

Computer Architecture

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr
nizamettinaydin@gmail.com

<http://www.yildiz.edu.tr/~naydin>

1

Computer Architecture

Parallel Processing

2

Outline

- 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
- Symmetric Multiprocessors
- Organization Classification
- Operating System Issues
- Multithreading and Chip Multiprocessors
- Implicit and Explicit Multithreading
- Scalar Processor Approaches
- Multiple Instruction Issue Processors
- Clusters
- Nonuniform Memory Access (NUMA)
- Vector Computation

3

Parallel Processing

- The computer is viewed as a sequential machine
- This view of the computer has never been entirely true.
 - At the micro-operation level, multiple control signals are generated at the same time
 - In instruction pipelining, fetch and execute stages overlap
 - In superscalar, there is instruction level parallelism

4

Parallel Processing

- Computer technology has evolved
- Cost of hardware has dropped
- Designers have sought more and more opportunities for parallelism ...
 - to enhance performance
 - to increase availability
- Some approaches:
 - Symmetric multiprocessors (SMPs)
 - Clusters
 - Multithreaded processors
 - Chip multiprocessors
 - Nonuniform memory access (NUMA) machines

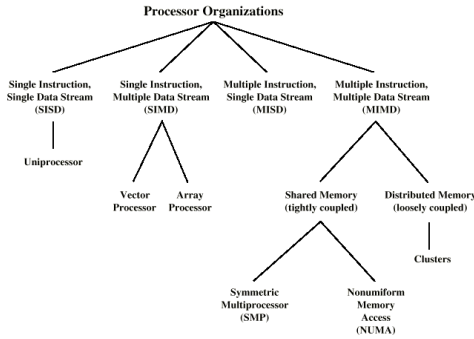
5

Multiple Processor Organization

- Categorization of systems with parallel processing capability:
 - Single instruction, single data stream – SISD
 - A single processor executes a single stream to operate on data stored in a single memory
 - Uniprocessors fall into this category
 - Single instruction, multiple data stream – SIMD
 - A single machine instruction controls the simultaneous execution of a number of processing elements
 - Vector and array processors fall into this category
 - Multiple instruction, single data stream – MISD
 - 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 stream – MIMD
 - A set of processors simultaneously execute different instruction sequences on different data sets.
 - SMPs, clusters, and NUMA systems fit into this category

6

Taxonomy of Parallel Processor Architectures



7

Multiple instruction, multiple data stream- (MIMD) - Overview

- Processors are general purpose
 - Each can process all instructions necessary
- Further classified by method of processor communication:
 - Shared memory (Tightly coupled)
 - Symmetric multiprocessor (SMP)
 - Nonuniform memory access (NUMA)
 - Distributed memory (Loosely coupled)
 - clusters

8

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
- The operating system takes care of scheduling of threads or processes on individual processors and of synchronization among processors

9

Tightly Coupled - NUMA

- Nonuniform memory access
- Access times to different regions of memory may differ for a NUMA processor

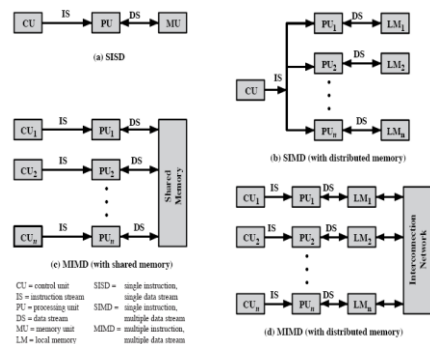
10

Loosely Coupled - Clusters

- Collection of independent uniprocessors or SMPs
- Interconnected to form a cluster
- Communication via fixed path or network connections

11

Multiple Processor Organization



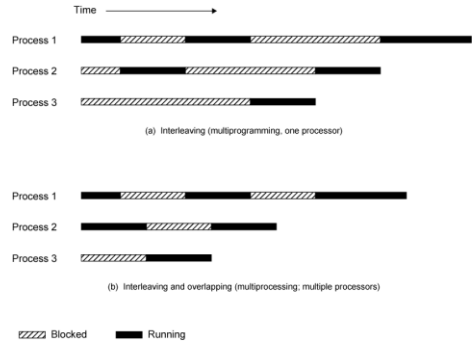
12

Symmetric Multiprocessors

- A stand alone computer with the following characteristics
 - Two or more similar processors of comparable capacity
 - 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

