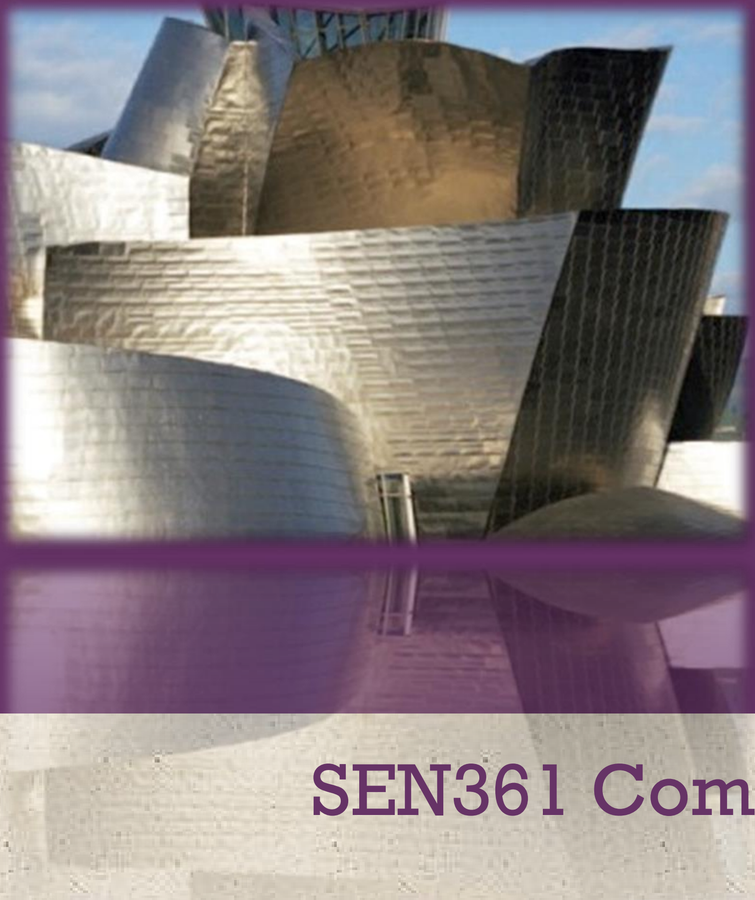


SEN361 Computer Organization

Prof. Dr. Hasan Hüseyin BALIK
(11th Week)

+

Outline

4. Parallel Organization

4.1 Parallel Processing

4.2 Multicore Computers

+

4.1 Parallel Processing

+

4.1 Outline

- Multiple Processor Organizations
- Symmetric Multiprocessors
- Cache Coherence and the MESI Protocol
- Multithreading and Chip Multiprocessors
- Clusters
- Nonuniform Memory Access
- Vector Computation

+

Multiple Processor Organization

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

- Single instruction, multiple data **(SIMD)** stream
  - A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis
  - Vector and array processors fall into this category

- 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
  - Not commercially implemented

- Multiple instruction, multiple data **(MIMD)** stream
  - A set of processors simultaneously execute different instruction sequences on different data sets
  - SMPs, clusters and NUMA systems fit this category

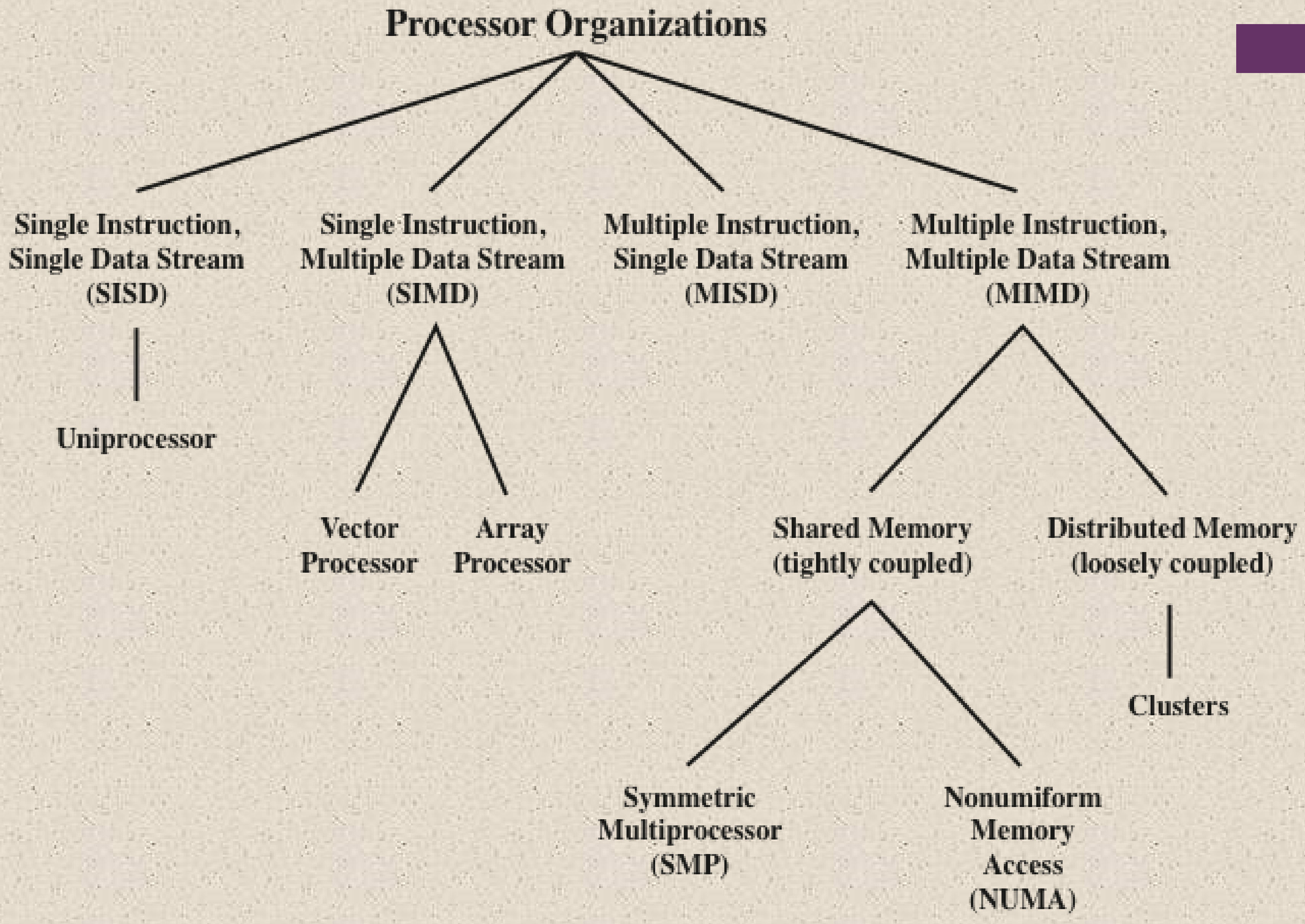
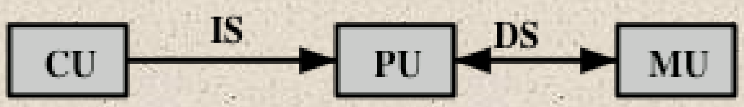
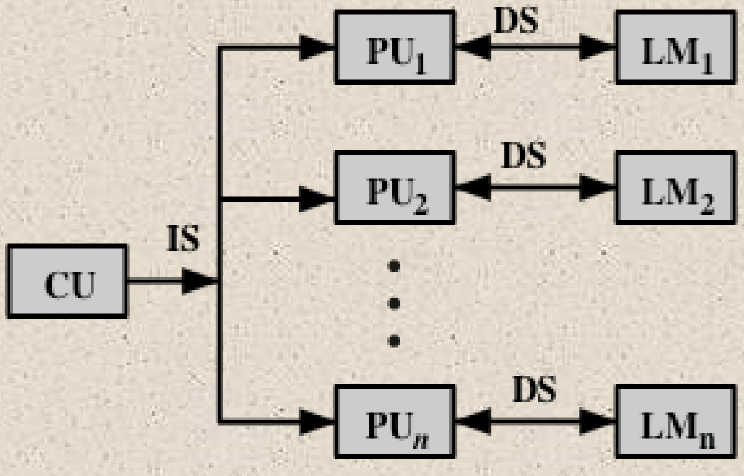

