Computer Architecture

Oguntunde, B.O.
RUN

Course Outline

- Levels of machine design: gates, register and processor levels
- Parallelism
- Multiprocessors and pipelining
- Memory system organization
- Fault tolerance

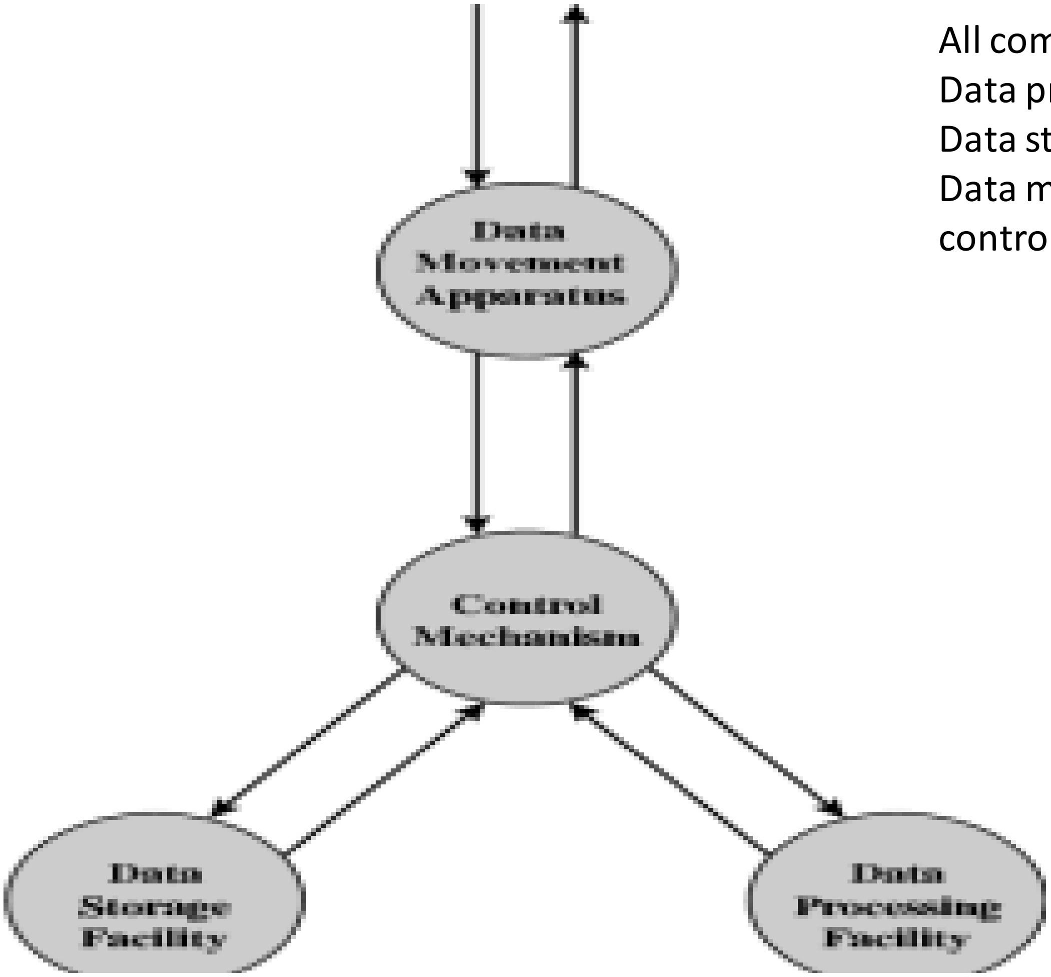
Introduction

- Computer architecture: attributes of a system that have a direct impact on the logical execution of a program. Examples: o the instruction set o the number of bits used to represent various data types o I/O mechanisms o memory addressing techniques
- attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - e.g. Is there a multiply instruction?

- Computer Organization refers to the operational units and their interconnections that realize the architectural specifications. Examples are things that are transparent to the programmer:
  - control signals
  - interfaces between computer and peripherals
  - the memory technology being used

  Organization is how features are implemented o Control signals, interfaces, memory technology. o e.g. Is there a hardware multiply unit or is it done by repeated addition? • All Intel x86 family share the same basic architecture • The IBM System/370 family share the same basic architecture • This gives code compatibility o At least backwards • Organization differs between different versions

Operating Environment (sources and destination of data)