13

Multiprogramming and Multiprocessing



14

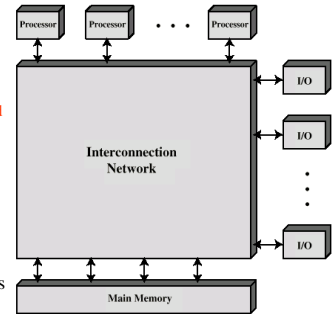
Potential SMP Advantages

- Performance
 - If some work can be done in parallel
- Availability
 - Since all processors can perform the same functions, failure of a single processor does not halt the system
- Incremental growth
 - User can enhance performance by adding additional processors
- Scaling
 - Vendors can offer range of products based on number of processors

15

Block Diagram of Tightly Coupled Multiprocessor

- Each processor...
 - is self contained
 - has ALU, registers, cache, etc.
 - has access to shared main memory
 - has access to I/O devices through some form of interconnection mechanism
- Simultaneous access to separate blocks in memory is possible



16

Organization Classification

- Time shared or common bus
- Multiport memory
- Central control unit

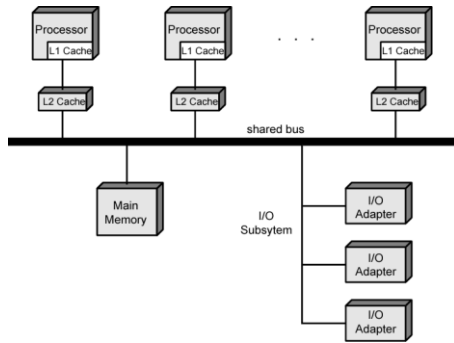
17

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

18

Symmetric Multiprocessor Organization



19

Time Share Bus - Advantages - Disadvantage

- Advantages:
 - Simplicity
 - Flexibility
 - Reliability
- Disadvantages:
 - Performance limited by bus cycle time
 - Each processor should have local cache
 - Reduce number of bus accesses
 - Leads to problems with cache coherence

20

Operating System Issues

- An SMP OS manages processor and other computer resources
- A multiprocessor OS must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors.
- So, the key design issues are the following:
 - Simultaneous concurrent processes
 - Scheduling
 - Synchronization
 - Memory management
 - Reliability and fault tolerance

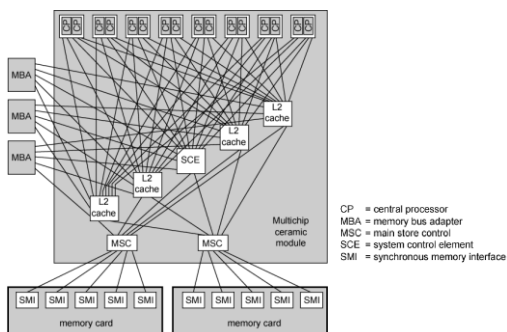
21

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
 - 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

22

IBM z990 Multiprocessor Structure



23

Cache Coherence and MESI Protocol

- MESI (Modified/Exclusive/Shared/Invalid)
- Problem - multiple copies of same data in different caches
- Can result in an inconsistent view of memory
- Write back policy can lead to inconsistency
- Write through can also give problems unless caches monitor memory traffic

24

Software Solutions

- Compiler and operating system deal with problem
- Overhead transferred to compile time
- Design complexity transferred from hardware to software
- However, software tends to make conservative decisions
 - Inefficient cache utilization
- Analyze code to determine safe periods for caching shared variables

25

Hardware Solution

- Cache coherence protocols
- Dynamic recognition of potential problems
- Run time
- More efficient use of cache
- Transparent to programmer
- Directory protocols
- Snoopy protocols

26

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

27

Snoopy Protocols

- Distribute cache coherence responsibility among cache controllers
- Cache recognizes that a line is shared
- Updates announced to other caches
- Suited to bus based multiprocessor
- Increases bus traffic
- Two basic approaches:
 - Write-invalidate
 - Write-update (write broadcast)