Processor Organizations
Single Instruction, Single Data Stream (SISD)
Single Instruction, Multiple Data Stream (SIMD)
Multiple Instruction, Single Data Stream (MISD)
Multiple Instruction, Multiple Data Stream (MIMD)
Uniprocessor
Vector Processor
Array Processor
Shared Memory (tightly coupled)
Distributed Memory (loosely coupled)
Clusters
Symmetric Multiprocessor (SMP)
Nonumiform Memory Access (NUMA)

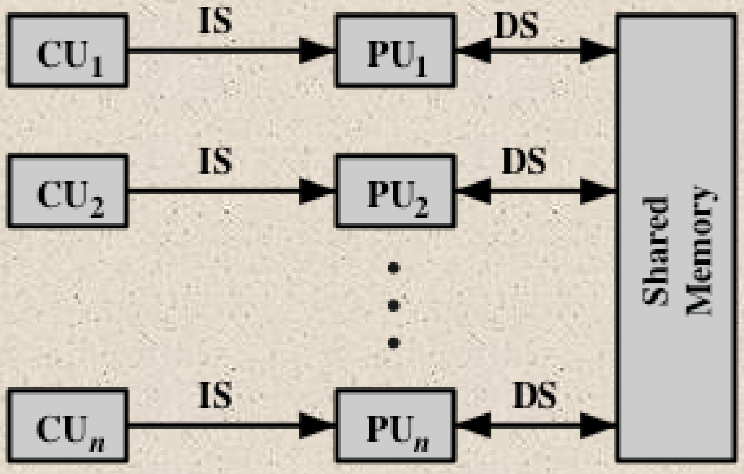
Figure 17.1 A Taxonomy of Parallel Processor Architectures



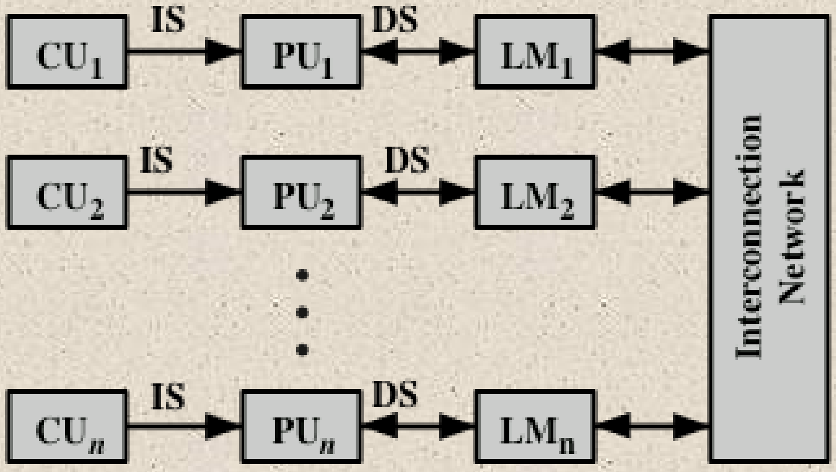
(a) SISD



(b) SIMD (with distributed memory)



(c) MIMD (with shared memory)



(d) MIMD (with distributed memory)

CU = control unit	SISD =	single instruction,
IS = instruction stream		single data stream
PU = processing unit	SIMD =	single instruction,
DS = data stream		multiple data stream
MU = memory unit	MIMD =	multiple instruction,
LM = local memory		multiple data stream

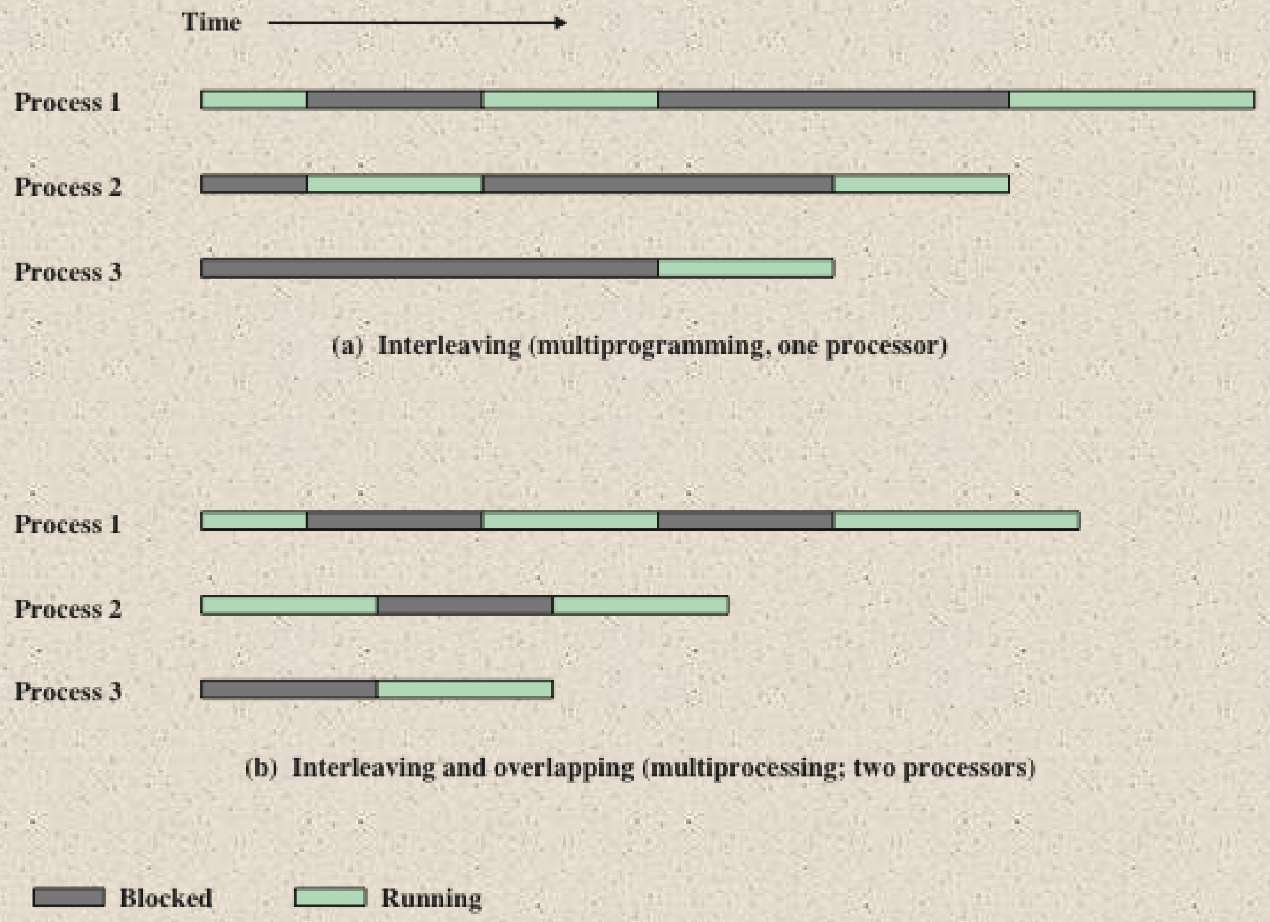
Figure 17.2 Alternative Computer Organizations

Symmetric Multiprocessor (SMP)

A stand alone computer with the following characteristics:

- Two or more similar processors of comparable capacity
- Processors share same memory and I/O facilities
  - 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 devices
  - 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
  - Provides interaction between processors and their programs at job, task, file and data element levels

