# Vector Addition Example

$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.903 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

$$A + B = C$$



# Approaches

- General purpose computers rely on iteration to do vector calculations
- In example this needs six calculations
- Vector processing
  - Assume possible to operate on one-dimensional vector of data
  - All elements in a particular row can be calculated in parallel
- Parallel processing
  - Independent processors functioning in parallel
  - Use FORK N to start individual process at location N
  - JOIN N causes N independent processes to join and merge following JOIN
    - O/S Co-ordinates JOINS
    - Execution is blocked until all N processes have reached JOIN

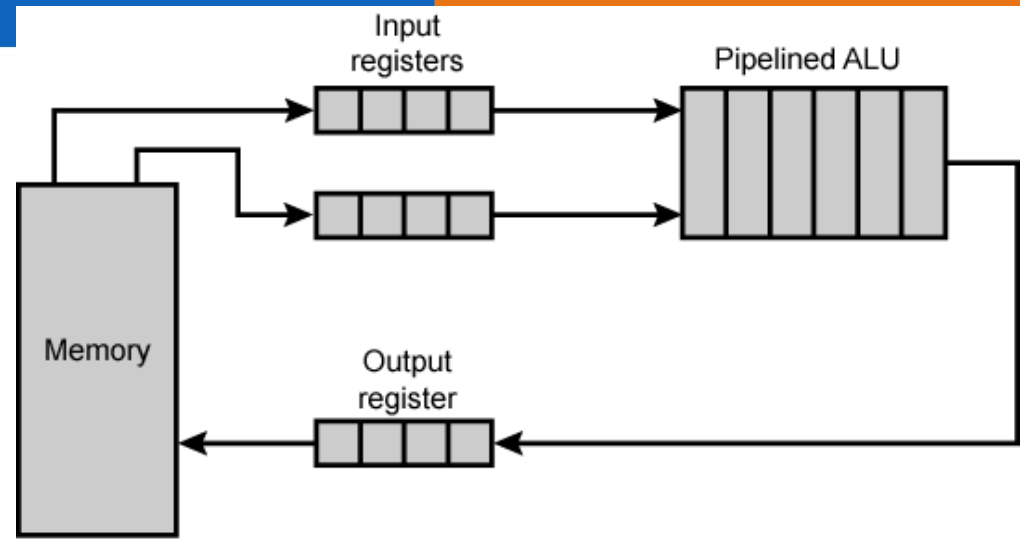


# Processor Designs

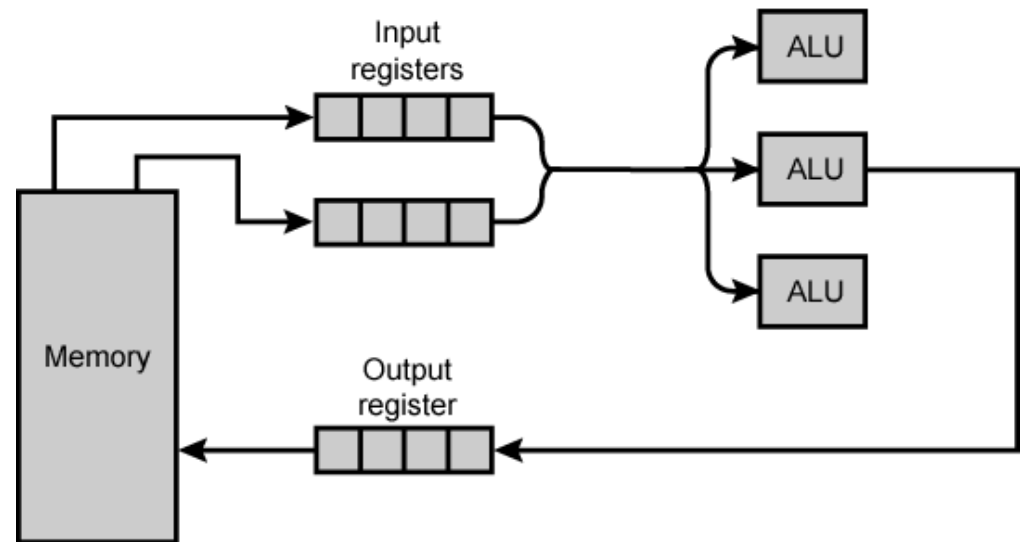
- Pipelined ALU
  - Within operations
  - Across operations
- Parallel ALUs
- Parallel processors



# Approaches to Vector Computation



(a) Pipelined ALU



(b) Parallel ALUs

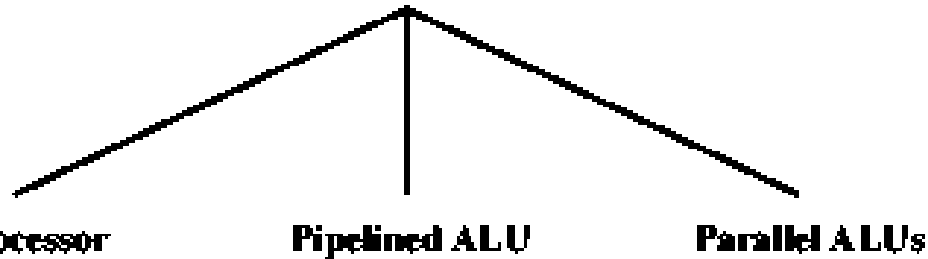
# Chaining

- Cray Supercomputers
- Vector operation may start as soon as first element of operand vector available and functional unit is free
- Result from one functional unit is fed immediately into another
- If vector registers used, intermediate results do not have to be stored in memory