All computer functions are:
Data processing
Data storage
Data movement
control

Structural Components

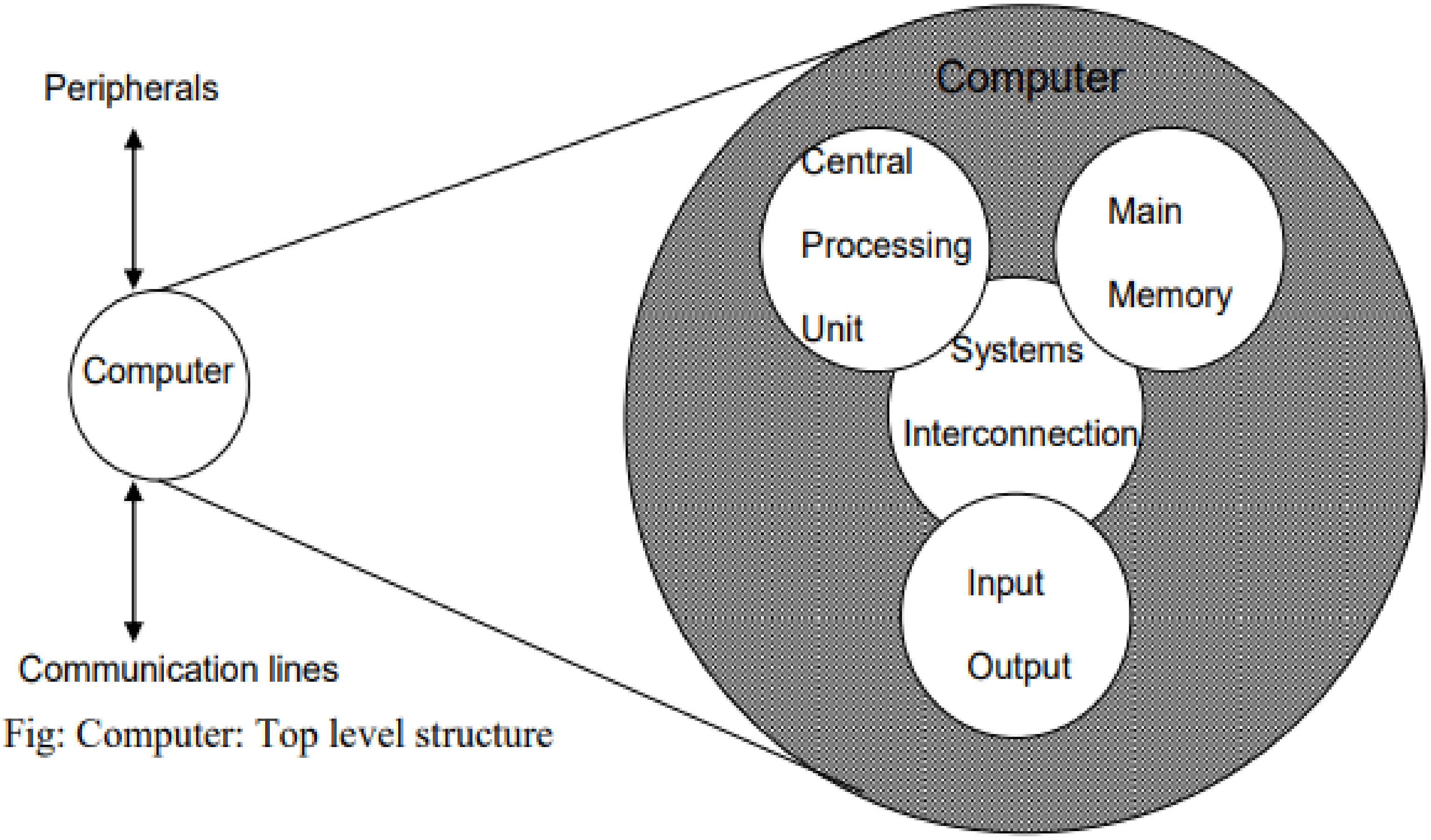


Fig: Computer: Top level structure

CPU structural components

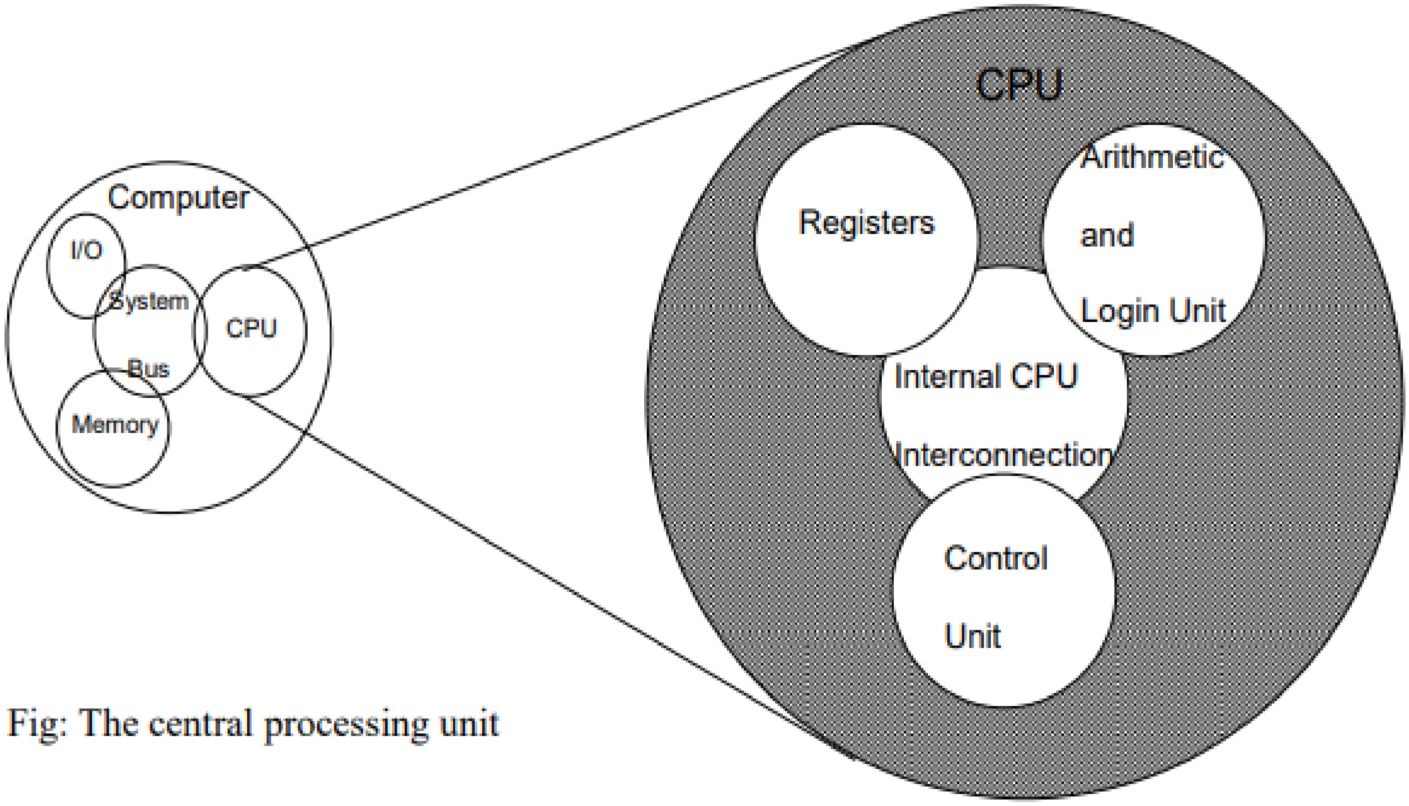


Fig: The central processing unit

Control Unit

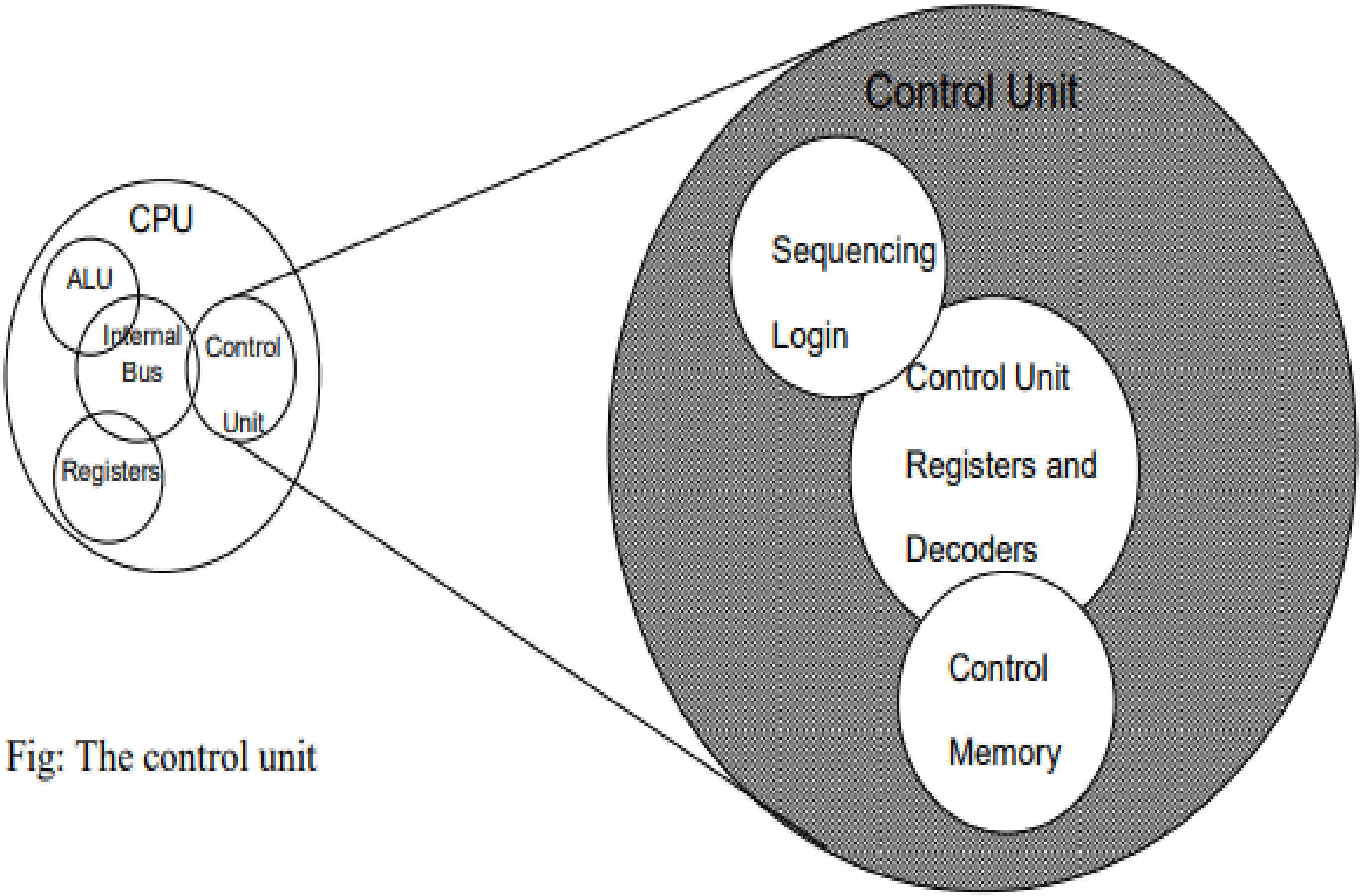


Fig: The control unit

Designing for performance

- Some of the driving factor behind the need to design for performance
  - Microprocessor Spedd
    - Pipeline
    - On board cache, L1 & L2
    - Branch prediction
    - Data flow analysis
    - Speculative execution

    Performance Mismatch

    processor speed increase

    Memory capacity increased

    Memory speed lags behind processor speed

- Solution
  - Increase number of bits retrieved at one time
    - Make DRAM wider rather than deeper to use wide bus data path
  - Change DRAM interface
    - Cache
  - Reduce frequency of memory access
    - More complex cache and cache on chip
  - Increase interconnection bandwidth
    - High speed buses
    - Hierarchy of buses

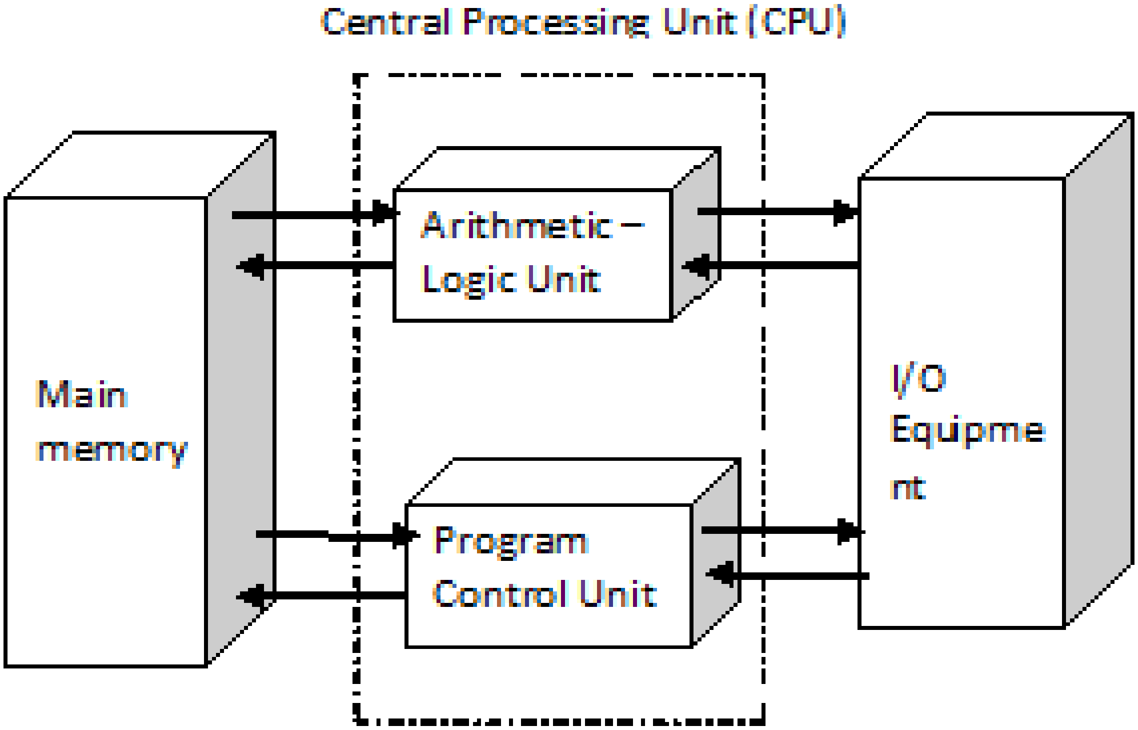
Introduction

- ENIAC (Electronic Numerical Integrator And Computer)
  - First completely electronic, operational general purpose computer
  - Weight : 30 tons
  - Size: 1500 square feet (72 square meters)
  - Power consumption : 140 KW
  - Components: 18,000 vacuum tubes
  - Speed: 5000 additions per second
  - Completed in 1946
  - Performance: read in 120 cards / minute, addition in 200µs, division 6 ms
  - Application: Ballistic calculation
  - Program and data are stored separately
  - Entering and altering program are tedious

Von Neumann Machine

- EDVAC (Electronic Discrete Variable Computer)
- Store program and data in memory
- Program could be set or altered by setting a portion of memory
- The prototype for subsequent general purpose computers

Structure of IAS Computer



Commercial Computers

- UNIVAC I (Universal Automatic Computer)
  - Scientific and commercial applications

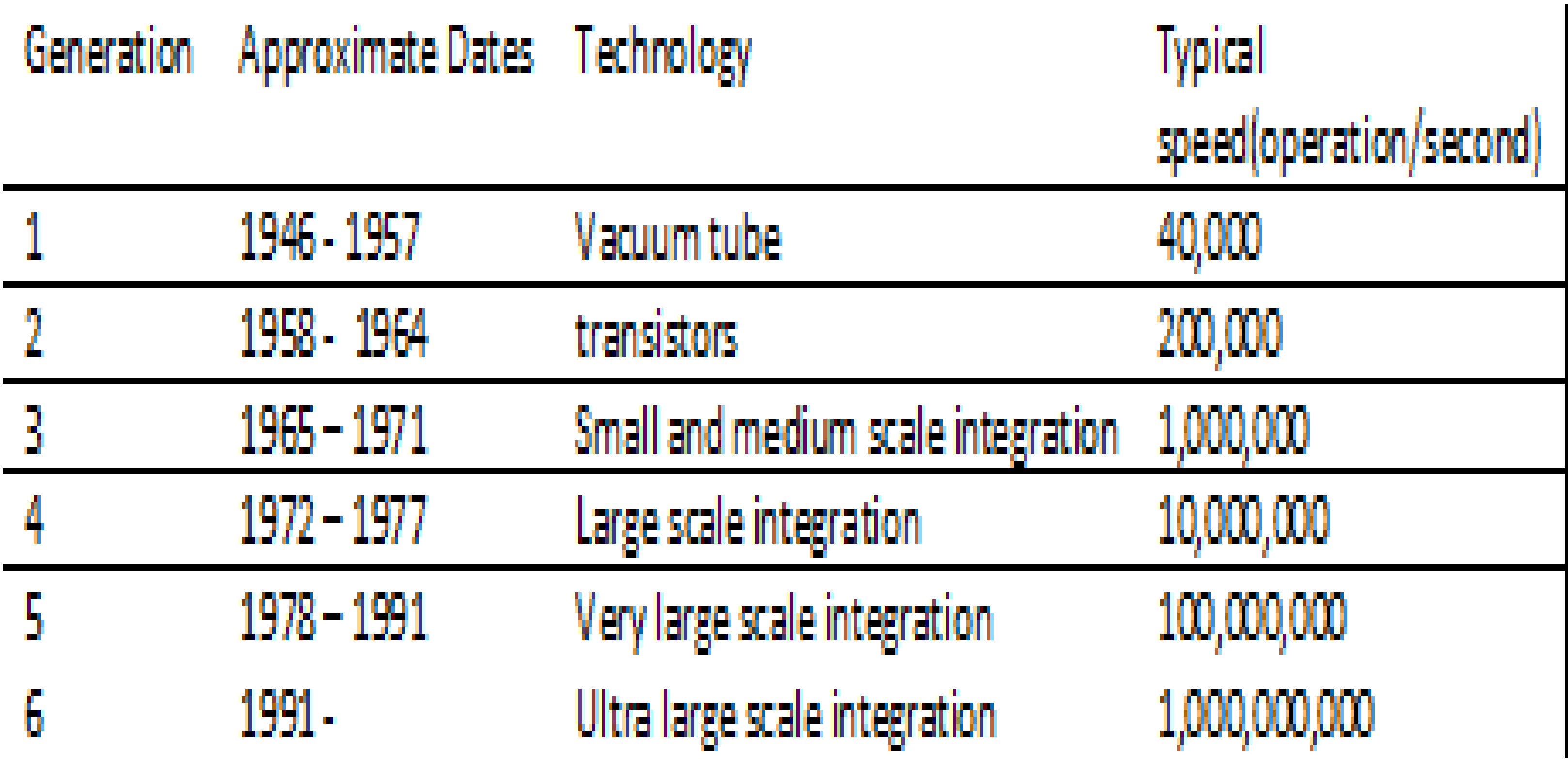
  UNIVAC II

  greater memory, higher performance

  UNIVAC 1100 series

  1103: scientific app involving long and complex calculations

Generations of Computers



Generation	Approximate Dates	Technology	Typical speed(operation/second)
1	1946 - 1957	Vacuum tube	40,000
2	1958 - 1964	transistors	200,000
3	1965 – 1971	Small and medium scale integration	1,000,000
4	1972 – 1977	Large scale integration	10,000,000
5	1978 – 1991	Very large scale integration	100,000,000
6	1991 -	Ultra large scale integration	1,000,000,000

Later generations

- Processors made of IC chips
- Memory made of IC chips
- Density continue to increase
- Performance and capacity improve
- Cost reduces