Multiprogramming and Multiprocessing



Time
Process 1
Process 2
Process 3
(a) Interleaving (multiprogramming, one processor)
Process 1
Process 2
Process 3
(b) Interleaving and overlapping (multiprocessing; two processors)
Blocked
Running

Figure 17.3 Multiprogramming and Multiprocessing

Symmetric Multiprocessor Organization

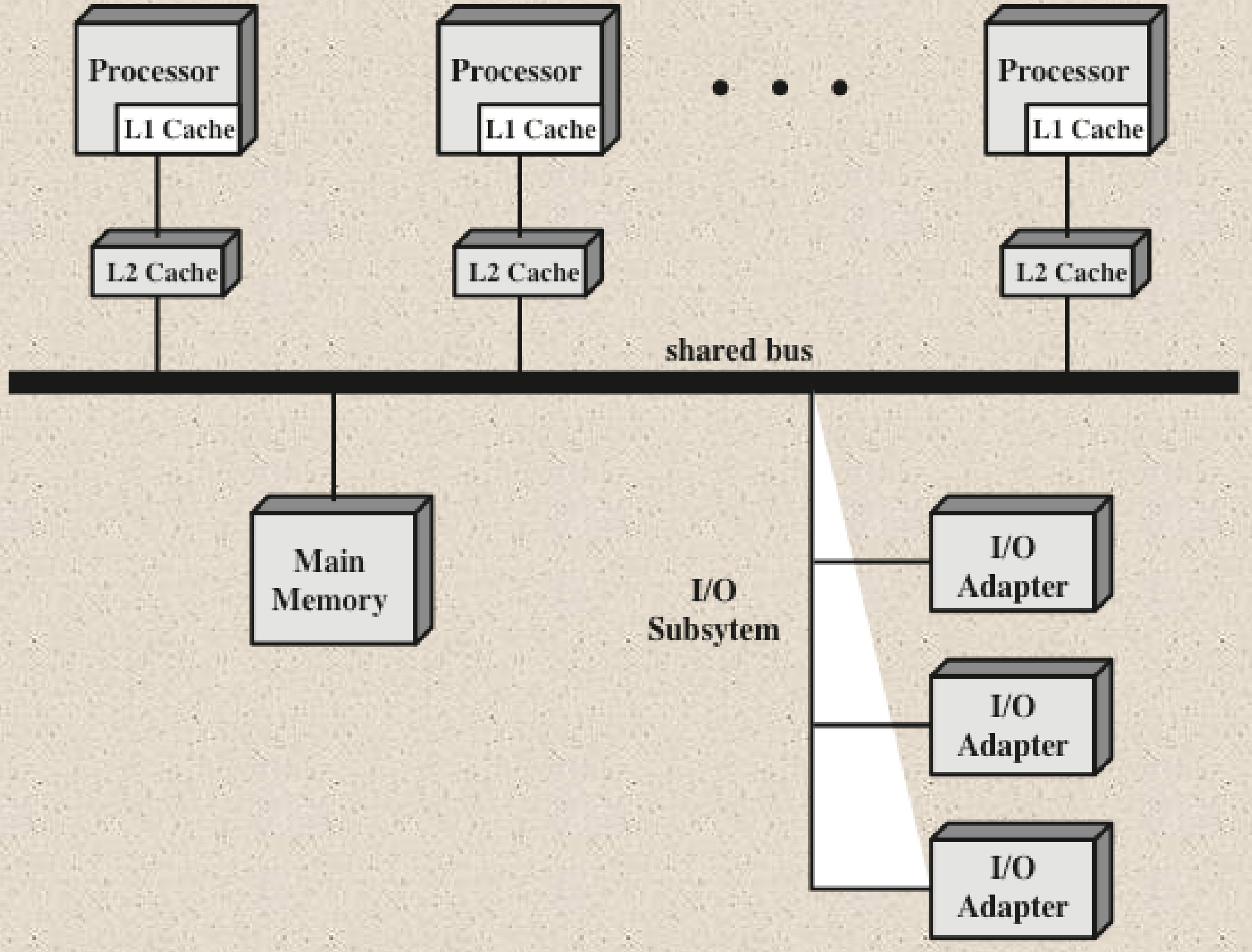
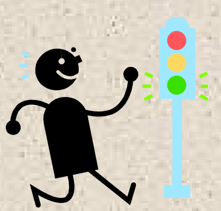


Figure 17.5 Symmetric Multiprocessor Organization

+

The bus organization has several attractive features:



- Simplicity
  - Simplest approach to multiprocessor organization
- Flexibility
  - Generally easy to expand the system by attaching more processors to the bus
- Reliability
  - The bus is essentially a passive medium and the failure of any attached device should not cause failure of the whole system

+

Disadvantages of the bus organization:

- Main drawback is performance
  - All memory references pass through the common bus
  - Performance is limited by bus cycle time
- Each processor should have cache memory
  - Reduces the number of bus accesses
- Leads to problems with *cache coherence*
  - If a word is altered in one cache it could conceivably invalidate a word in another cache
    - To prevent this the other processors must be alerted that an update has taken place
  - Typically addressed in hardware rather than the operating system

+

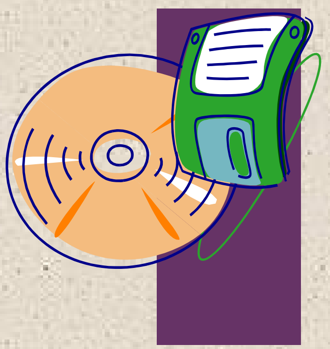
Multiprocessor Operating System Design Considerations

- **Simultaneous concurrent processes**
  - OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously
  - OS tables and management structures must be managed properly to avoid deadlock or invalid operations
- **Scheduling**
  - Any processor may perform scheduling so conflicts must be avoided
  - Scheduler must assign ready processes to available processors
- **Synchronization**
  - With multiple active processes having potential access to shared address spaces or I/O resources, care must be taken to provide effective synchronization
  - Synchronization is a facility that enforces mutual exclusion and event ordering
- **Memory management**
  - In addition to dealing with all of the issues found on uniprocessor machines, the OS needs to exploit the available hardware parallelism to achieve the best performance
  - Paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement
- **Reliability and fault tolerance**
  - OS should provide graceful degradation in the face of processor failure
  - Scheduler and other portions of the operating system must recognize the loss of a processor and restructure accordingly

+

Cache Coherence

Software Solutions



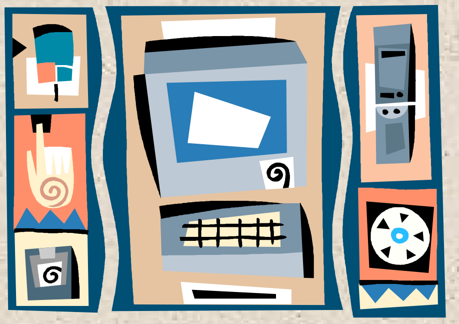
- Attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem

- Attractive because the overhead of detecting potential problems is transferred from run time to compile time, and the design complexity is transferred from hardware to software
  - However, compile-time software approaches generally must make conservative decisions, leading to inefficient cache utilization

+

Cache Coherence

Hardware-Based Solutions



- Generally referred to as cache coherence protocols
- These solutions provide dynamic recognition at run time of potential inconsistency conditions
- Because the problem is only dealt with when it actually arises there is more effective use of caches, leading to improved performance over a software approach
- Approaches are transparent to the programmer and the compiler, reducing the software development burden
- Can be divided into two categories:
  - Directory protocols
  - Snoopy protocols