28

Write Invalidate

- Multiple readers, one writer
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line required by another processor
- Used in Pentium II and PowerPC systems
- State of every line is marked as Modified, Exclusive, Shared or Invalid
- Therefore also called MESI

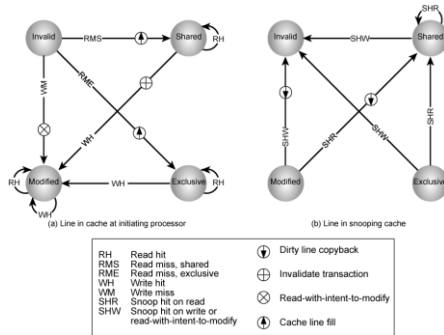
29

Write Update

- Multiple readers and writers
- Updated word is distributed to all other processors
- Some systems use an adaptive mixture of both solutions

30

MESI State Transition Diagram



31

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 is average instructions per cycle
 - MIPS, million instructions per second
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- May be reaching limit
 - Complexity
 - Power consumption

32

Multithreading and Chip Multiprocessors

- In multithreading:
 - Instruction stream divided into smaller streams (threads)
- Threads can be executed in parallel
- Wide variety of multithreading designs
- Thread in multithreaded processors may or may not be the same as software threads.

33

Definitions of Threads and Processes

- Some important terms are:
 - Process:
 - An instance of program running on a computer, which embodies two key characteristics:
 - Resource ownership
 - Virtual address space to hold process image
 - Scheduling/execution
 - Process switch:
 - An operation that switches the processor from one process to another
 - Thread: A dispatchable unit of work within process
 - 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

34

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

35

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
- Blocked
 - Coarse-grained
 - Thread executed until event causes delay
 - E.g. Cache miss
 - Effective on in-order processor
 - Avoids pipeline stall
- Simultaneous (SMT)
 - Instructions simultaneously issued from multiple threads to execution units of superscalar processor
- Chip multiprocessing
 - Processor is replicated on a single chip
 - Each processor handles separate threads

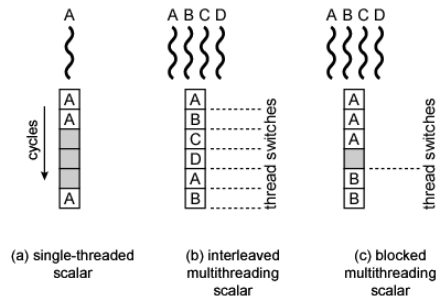
36

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
 - 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

37

Scalar Diagrams



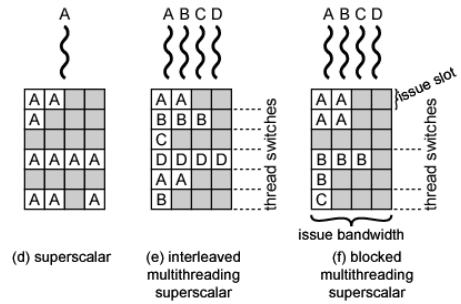
38

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

39

Multiple Instruction Issue Diagram (1)



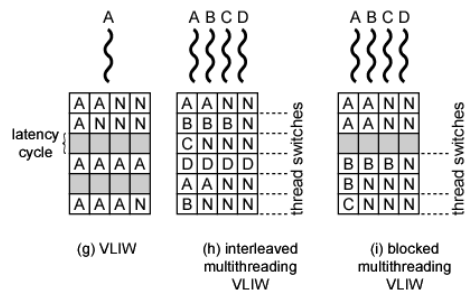
40

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

41

Multiple Instruction Issue Diagram (2)