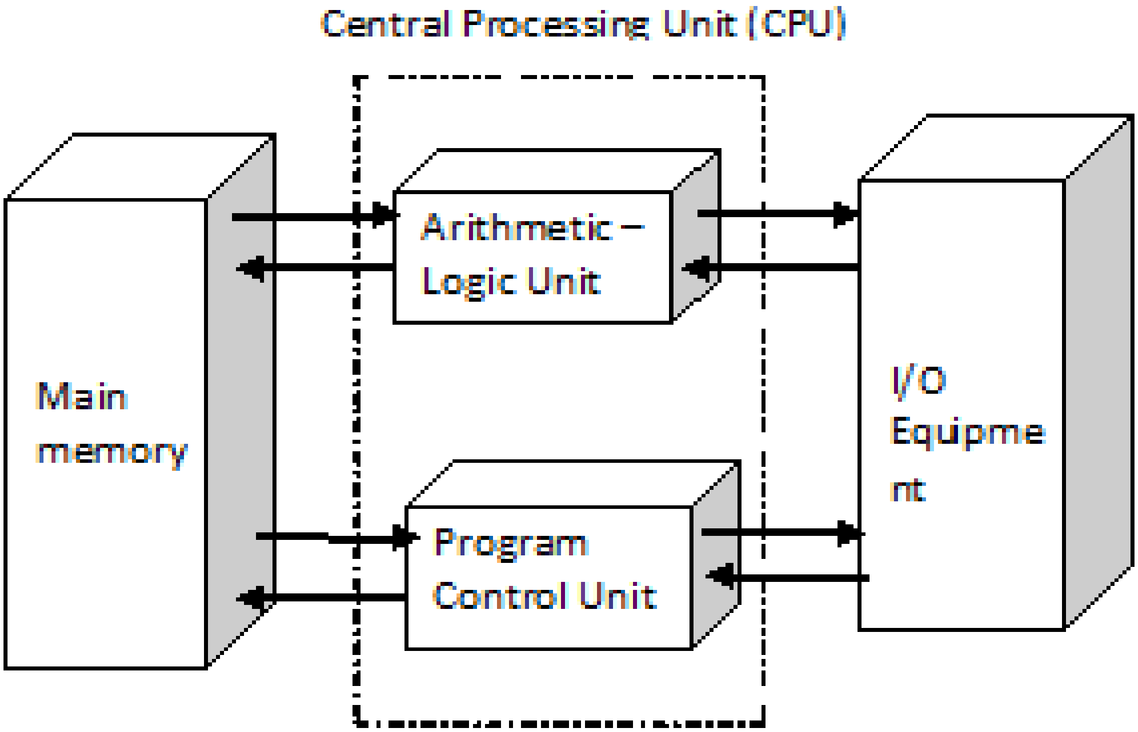
Designing for performance

- Applications require greater power
  - Image processing
  - Speech recognition
  - Videoconferencing
  - Multimedia authoring
  - Voice and video annotation of files
  - Simulation modeling

Performance

- Microprocessor Speed
- Performance Balance
- Improvement in chip organization and Architecture

Basic Structure of Computer system



Computer Architecture’s Changing Definition

- 1950s Computer Architecture
  - Computer Arithmetic
- 1960s
  - Operating system support, especially memory management
- 1970s to mid 1980s Computer Architecture
  - Instruction Set Design, especially ISA appropriate for compilers
  - Vector processing and shared memory multiprocessors

- 1990s Computer Architecture
  - Design of CPU, memory system, I/O system, Multi-processors, Networks
  - Design for VLSI

- 2000s Computer Architecture:
  - Special purpose architectures, Functionally reconfigurable, Special considerations for low power/mobile processing, highly parallel structures

Basic blocks

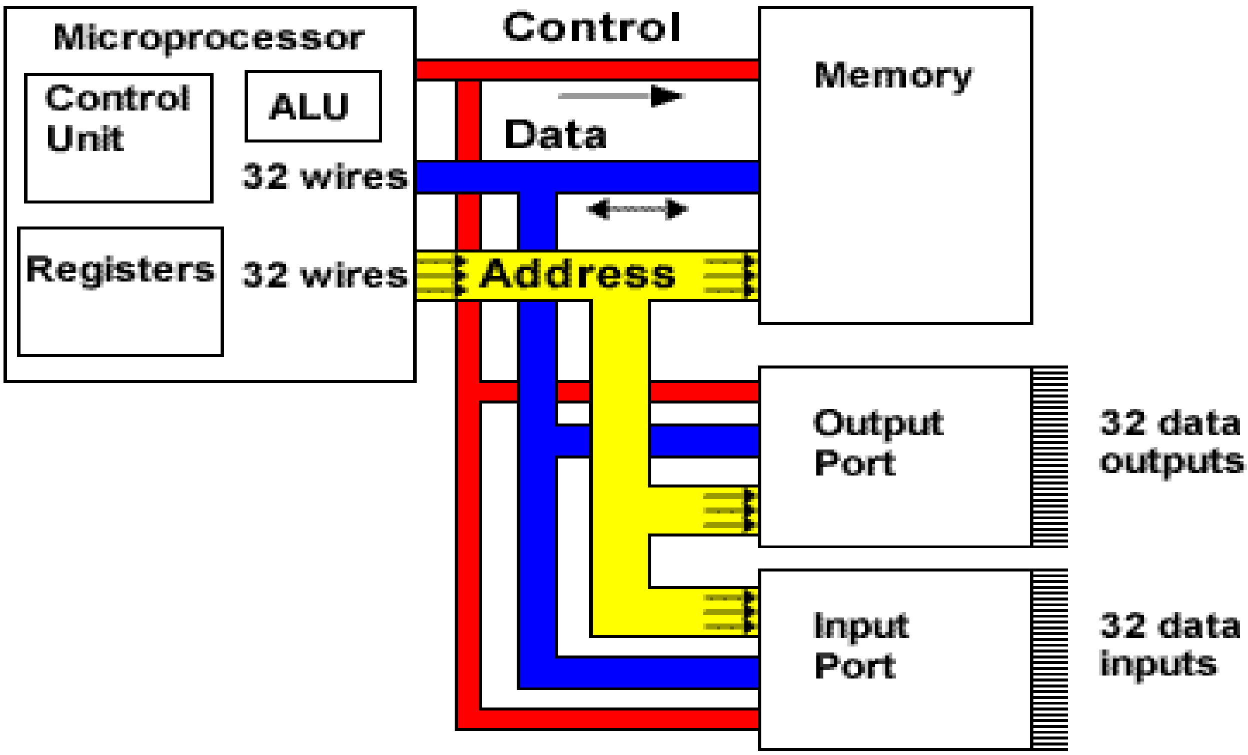
- Central processing unit
- Memory unit
  - ROM
  - RAM

  I/O unit

Microprocessor Bus

- Address bus: unidirectional, selects the address of the source/destination for the data transfer. The address bus is an output from the microprocessor.
- Data bus: bi-directional, transfer between the source and destination, one of which will normally be the microprocessor.
- Control bus: consists of a number of signals that are used to synchronize the operations of the individual microcomputer elements

Processor Schematic Architecture



Registers

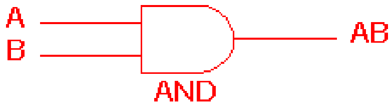
- Control unit and ALU contain storage locations called registers
- Memory buffer register (MBR):
- Memory address register (MAR):
- Instruction register (IR):
- Instruction buffer register (IBR):
- Program counter (PC):
- Accumulator (AC) and multiplier quotient (MQ)

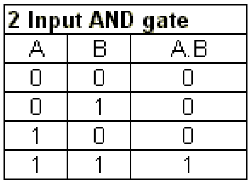
Analog and digital systems

- Analog: continuous values
- Digital: discrete values

Fundamental gates

- Gate: 2- state device, open/close
- AND gate

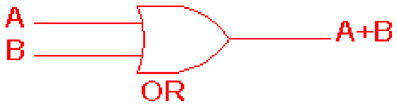


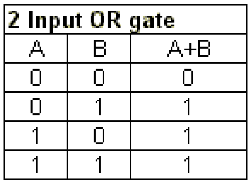


2 Input AND gate

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

- OR gate

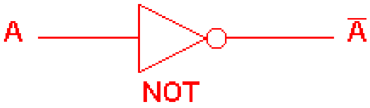


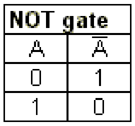


2 Input OR gate

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

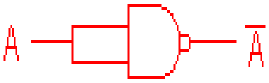
- NOT gate

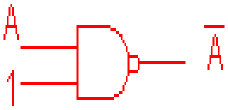




NOT gate

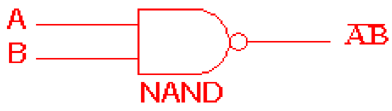
A	$\overline{A}$
0	1
1	0

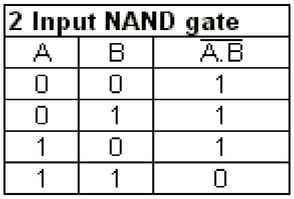




Logic Gates cnt

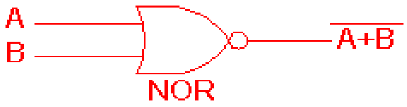
- NAND gate

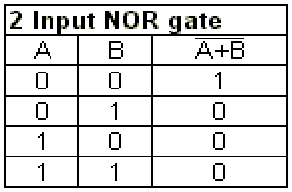



2 Input NAND gate

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

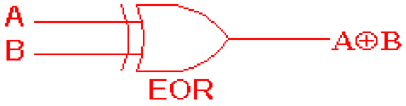
- NOR gate

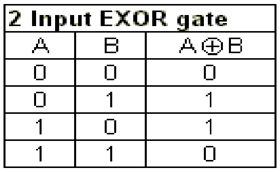



2 Input NOR gate

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

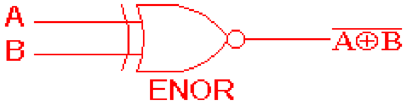
- EXOR gate

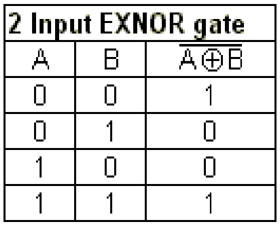



2 Input EXOR gate

A	B	$A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

- EXNOR gate




2 Input EXNOR gate

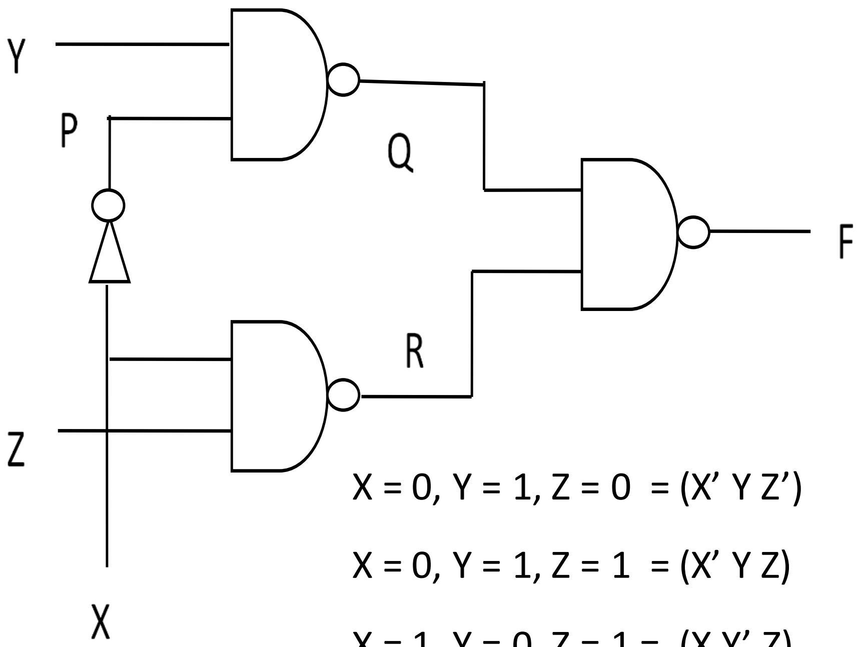
A	B	$\overline{A\oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Application of Gates

- Circuits are constructed from connecting gates together.
- The output from one gate can be connected to the input of one or more other gates.
- Two outputs however cannot be connected together.

DERIVATION

- To derive expressions for output in terms of its input can be done in two ways
- From the circuit diagram by writing the output of each gate in terms of its inputs
- From the truth table, each time a logic one appears in the output column, write down the set of inputs that cause the output to be true.