Directory Protocols

Collect and maintain information about copies of data in cache

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

- Distribute the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor
  - A cache must recognize when a line that it holds is shared with other caches
  - When updates are performed on a shared cache line, it must be announced to other caches by a broadcast mechanism
  - Each cache controller is able to "snoop" on the network to observe these broadcast notifications and react accordingly
- Suited to bus-based multiprocessor because the shared bus provides a simple means for broadcasting and snooping
  - Care must be taken that the increased bus traffic required for broadcasting and snooping does not cancel out the gains from the use of local caches
- Two basic approaches have been explored:
  - Write invalidate
  - Write update (or write broadcast)

+
Write Invalidate

- Multiple readers, but only one writer at a time
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line is required by another processor
- Most widely used in commercial multiprocessor systems such as the Pentium 4 and PowerPC
- State of every line is marked as modified, exclusive, shared or invalid
  - For this reason the write-invalidate protocol is called *MESI*

+
Write Update

- Can be multiple readers and writers
- When a processor wishes to update a shared line the word to be updated is distributed to all others and caches containing that line can update it
- Some systems use an adaptive mixture of both write-invalidate and write-update mechanisms

+

MESI Protocol

To provide cache consistency on an SMP the data cache supports a protocol known as MESI:

- Modified
  - The line in the cache has been modified and is available only in this cache
- Exclusive
  - The line in the cache is the same as that in main memory and is not present in any other cache
- Shared
  - The line in the cache is the same as that in main memory and may be present in another cache
- Invalid
  - The line in the cache does not contain valid data

+

Multithreading and Chip Multiprocessors

- Processor performance can be measured by the rate at which it executes instructions
- MIPS rate = f * IPC
  - f = processor clock frequency, in MHz
  - IPC = average instructions per cycle
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- Multithreading
  - Allows for a high degree of instruction-level parallelism without increasing circuit complexity or power consumption
  - Instruction stream is divided into several smaller streams, known as threads, that can be executed in parallel

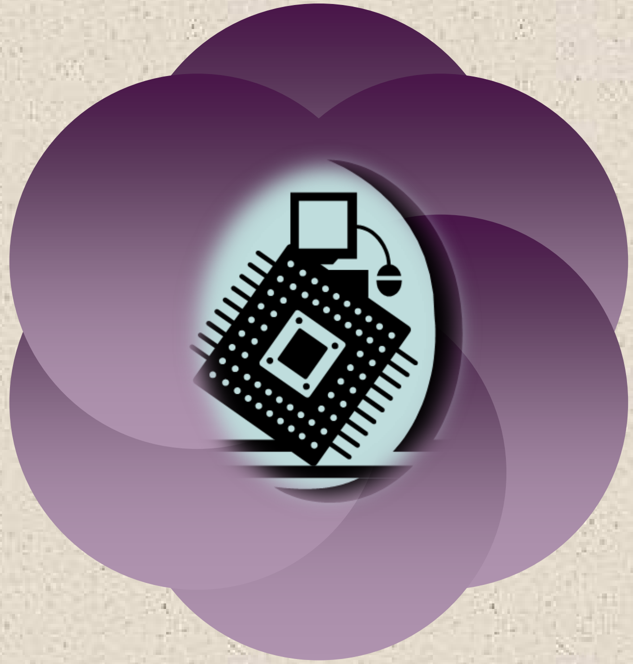
Definitions of Threads and Processes

Thread in multithreaded processors may or may not be the same as the concept of software threads in a multiprogrammed operating system

Thread is concerned with scheduling and execution, whereas a process is concerned with both scheduling/execution and resource and resource ownership

Thread switch

- The act of switching processor control between threads within the same process
- Typically less costly than process switch



Thread:

- Dispatchable unit of work within a process
- Includes processor context (which includes the program counter and stack pointer) and data area for stack
- Executes sequentially and is interruptible so that the processor can turn to another thread

Process:

- An instance of program running on computer
- Two key characteristics:
  - Resource ownership
  - Scheduling/execution

Process switch

- Operation that switches the processor from one process to another by saving all the process control data, registers, and other information for the first and replacing them with the process information for the second

Implicit and Explicit Multithreading

- All commercial processors and most experimental ones use explicit multithreading
  - Concurrently execute instructions from different explicit threads
  - Interleave instructions from different threads on shared pipelines or parallel execution on parallel pipelines
- Implicit multithreading is concurrent execution of multiple threads extracted from single sequential program
  - Implicit threads defined statically by compiler or dynamically by hardware

+

Approaches to Explicit Multithreading

- Interleaved
  - Fine-grained
  - Processor deals with two or more thread contexts at a time
  - Switching thread at each clock cycle
  - If thread is blocked it is skipped
- Simultaneous (SMT)
  - Instructions are simultaneously issued from multiple threads to execution units of superscalar processor
- Blocked
  - Coarse-grained
  - Thread executed until event causes delay
  - Effective on in-order processor
  - Avoids pipeline stall
- Chip multiprocessing
  - Processor is replicated on a single chip
  - Each processor handles separate threads
  - Advantage is that the available logic area on a chip is used effectively

+

Example Systems

Pentium 4

- More recent models of the Pentium 4 use a multithreading technique that Intel refers to as *hyperthreading*
- Approach is to use SMT with support for two threads
- Thus the single multithreaded processor is logically two processors

IBM Power5

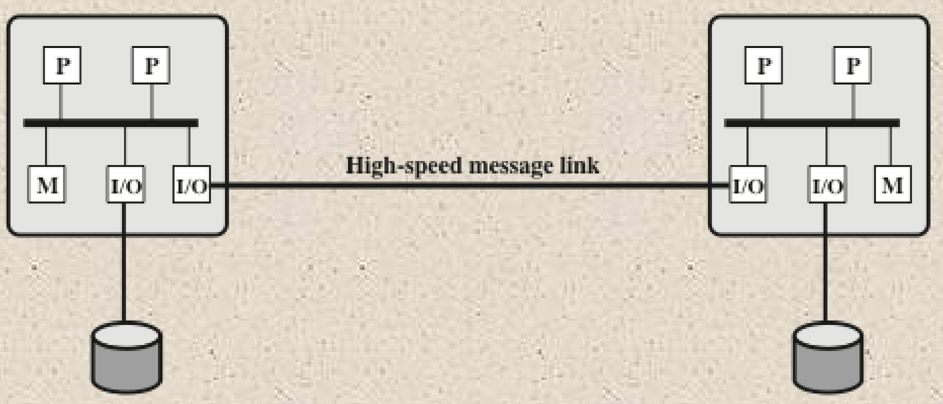
- Chip used in high-end PowerPC products
- Combines chip multiprocessing with SMT
  - Has two separate processors, each of which is a multithreaded processor capable of supporting two threads concurrently using SMT
  - Designers found that having two two-way SMT processors on a single chip provided superior performance to a single four-way SMT processor

Clusters

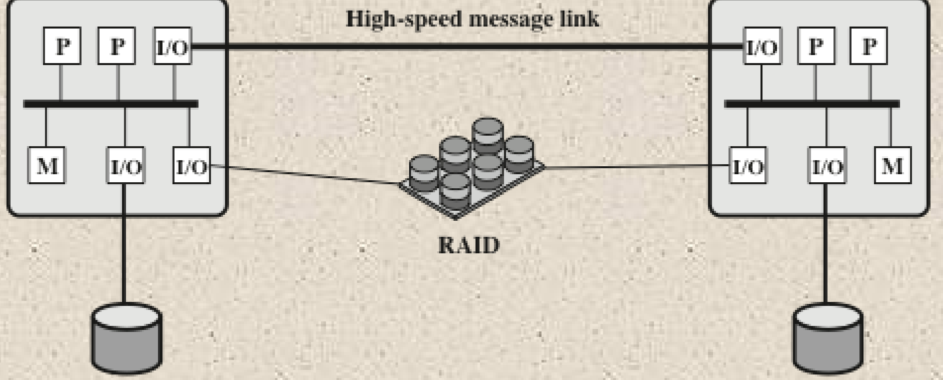
- Alternative to SMP as an approach to providing high performance and high availability
- Particularly attractive for server applications
- Defined as:
  - A group of interconnected whole computers working together as a unified computing resource that can create the illusion of being one machine
  - (The term *whole computer* means a system that can run on its own, apart from the cluster)
- Each computer in a cluster is called a node
- Benefits:
  - Absolute scalability
  - Incremental scalability
  - High availability
  - Superior price/performance