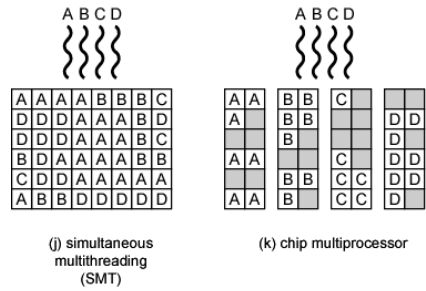
42

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

43

Parallel Diagram



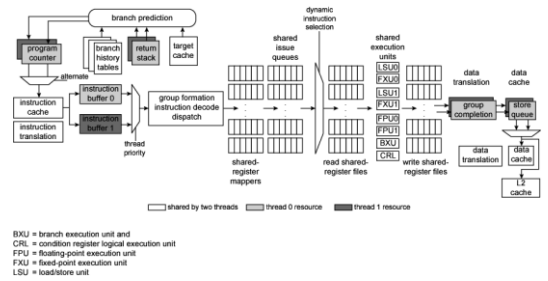
44

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

45

Power 5 Instruction Data Flow



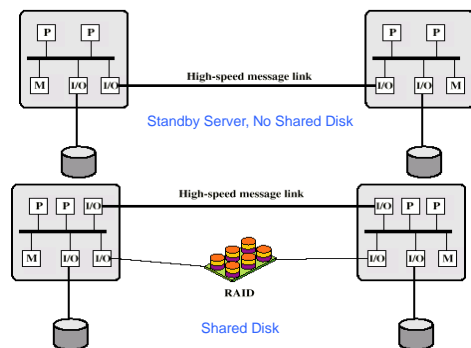
46

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

47

Cluster Configurations



48

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

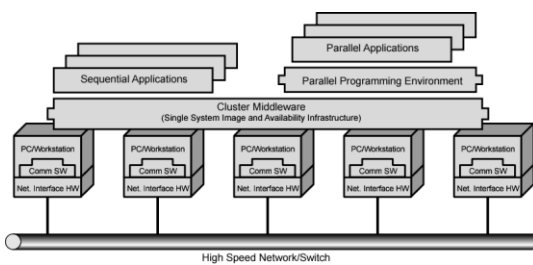
49

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

50

Cluster Computer Architecture



51

Cluster Middleware

- Unified image to user
 - Single system image
- Single point of entry
- Single file hierarchy
- Single control point
- Single virtual networking
- Single memory space
- Single job management system
- Single user interface
- Single I/O space
- Single process space
- Checkpointing
- Process migration

52

Cluster vs 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

53

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

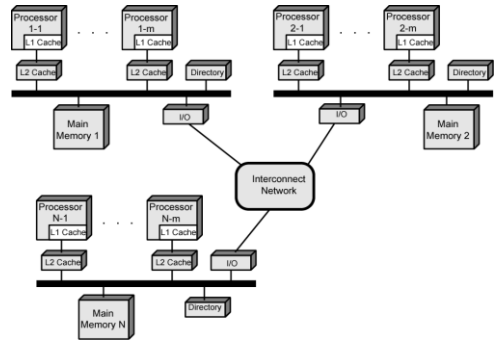
54

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

55

CC-NUMA Organization



56

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

57

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

58

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

59

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?

60

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

61

Vector Addition Example

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

$A + B = C$

62

Approaches

- General purpose computers rely on iteration to do vector calculations
 - In example this needs six calculations
- Vector processing
 - Assume possible to operate on 1 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

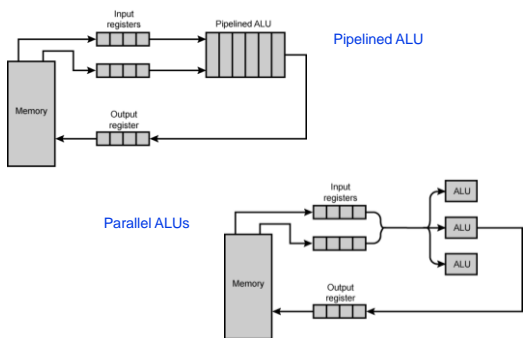
63

Processor Designs

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

64

Approaches to Vector Computation



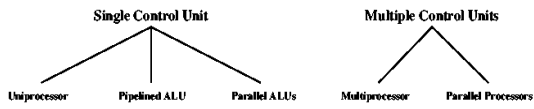
65

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

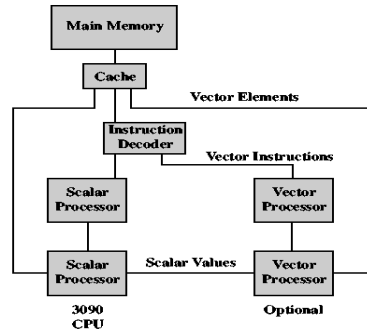
66

Computer Organizations



67

IBM 3090 with Vector Facility



68

69