X = 0, Y = 1, Z = 0 = (X’ Y Z’)

X = 0, Y = 1, Z = 1 = (X’ Y Z)

X = 1, Y = 0, Z = 1 = (X Y’ Z)

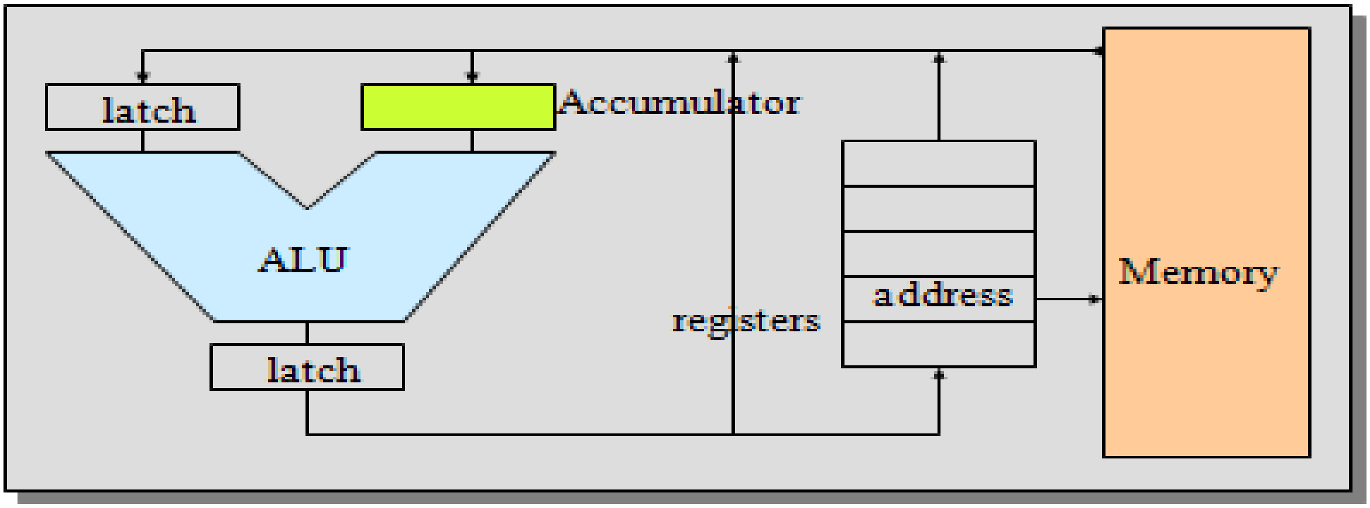
X = 1, Y = 1 Z =1 =   (X Y Z)

X	Y	Z	P = X	Q= P.Y	R = X.Z	F= Q.R
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	1	0	1	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	0
1	0	1	0	1	0	1
1	1	0	0	1	1	0
1	1	1	0	1	0	1

Computer Architecture

- Computer hardware comprises of the components from which computers are built, i.e computer organization
- The science of integrating those components to achieve a level of functionality and performance.
- The structure and organization of a computer’s hardware or software systems, or the structure and organization of different components of a computer system.
- It is the combination of machine organization and instruction set architecture.
- In its broadest definition, computer architecture is the *design of the abstraction layers* that allow us to implement information processing applications efficiently using available manufacturing technologies.

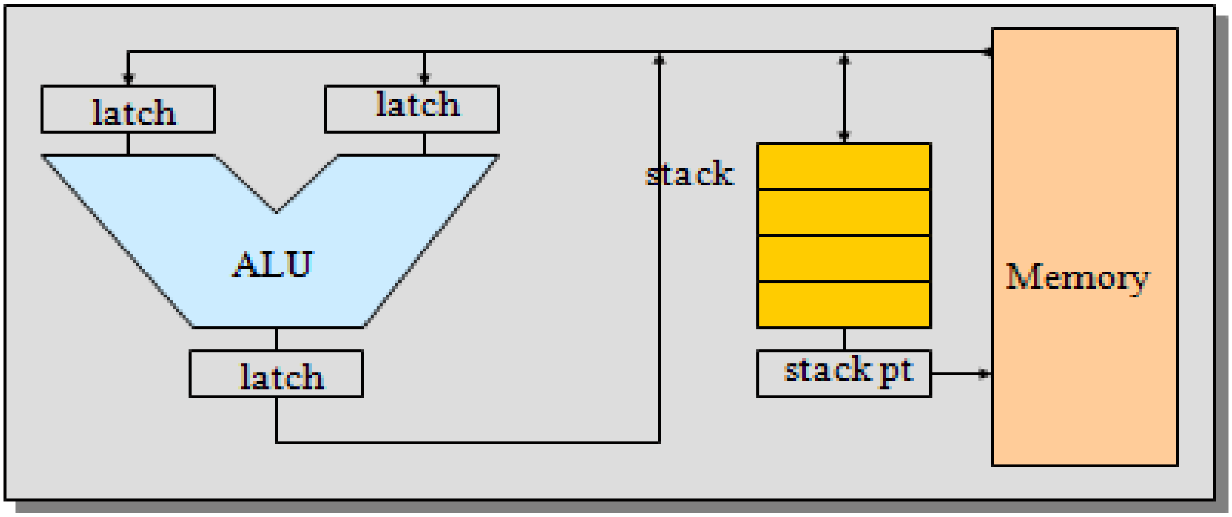
Accumulator Architecture



Example code: `a = b+c;`

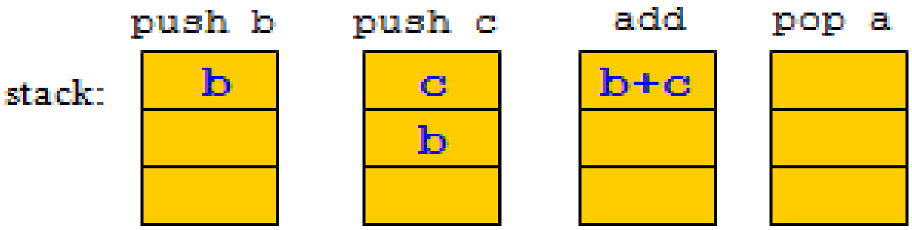
```
load  b;    // accumulator is implicit operand
add   c;
store a;
```

Stack Architecture



Example code: `a = b+c;`

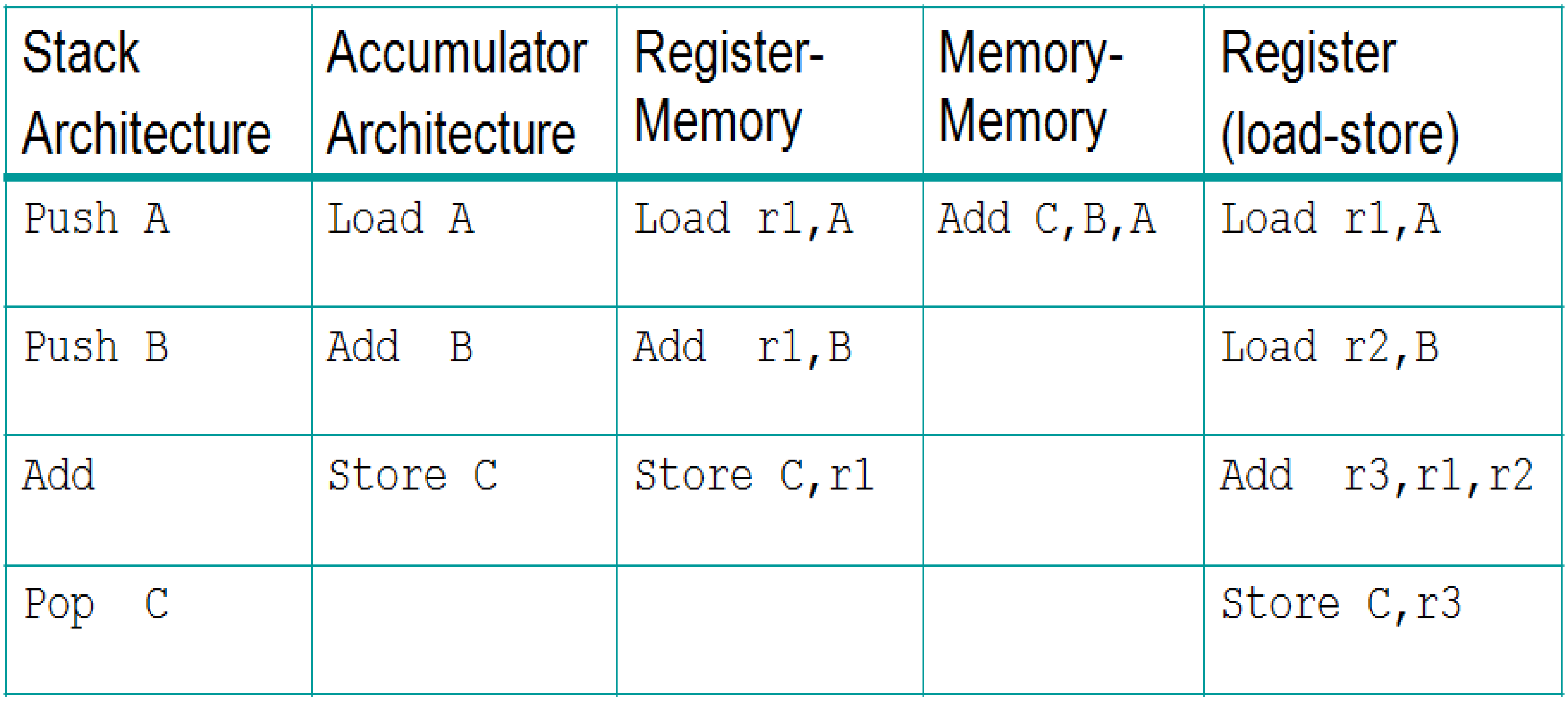
```
push b;
push c;
add;
pop  a;
```



Other architecture styles

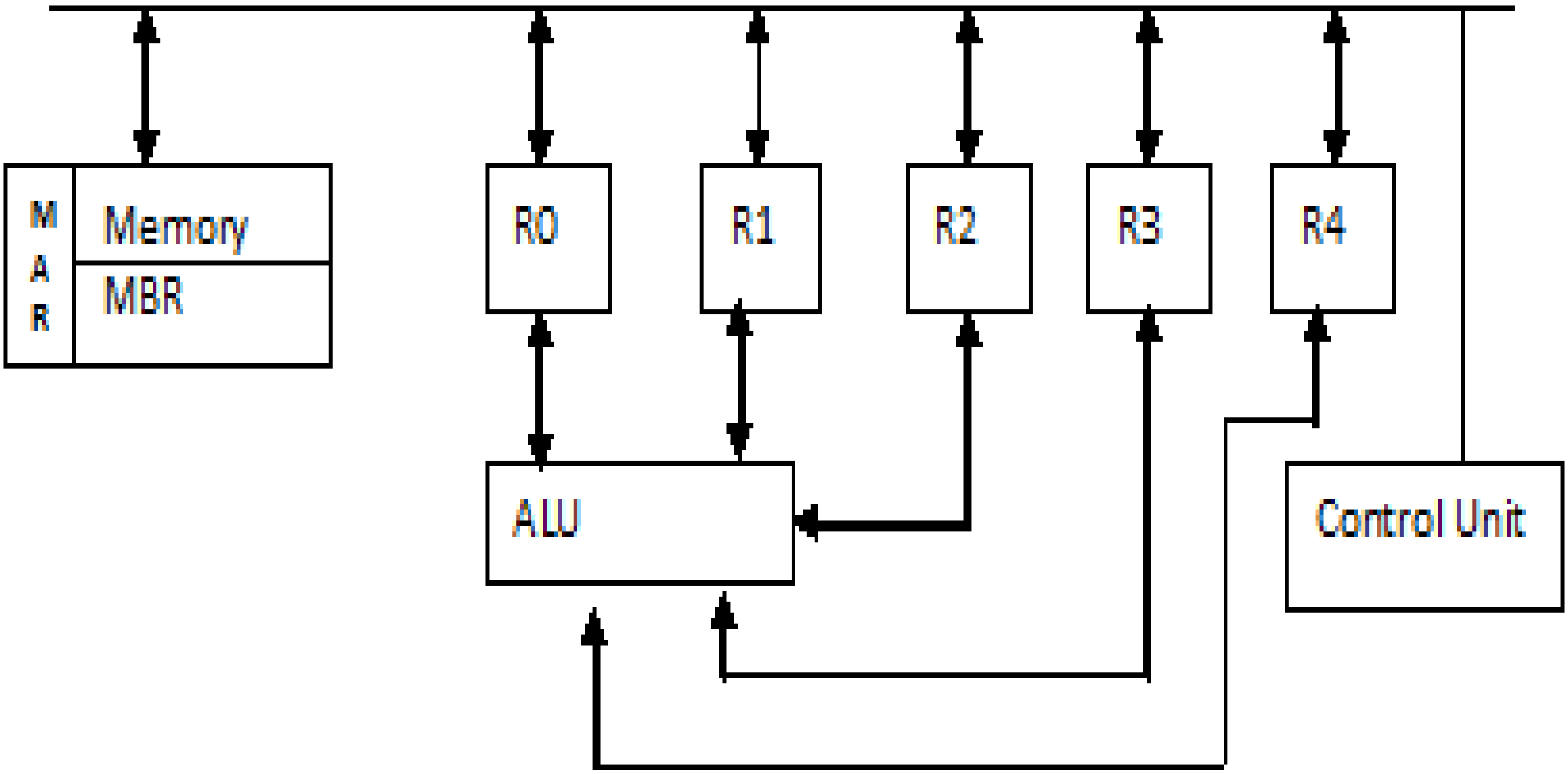
- Accumulator architecture
  - one operand (in register or memory), accumulator almost always implicitly used
- Stack
  - zero operand: all operands implicit (on TOS)
- Register (load store)
  - three operands, all in registers
  - loads and stores are the only instructions accessing memory (i.e. with a memory (indirect) addressing mode
- Register-Memory
  - two operands, one in memory
- Memory-Memory
  - three operands, may be all in memory