+

Cluster Configurations



(a) Standby server with no shared disk



(b) Shared disk

Figure 17.9 Cluster Configurations

+

Operating System Design Issues

- How failures are managed depends on the clustering method used
- Two approaches:
  - Highly available clusters
  - Fault tolerant clusters
- Failover
  - The function of switching applications and data resources over from a failed system to an alternative system in the cluster
- Failback
  - Restoration of applications and data resources to the original system once it has been fixed
- Load balancing
  - Incremental scalability
  - Automatically include new computers in scheduling
  - Middleware needs to recognize that processes may switch between machines

Parallelizing Computation

Effective use of a cluster requires executing software from a single application in parallel

Three approaches are:

Parallelizing complier

- Determines at compile time which parts of an application can be executed in parallel
- These are then split off to be assigned to different computers in the cluster

Parallelized application

- Application written from the outset to run on a cluster and uses message passing to move data between cluster nodes

Parametric computing

- Can be used if the essence of the application is an algorithm or program that must be executed a large number of times, each time with a different set of starting conditions or parameters

+

Clusters Compared to SMP

- Both provide a configuration with multiple processors to support high demand applications
- Both solutions are available commercially

SMP	Clustering
Easier to manage and configure	Far superior in terms of incremental and absolute scalability
Much closer to the original single processor model for which nearly all applications are written	Superior in terms of availability
Less physical space and lower power consumption	All components of the system can readily be made highly redundant
Well established and stable	

+

Nonuniform Memory Access (NUMA)

- Alternative to SMP and clustering
- Uniform memory access (UMA)
  - All processors have access to all parts of main memory using loads and stores
  - Access time to all regions of memory is the same
  - Access time to memory for different processors is the same
- Nonuniform memory access (NUMA)
  - All processors have access to all parts of main memory using loads and stores
  - Access time of processor differs depending on which region of main memory is being accessed
  - Different processors access different regions of memory at different speeds
- Cache-coherent NUMA (CC-NUMA)
  - A NUMA system in which cache coherence is maintained among the caches of the various processors

Motivation

SMP has practical limit to number of processors that can be used

- Bus traffic limits to between 16 and 64 processors

In clusters each node has its own private main memory

- Applications do not see a large global memory
- Coherency is maintained by software rather than hardware

NUMA retains SMP flavor while giving large scale multiprocessing

Objective with NUMA is to maintain a transparent system wide memory while permitting multiple multiprocessor nodes, each with its own bus or internal interconnect system

+

NUMA Pros and Cons

- Main advantage of a CC-NUMA system is that it can deliver effective performance at higher levels of parallelism than SMP without requiring major software changes

- Bus traffic on any individual node is limited to a demand that the bus can handle

- If many of the memory accesses are to remote nodes, performance begins to break down

- Does not transparently look like an SMP

- Software changes will be required to move an operating system and applications from an SMP to a CC-NUMA system

- Concern with availability

+
Vector Computation

- There is a need for computers to solve mathematical problems of physical processes in disciplines such as aerodynamics, seismology, meteorology, and atomic, nuclear, and plasma physics

- Need for high precision and a program that repetitively performs floating point arithmetic calculations on large arrays of numbers
  - Most of these problems fall into the category known as *continuous-field simulation*

- Supercomputers were developed to handle these types of problems
  - However they have limited use and a limited market because of their price tag
  - There is a constant demand to increase performance

- Array processor
  - Designed to address the need for vector computation
  - Configured as peripheral devices by both mainframe and minicomputer users to run the vectorized portions of programs

+

4.2 Multicore Computers

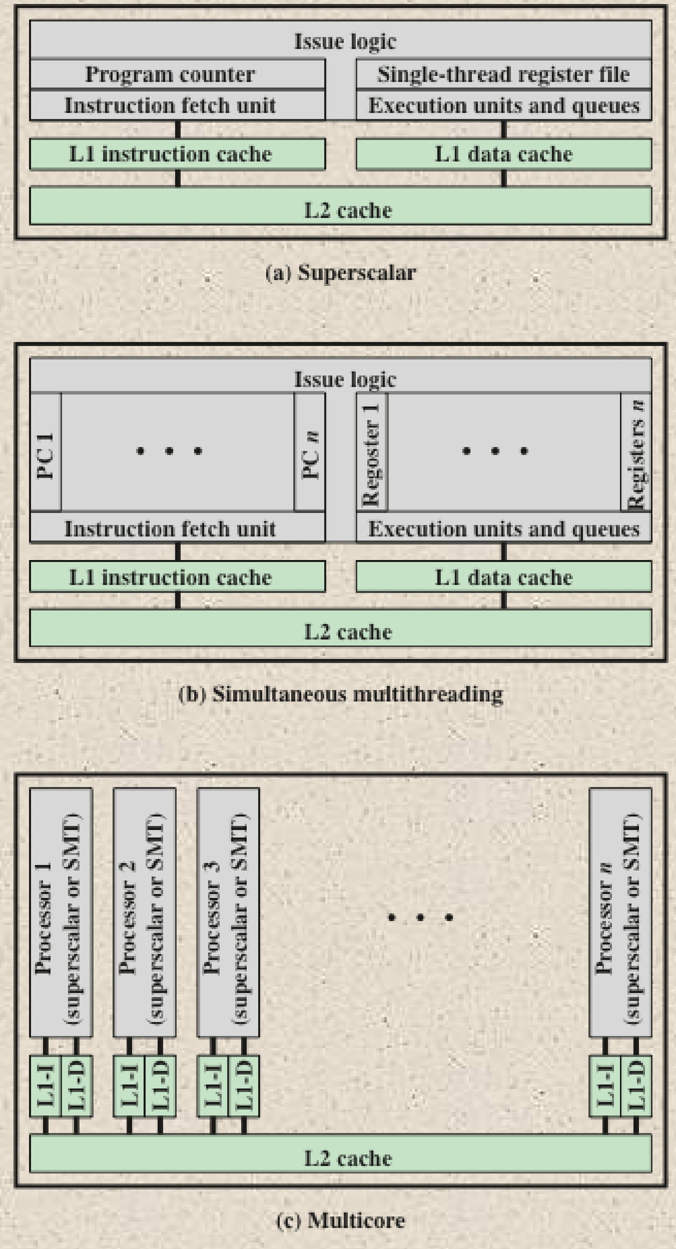
+

4.1 Outline

- Hardware Performance Issues
- Software Performance Issues
- Multicore Organization
- Intel x86 Multicore Organization
- ARM11 MPCore
- IBM zEnterprise 196 Mainframe

+

Alternative Chip Organization



Issue logic
Program counter
Single-thread register file
Instruction fetch unit
Execution units and queues
L1 instruction cache
L1 data cache
L2 cache
(a) Superscalar
Issue logic
PC 1
PC n
Regoster 1
Registers n
Instruction fetch unit
Execution units and queues
L1 instruction cache
L1 data cache
L2 cache
(b) Simultaneous multithreading
Processor 1 (superscalar or SMT)
Processor 2 (superscalar or SMT)
Processor 3 (superscalar or SMT)
Processor n (superscalar or SMT)
L1-I
L1-D
L2 cache
(c) Multicore