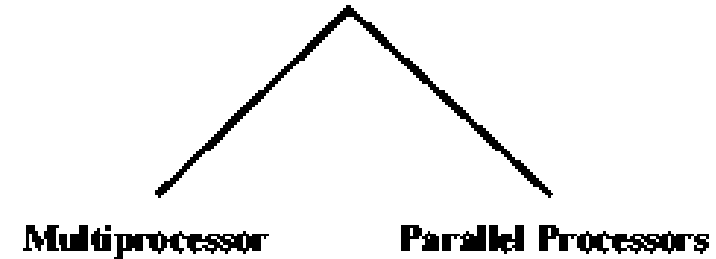


# Computer Organizations

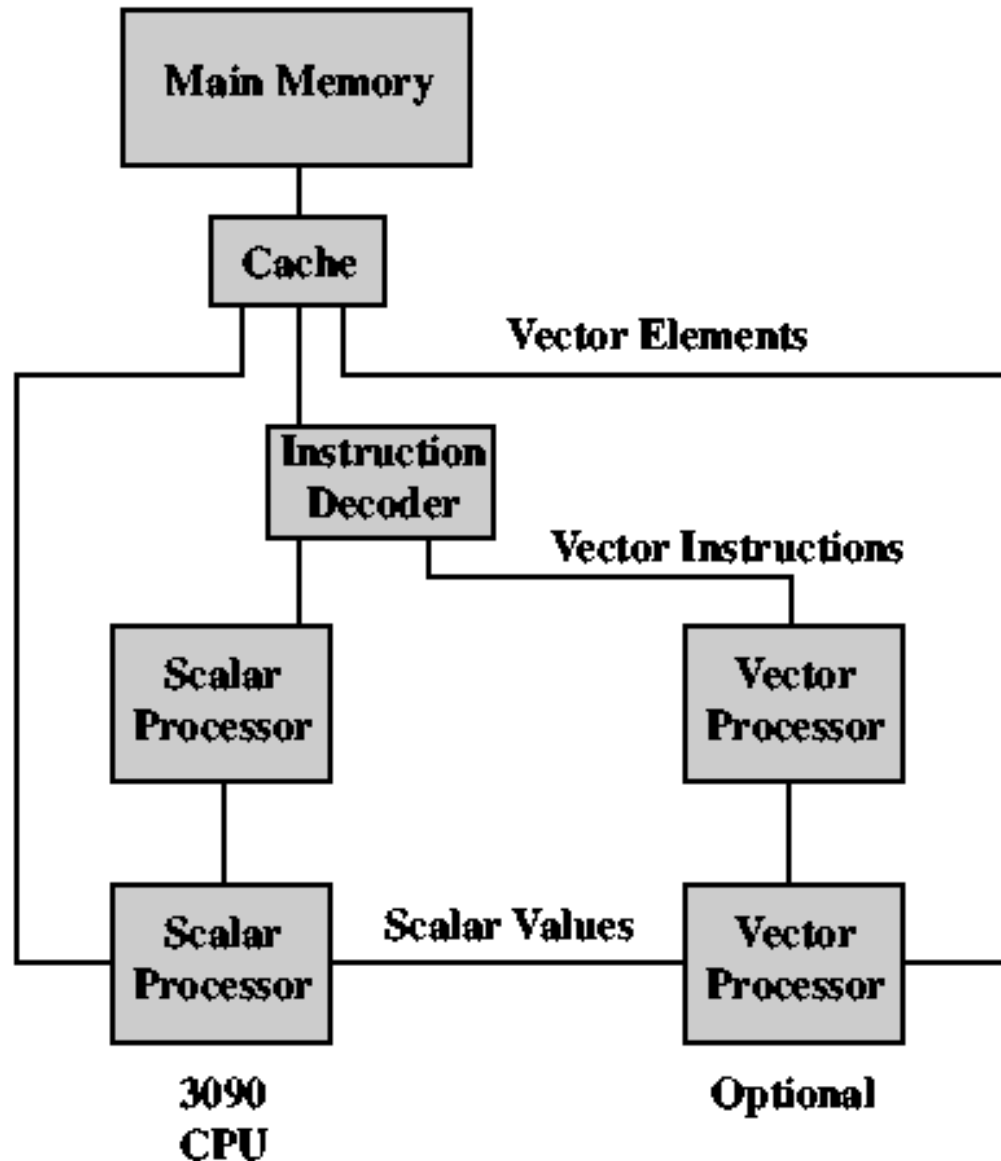
## Single Control Unit



## Multiple Control Units



# IBM 3090 with Vector Facility



# TERIMAKASIH



**UNDIP** | UNIVERSITAS  
DIPONEGORO  
becomes an excellent research university