Let's look at the code for C = A + B



Stack Architecture	Accumulator Architecture	Register-Memory	Memory-Memory	Register (load-store)
`Push A`	`Load A`	`Load r1,A`	`Add C,B,A`	`Load r1,A`
`Push B`	`Add  B`	`Add  r1,B`		`Load r2,B`
`Add`	`Store C`	`Store C,r1`		`Add  r3,r1,r2`
`Pop  C`				`Store C,r3`

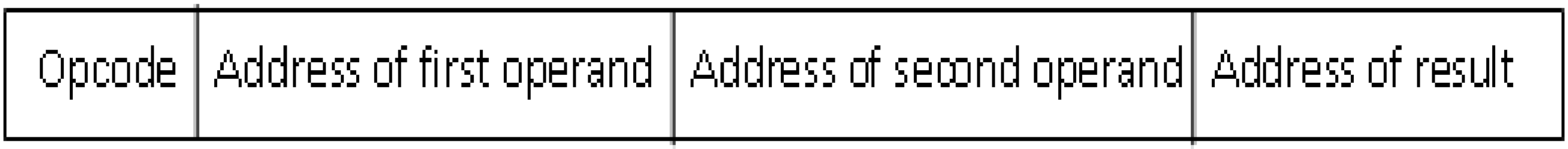
Primitive Machines



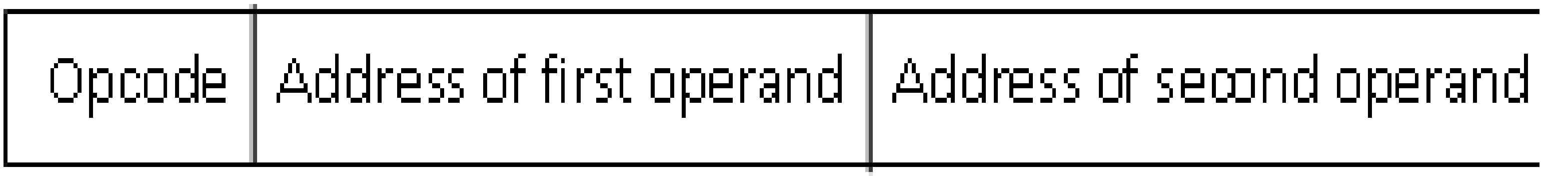
primitive machines

- Advantages of primitive machines
  - simple and systematic machine operation
  - All the functions may have the same format
  - A relatively small number of operation code is needed to specify the essential operations
- Disadvantages
  - It is wastes memory space,
  - waste memory access time
  - wasteful of information transfer.

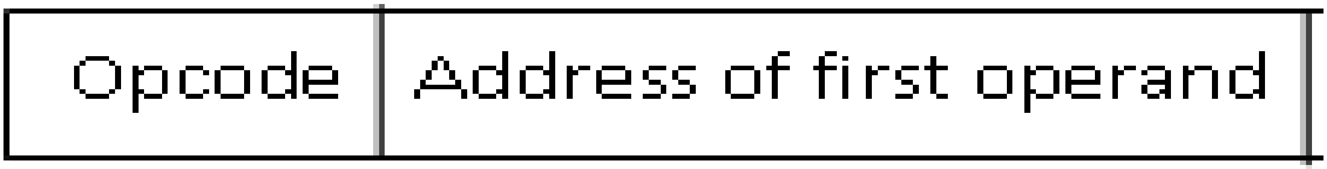
Machine Addressing



Opcode	Address of first operand	Address of second operand	Address of result

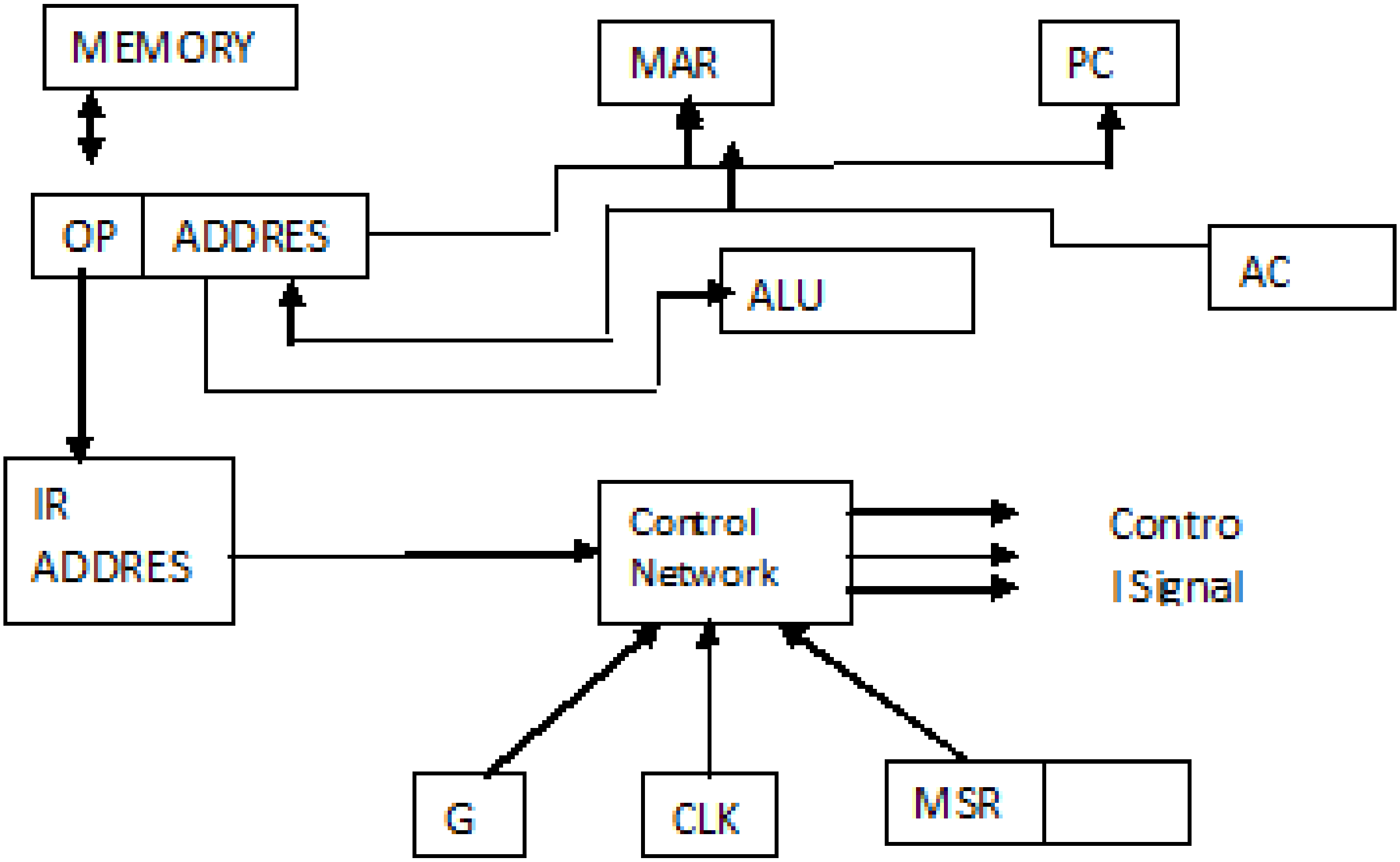


Opcode	Address of first operand	Address of second operand



Opcode	Address of first operand

Classical Machine Architecture



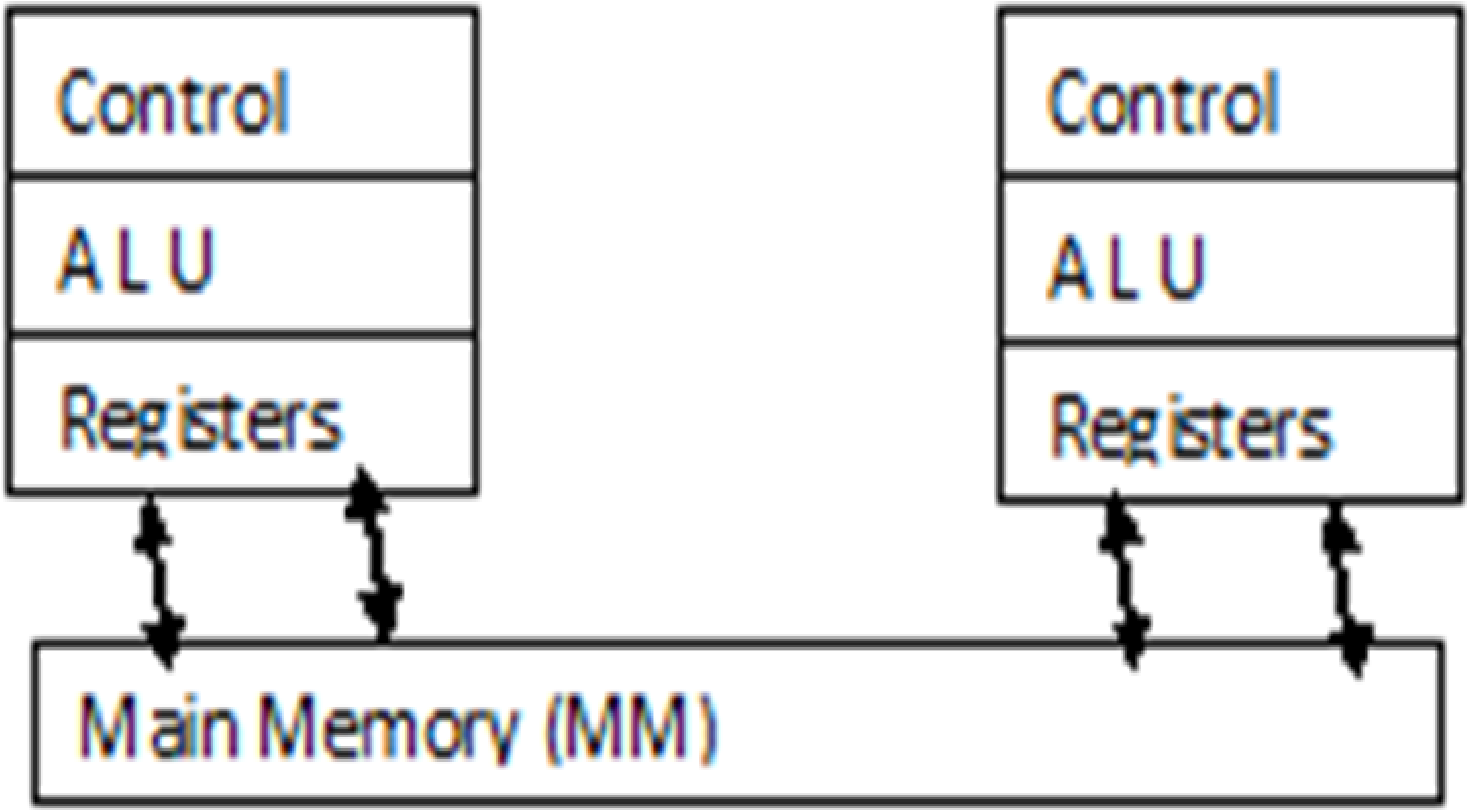
Modern Computer Organisation

- technological advancement
- reduced the cost of digital electronics
- wastefulness of hardware components
  - Improve overall performance in speed, precision, versatility.
  - Simple design and construction, thus economical.
  - easier and cheaper programming