Figure 18.1 Alternative Chip Organizations

+

Processor Trends

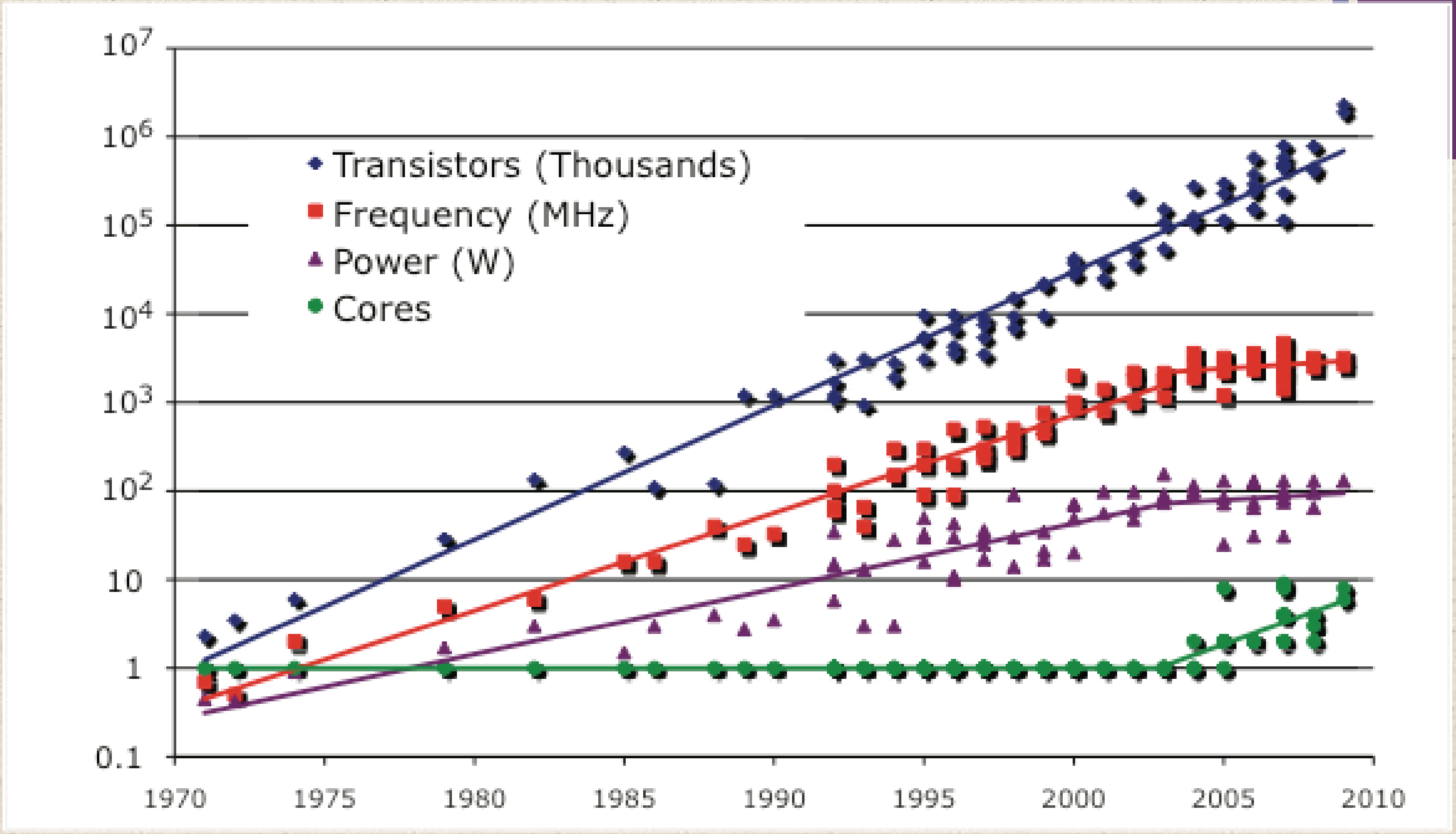


Figure 18.3 Processor Trends

+

Power Consumption

- By 2015 we can expect to see microprocessor chips with about 100 billion transistors on a 300 mm^2 die
- Assuming that about 50-60% of the chip area is devoted to memory, the chip will support cache memory of about 100 MB and leave over 1 billion transistors available for logic
- How to use all those logic transistors is a key design issue
- Pollack's Rule
  - States that performance increase is roughly proportional to square root of increase in complexity

+

Effective Applications for Multicore Processors

- **Multi-threaded native applications**
  - Characterized by having a small number of highly threaded processes
  - Lotus Domino, Siebel CRM (Customer Relationship Manager)
- **Multi-process applications**
  - Characterized by the presence of many single-threaded processes
  - Oracle, SAP, PeopleSoft
- **Java applications**
  - Java Virtual Machine is a multi-threaded process that provides scheduling and memory management for Java applications
  - Sun's Java Application Server, BEA's Weblogic, IBM Websphere, Tomcat
- **Multi-instance applications**
  - One application running multiple times
  - If multiple application instances require some degree of isolation, virtualization technology can be used to provide each of them with its own separate and secure environment

Multicore Organization Alternatives

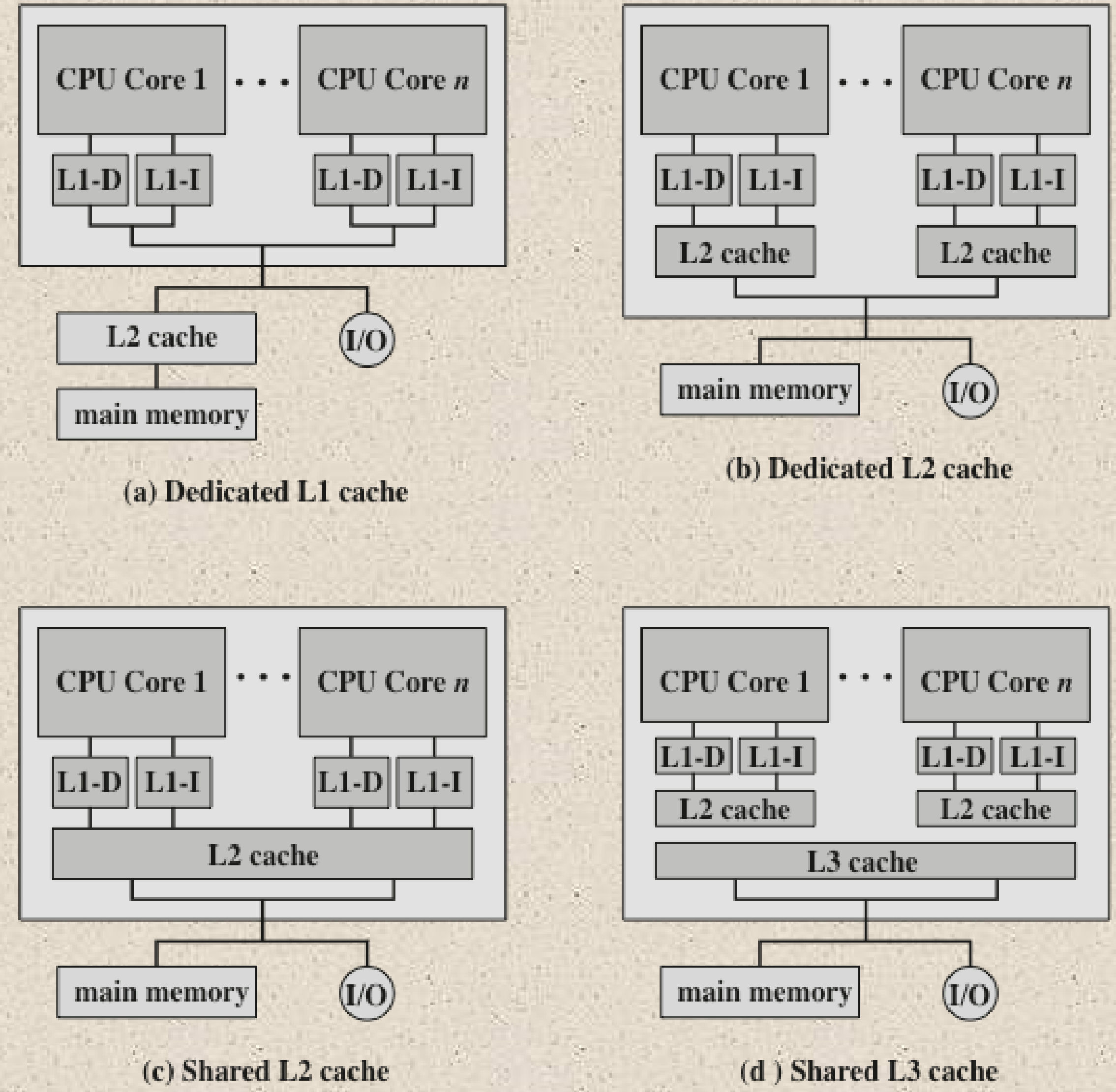


Figure 18.8 Multicore Organization Alternatives

Intel Core Duo Block Diagram

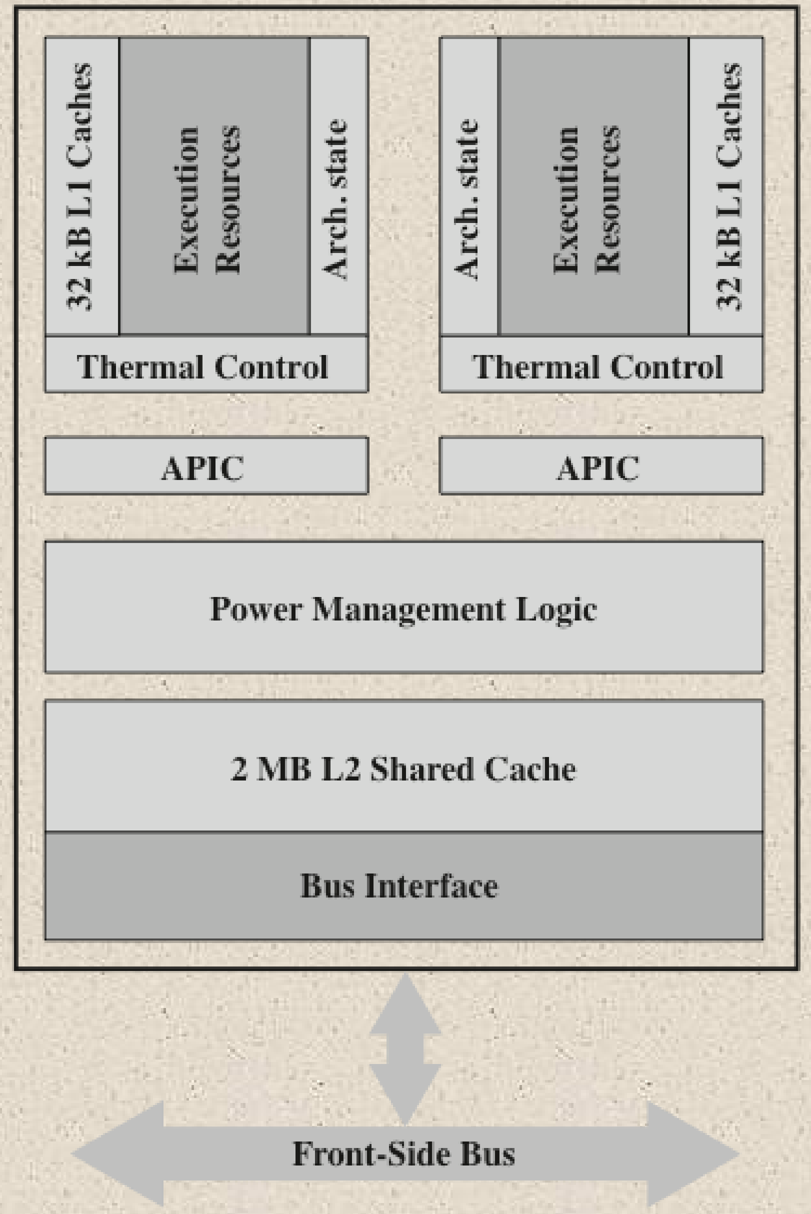


Figure 18.9 Intel Core Duo Block Diagram

+

Intel x86 Multicore Organization Core Duo

- **Advanced Programmable Interrupt Controller (APIC)**
  - Provides inter-processor interrupts which allow any process to interrupt any other processor or set of processors
  - Accepts I/O interrupts and routes these to the appropriate core
  - Includes a timer which can be set by the OS to generate an interrupt to the local core
- **Power management logic**
  - Responsible for reducing power consumption when possible, thus increasing battery life for mobile platforms
  - Monitors thermal conditions and CPU activity and adjusts voltage levels and power consumption appropriately
  - Includes an advanced power-gating capability that allows for an ultra fine grained logic control that turns on individual processor logic subsystems only if and when they are needed

Continued . . .

+
Intel x86 Multicore Organization Core Duo

- **2MB shared L2 cache**
  - Cache logic allows for a dynamic allocation of cache space based on current core needs
  - MESI support for L1 caches
  - Extended to support multiple Core Duo in SMP
  - L2 cache controller allows the system to distinguish between a situation in which data are shared by the two local cores, and a situation in which the data are shared by one or more caches on the die as well as by an agent on the external bus
- **Bus interface**
  - Connects to the external bus, known as the Front Side Bus, which connects to main memory, I/O controllers, and other processor chips