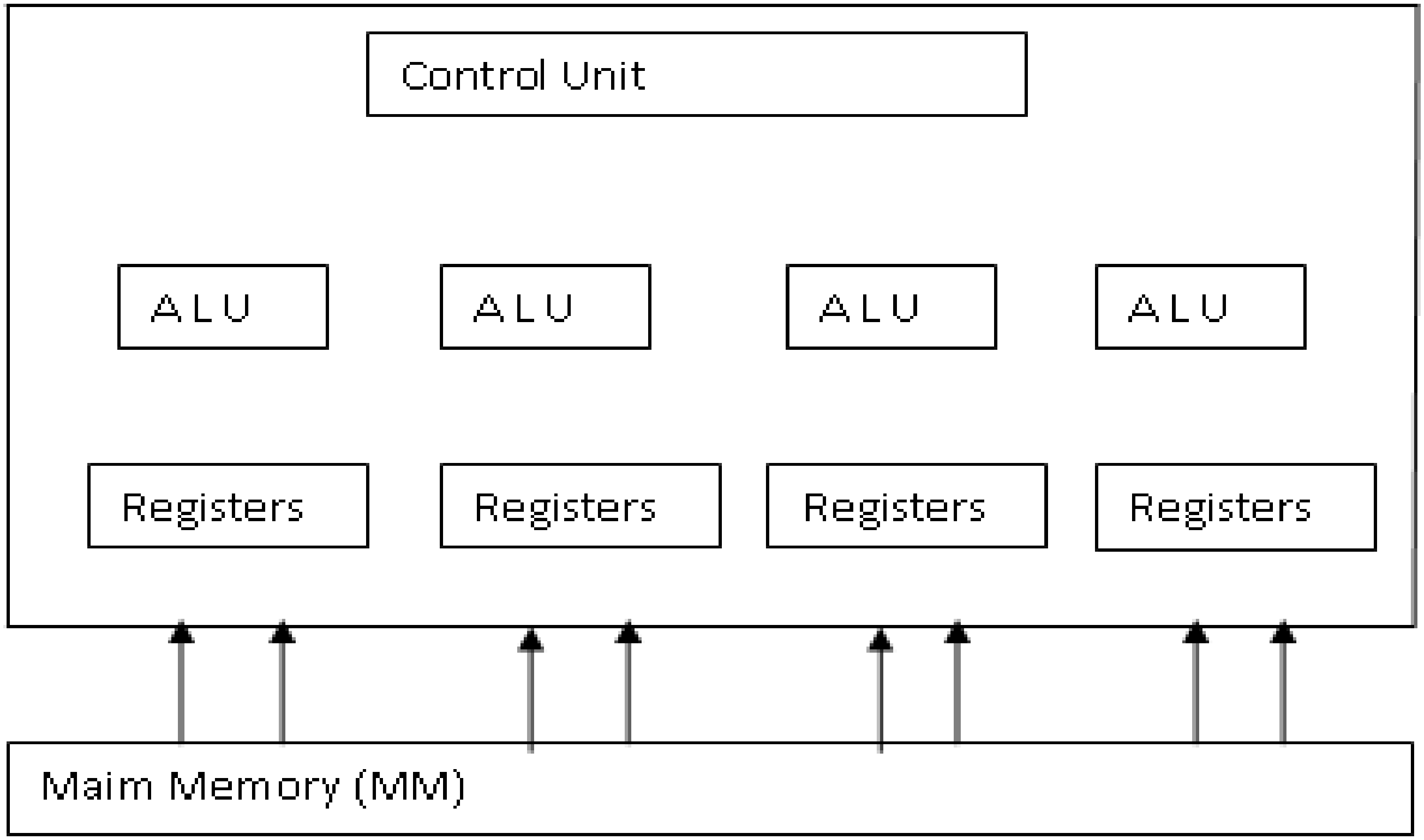
Features

- Modular design
  - parallel or pipeline operation
- Micro-programmed control
- Memory hierarchy

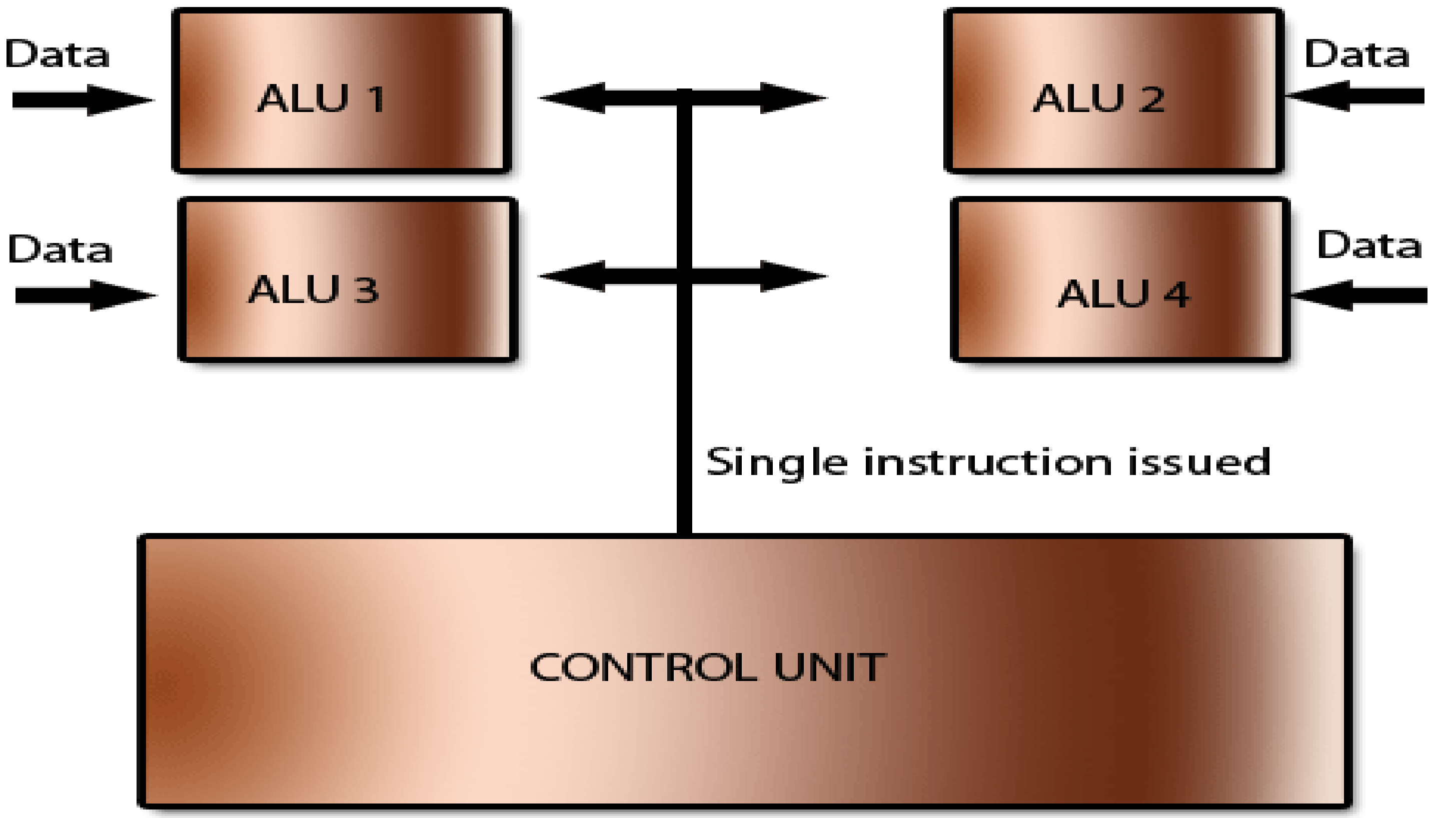
MULTIPROCESSOR SYSTEM



SIMD


Control Unit
ALU
ALU
ALU
ALU
Registers
Registers
Registers
Registers
Maim Memory (MM)

SIMD



An array processor - a Single Instruction Multiple Data computer

(c)www.teach-ict.com

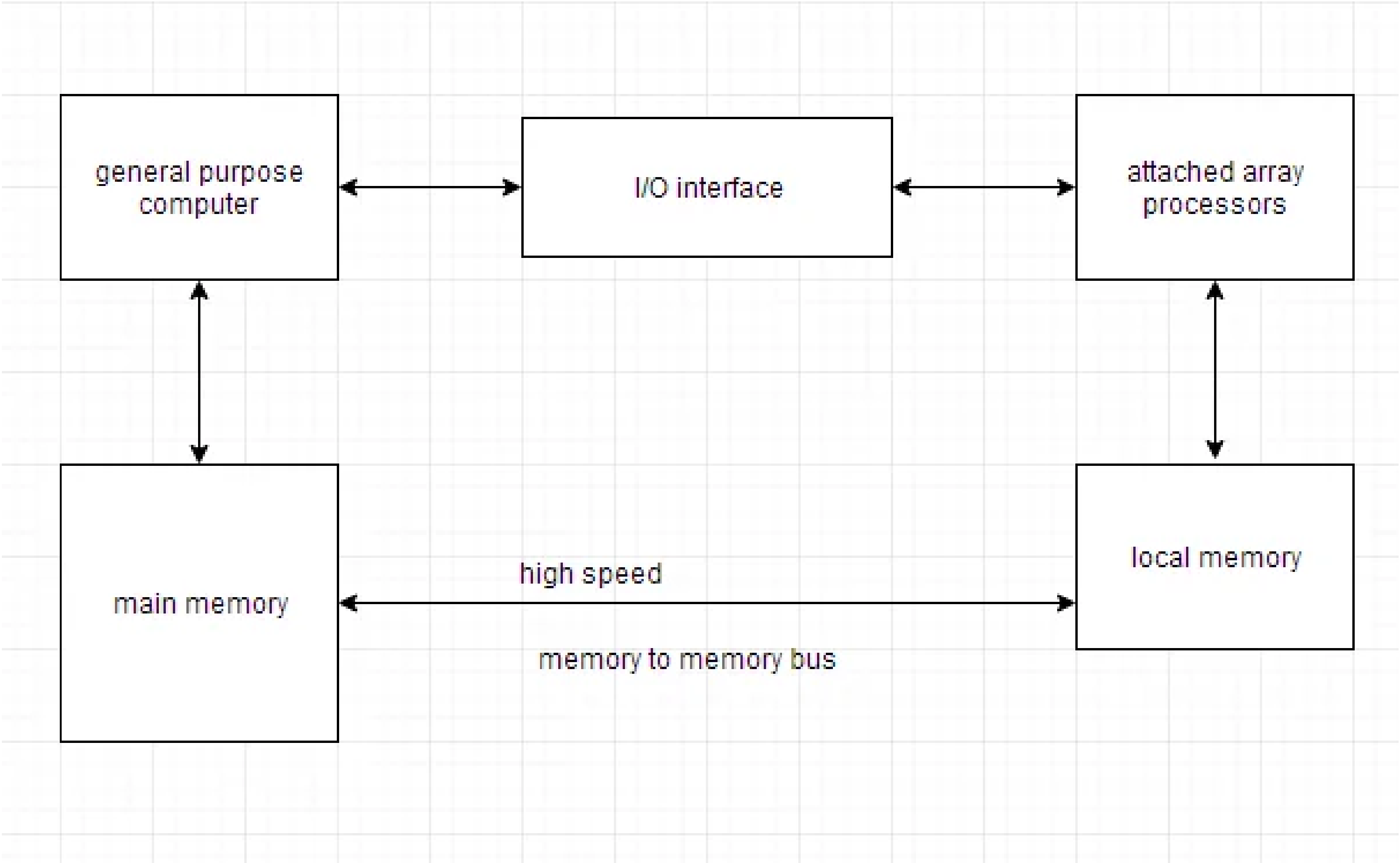
Array processors

- Array processors are also known as multiprocessors or vector processors. They perform computations on large arrays of data. Thus, they are used to improve the performance of the computer.
- Two types of array processors:

1. Attached Array Processors
2. SIMD Array Processors

attached array processor

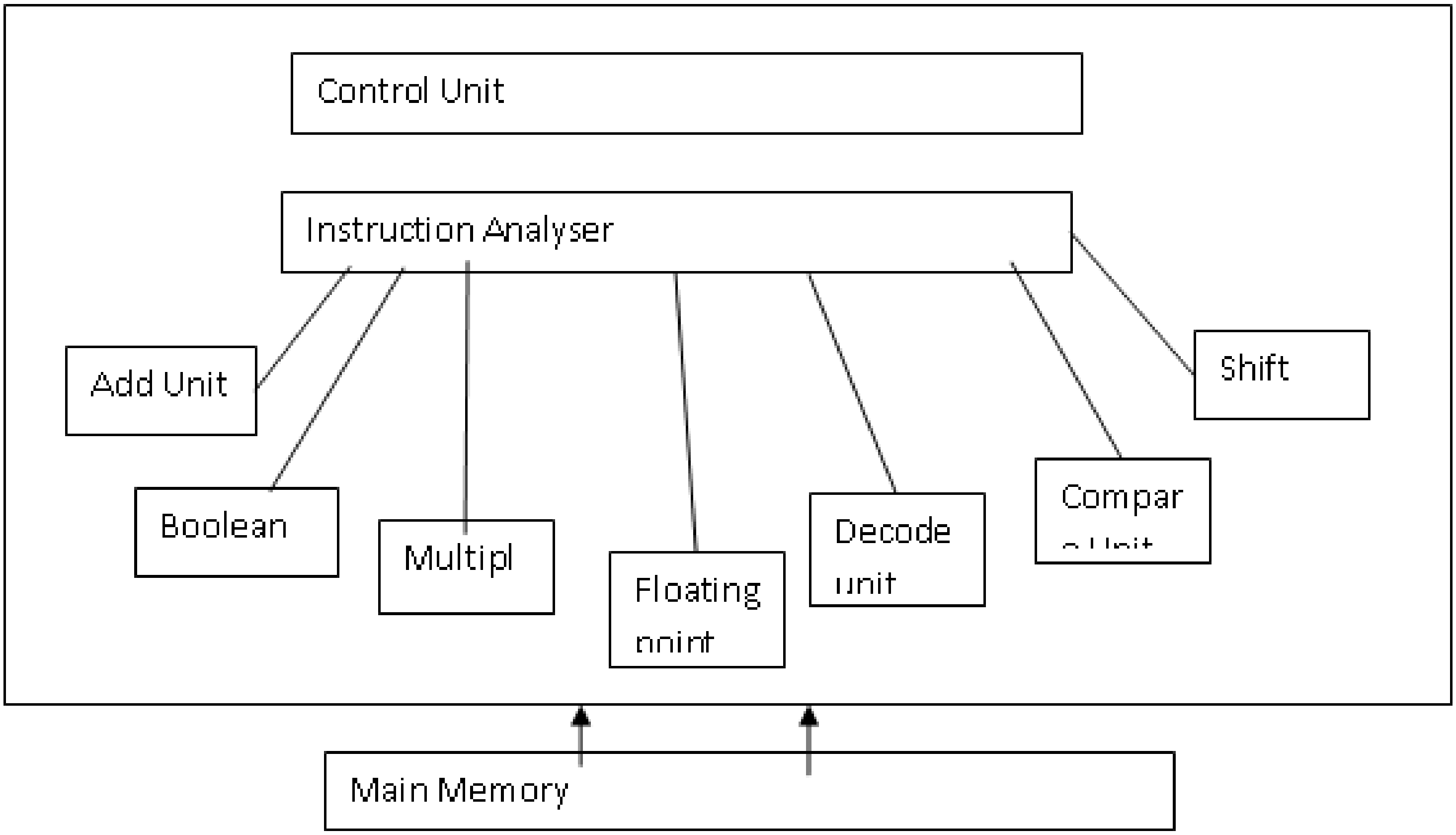
- An attached array processor is a processor which is attached to a general purpose computer and its purpose is to enhance and improve the performance of that computer in numerical computational tasks. It achieves high performance by means of parallel processing with multiple functional units.


general purpose computer
I/O interface
attached array processors
main memory
high speed
memory to memory bus
local memory

SIMD Array Processors

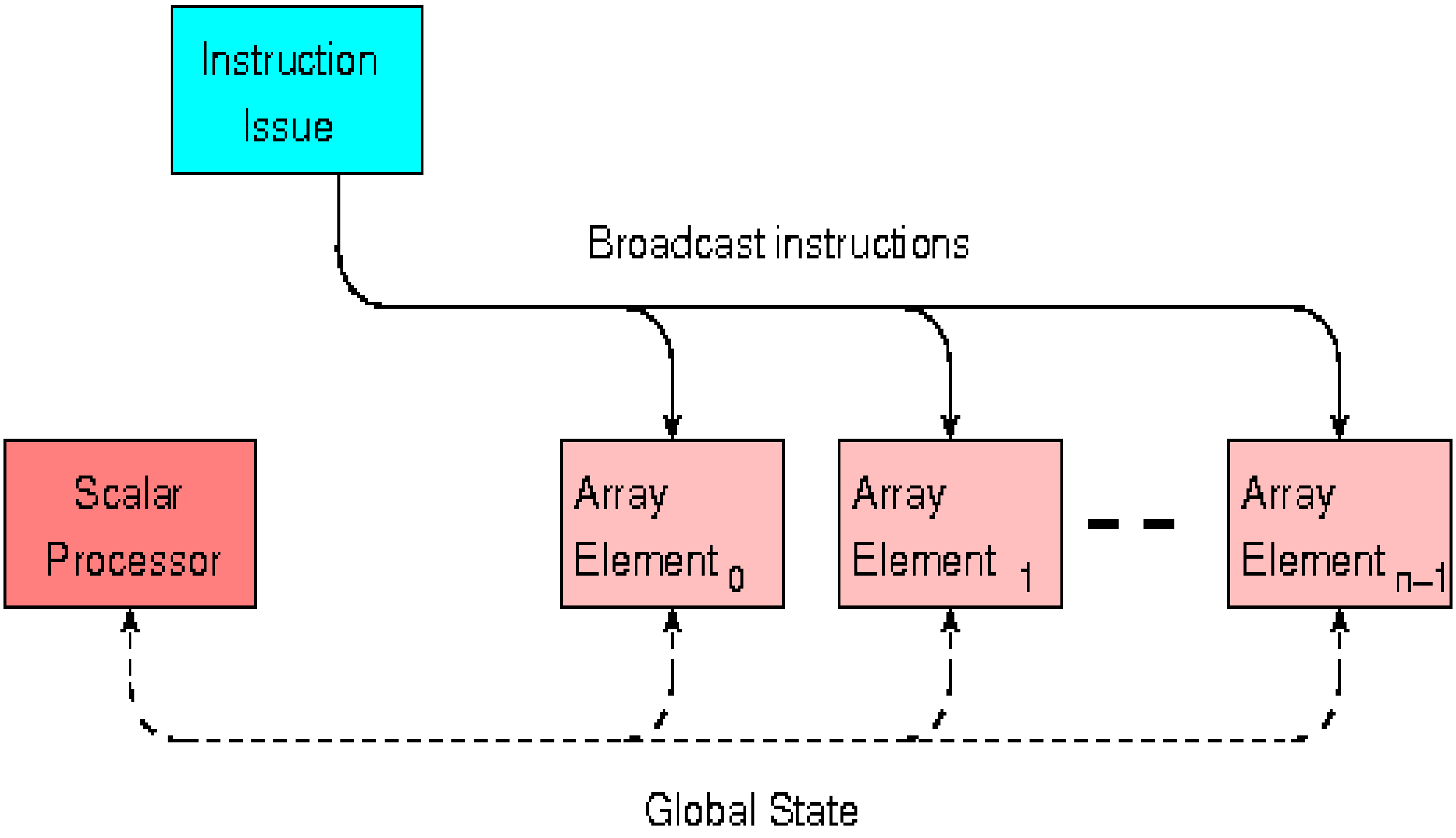
- SIMD is the organization of a single computer containing multiple processors operating in parallel. The processing units are made to operate under the control of a common control unit, thus providing a single instruction stream and multiple data streams.
- A general block diagram of an array processor is shown below. It contains a set of identical processing elements (PE's), each of which is having a local memory M. Each processor element includes an **ALU** and **registers**. The master control unit controls all the operations of the processor elements. It also decodes the instructions and determines how the instruction is to be executed.
- The main memory is used for storing the program. The control unit is responsible for fetching the instructions. Vector instructions are send to all PE's simultaneously and results are returned to the memory.
- The best known SIMD array processor is the **ILLIAC IV** computer developed by the **Burroughs corps**. SIMD processors are highly specialized computers. They are only suitable for numerical problems that can be expressed in vector or matrix form and they are not suitable for other types of computations.