Intel Core i7-990X Block Diagram

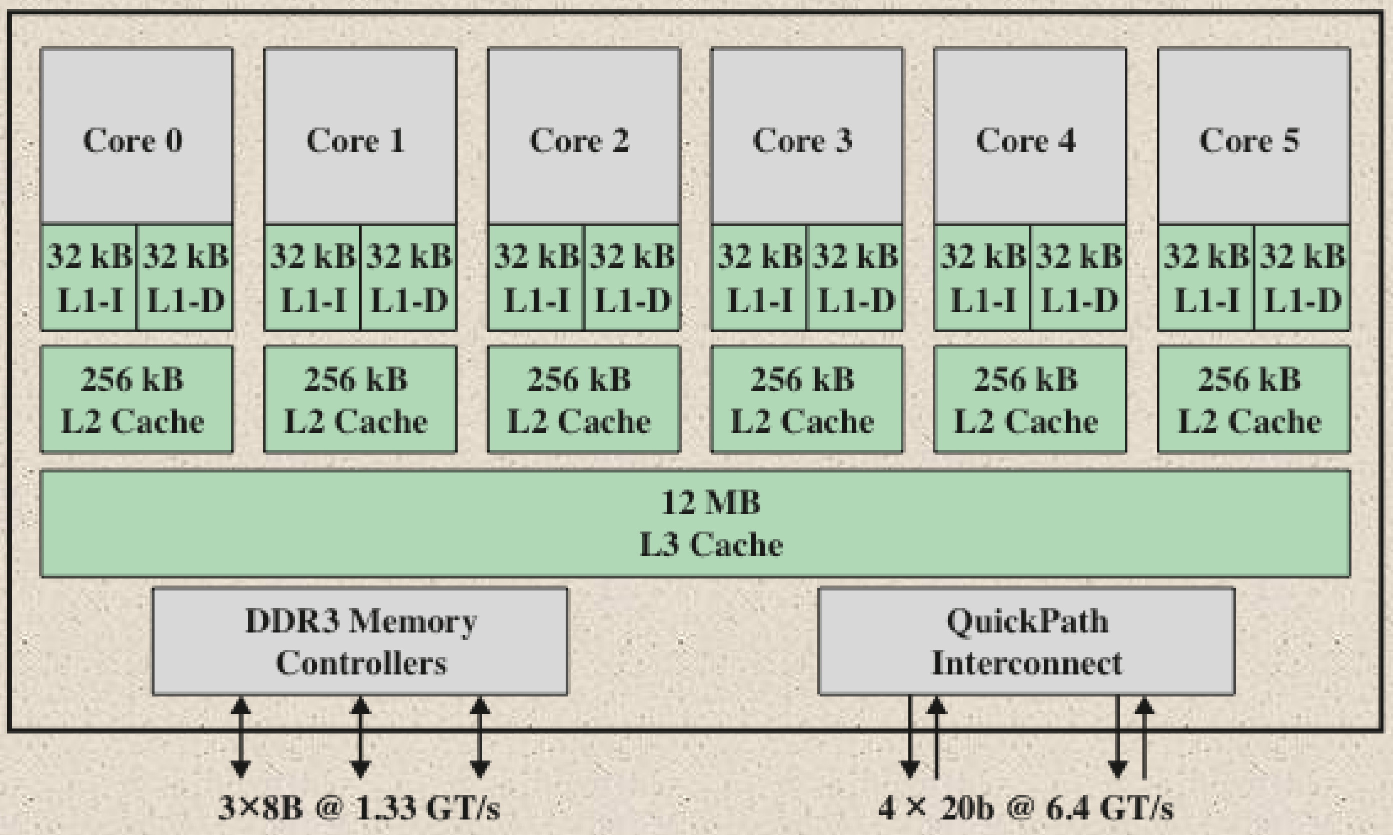


Figure 18.10 Intel Core i7-990X Block Diagram

+

Interrupt Handling

- Distributed Interrupt Controller (DIC) collates interrupts from a large number of sources
- It provides:
  - Masking of interrupts
  - Prioritization of the interrupts
  - Distribution of the interrupts to the target MP11 CPUs
  - Tracking status of interrupts
  - Generation of interrupts by software
- Is a single function unit that is placed in the system alongside MP11 CPUs
- Memory mapped
- Accessed by CPUs via private interface through SCU
- Provides a means of routing an interrupt request to a single CPU or multiple CPUs, as required
- Provide a means of interprocessor communication so that a thread on one CPU can cause activity by a thread on another CPU

+

DIC Routing

- The DIC can route an interrupt to one or more CPUs in the following three ways:
  - An interrupt can be directed to a specific processor only
  - An interrupt can be directed to a defined group of processors
  - An interrupt can be directed to all processors
- OS can generate interrupt to:
  - All but self
  - Self
  - Other specific CPU
- Typically combined with shared memory for inter-process communication
- 16 interrupt IDs available for inter-processor communication

+

Interrupt States

From the point of view of an MP11 CPU, an interrupt can be:

Inactive

- Is one that is nonasserted, or which in a multi-processing environment has been completely processed by that CPU but can still be either Pending or Active in some of the CPUs to which it is targeted, and so might not have been cleared at the interrupt source

Pending

- Is one that has been asserted, and for which processing has not started on that CPU

Active

- Is one that has been started on that CPU, but processing is not complete
- An Active interrupt can be pre-empted when a new interrupt of higher priority interrupts MP11 CPU interrupt processing

+

Interrupt Sources

- Inter-process Interrupts (IPI)
  - Private to CPU
  - ID0-ID15
  - Software triggered
  - Priority depends on target CPU not source
- Private timer and/or watchdog interrupt
  - ID29 and ID30
- Legacy FIQ line
  - Legacy FIQ pin, per CPU, bypasses interrupt distributor
  - Directly drives interrupts to CPU
- Hardware
  - Triggered by programmable events on associated interrupt lines
  - Up to 224 lines
  - Start at ID32

ARM11 MPCore Interrupt Distributor

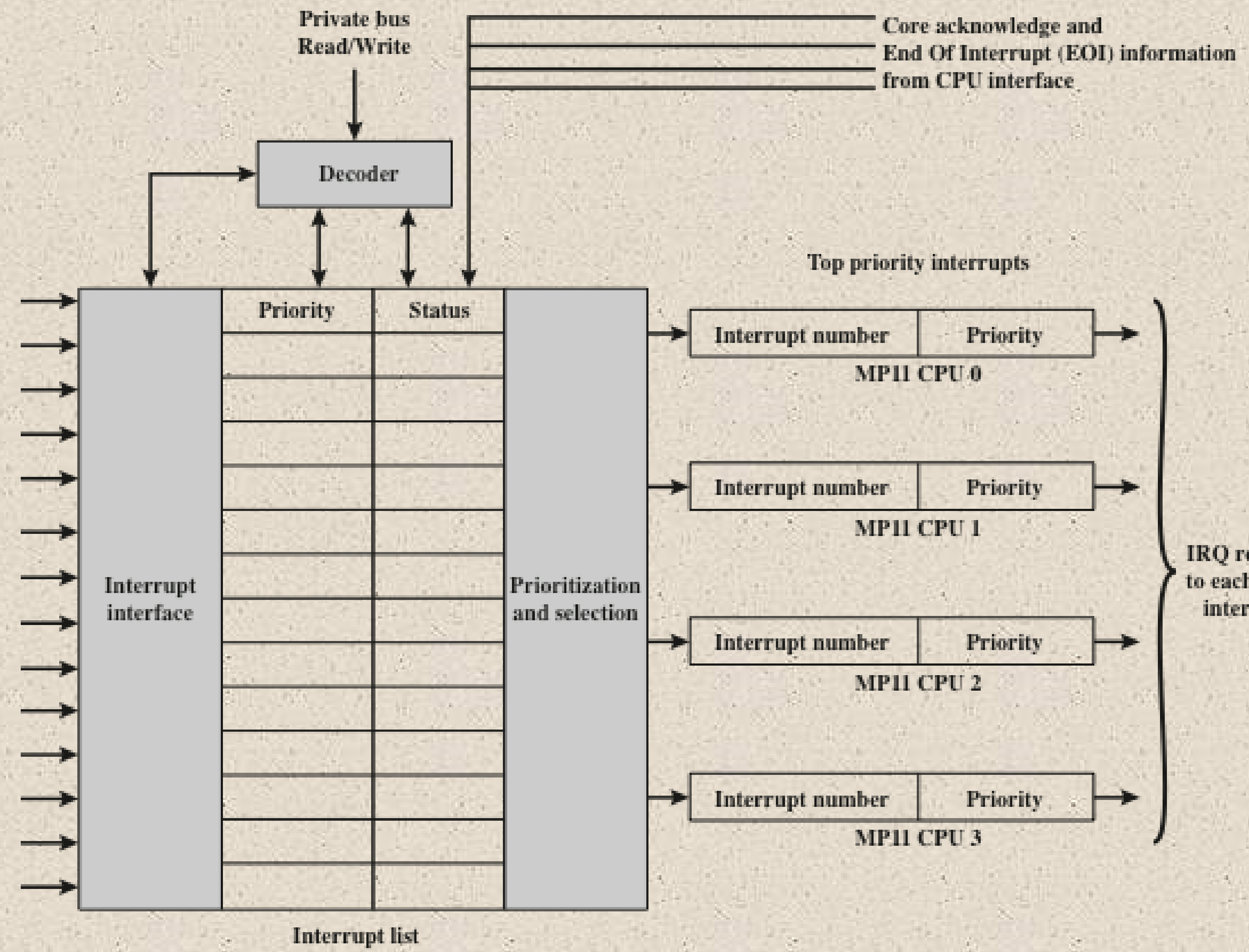


Figure 18.12 Interrupt Distributor Block Diagram

+
Cache Coherency

- Snoop Control Unit (SCU) resolves most shared data bottleneck issues
- L1 cache coherency scheme is based on the MESI protocol
- Direct Data Intervention (DDI)
  - Enables copying clean data between L1 caches without accessing external memory
  - Reduces read after write from L1 to L2
  - Can resolve local L1 miss from remote L1 rather than L2
- Duplicated tag RAMs
  - Cache tags implemented as separate block of RAM
  - Same length as number of lines in cache
  - Duplicates used by SCU to check data availability before sending coherency commands
  - Only send to CPUs that must update coherent data cache
- Migratory lines
  - Allows moving dirty data between CPUs without writing to L2 and reading back from external memory