Array Processor

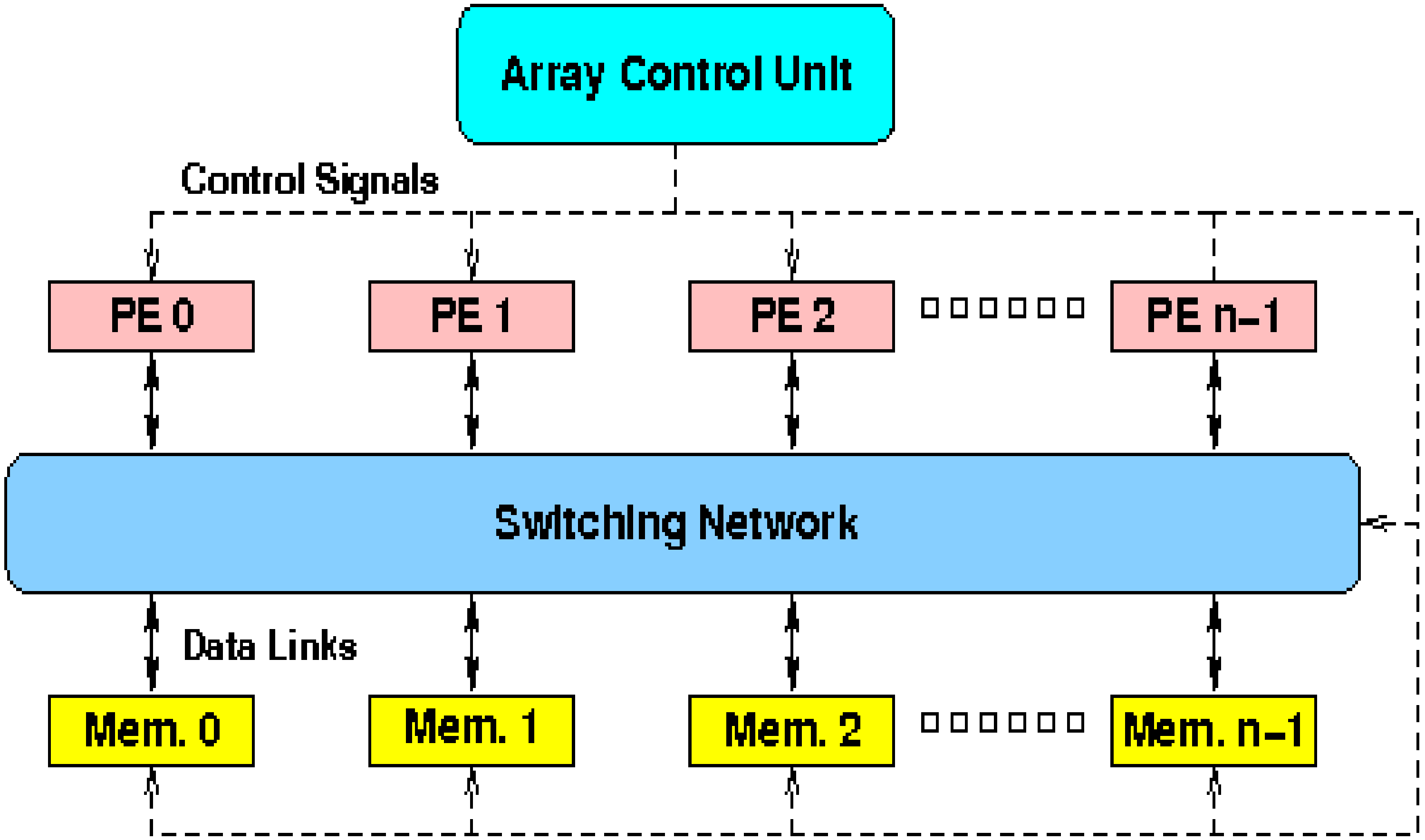


Single CPU with multiple functional units

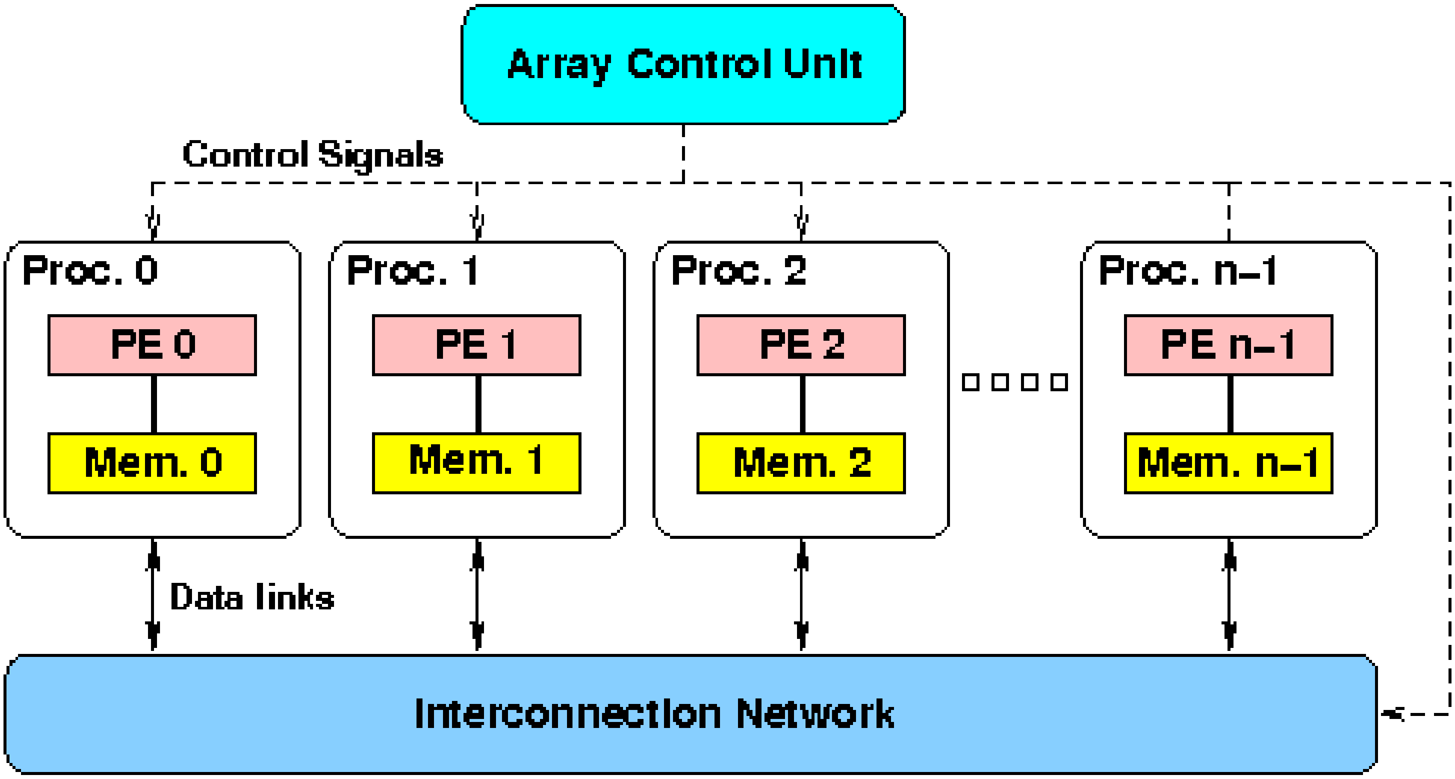
Classical SIMD Array Architecture



Array Processor with Shared Memory



Array Processor with Distributed Memory

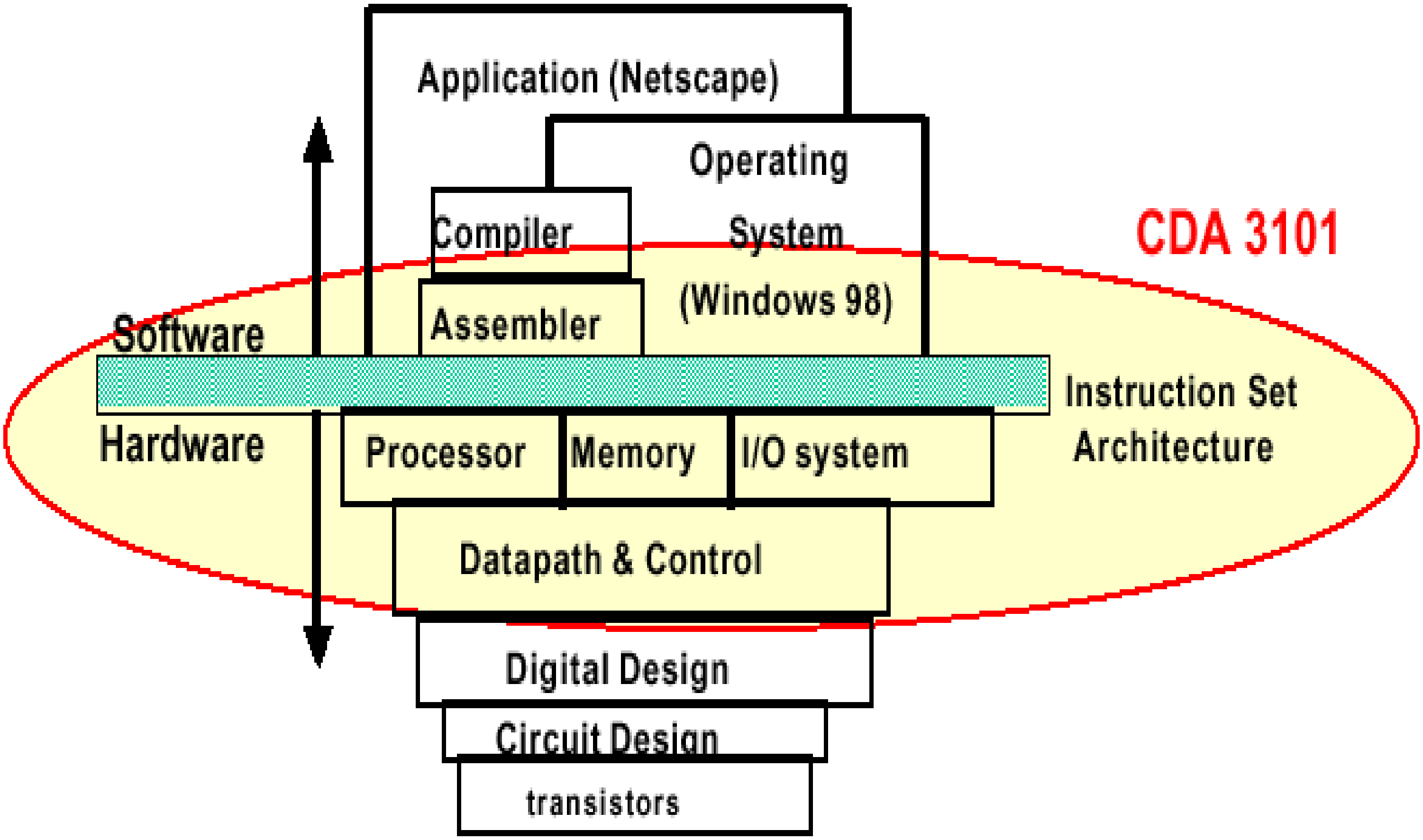


Why use the Array Processor

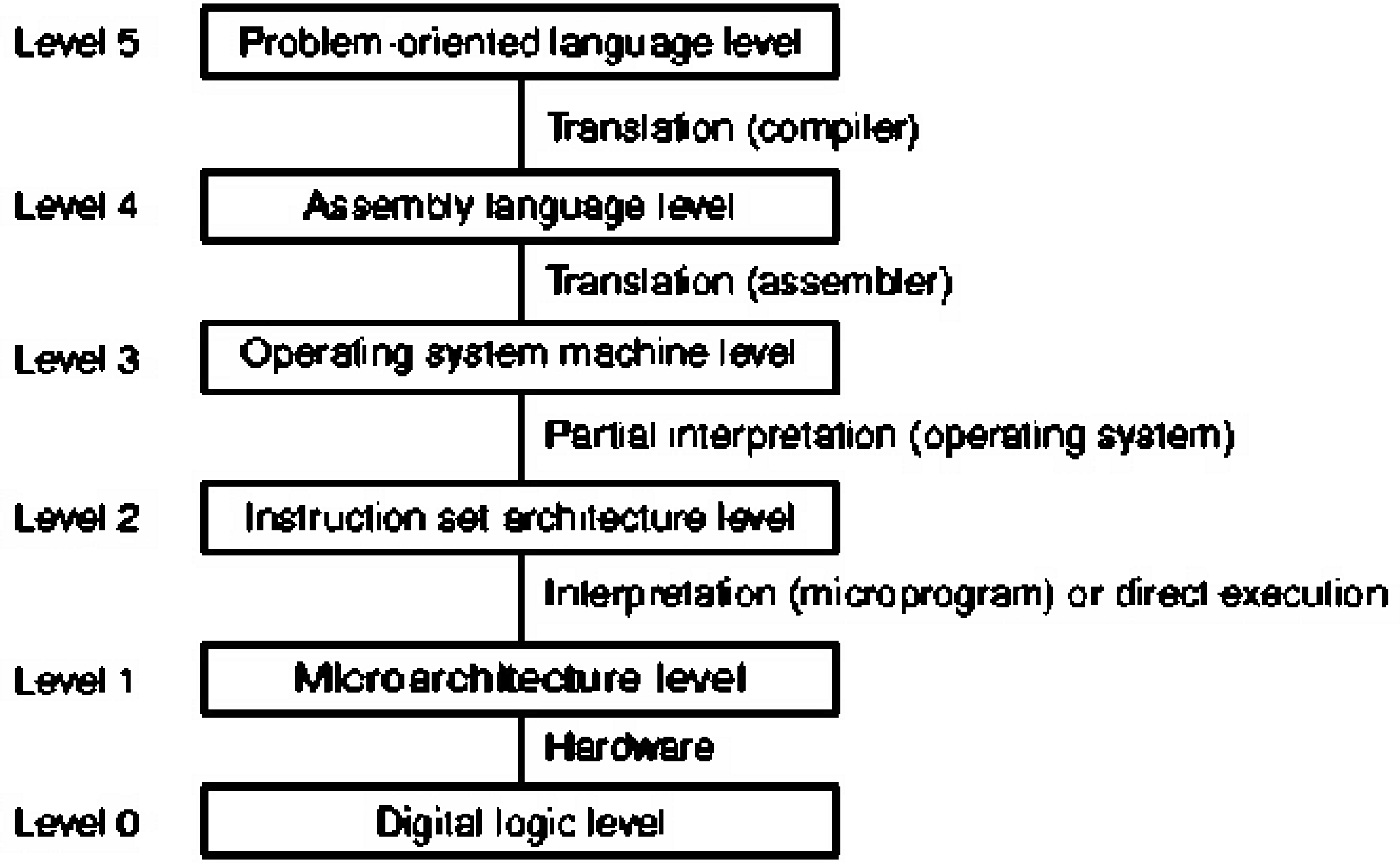
- Array processors increases the overall instruction processing speed.
- As most of the Array processors operates asynchronously from the host CPU, hence it improves the overall capacity of the system.
- Array Processors has its own local memory, hence providing extra memory for systems with low memory.

Level of Abstraction

Overview of Computer Abstraction



Levels of Abstraction



Level 5 | Problem-oriented language level

Translation (compiler)

Level 4 | Assembly language level

Translation (assembler)

Level 3 | Operating system machine level

Partial interpretation (operating system)

Level 2 | Instruction set architecture level

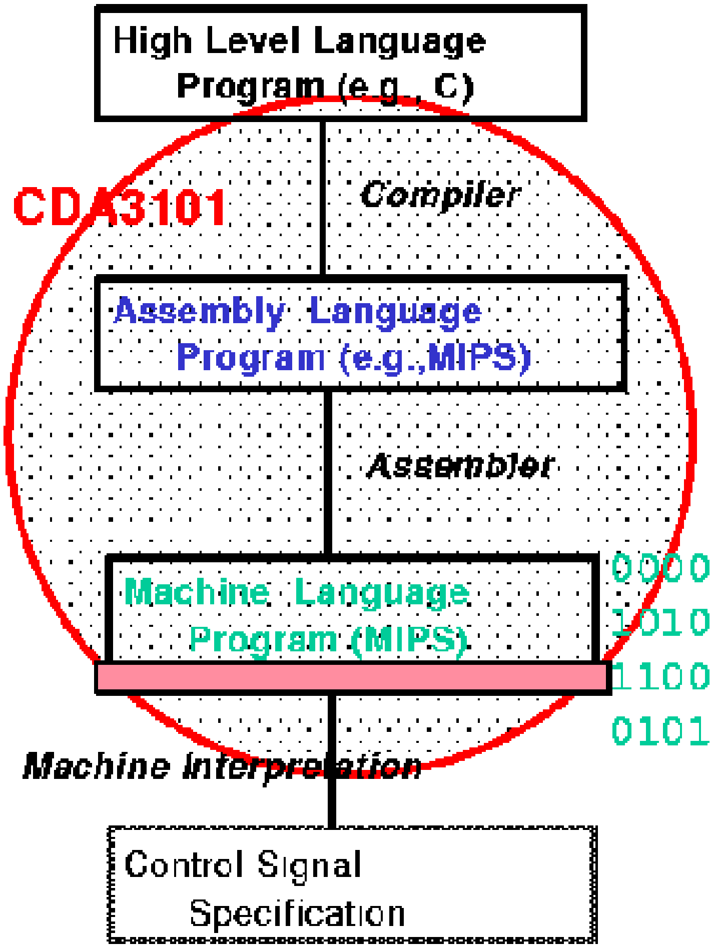
Interpretation (microprogram) or direct execution

Level 1 | Microarchitecture level

Hardware

Level 0 | Digital logic level

Levels of Abstraction cntd


High Level Language Program (e.g., C)
Compiler
CDA3101
Assembly Language Program (e.g.,MIPS)
Assembler
Machine Language Program (MIPS)
Machine Interpretation
Control Signal Specification

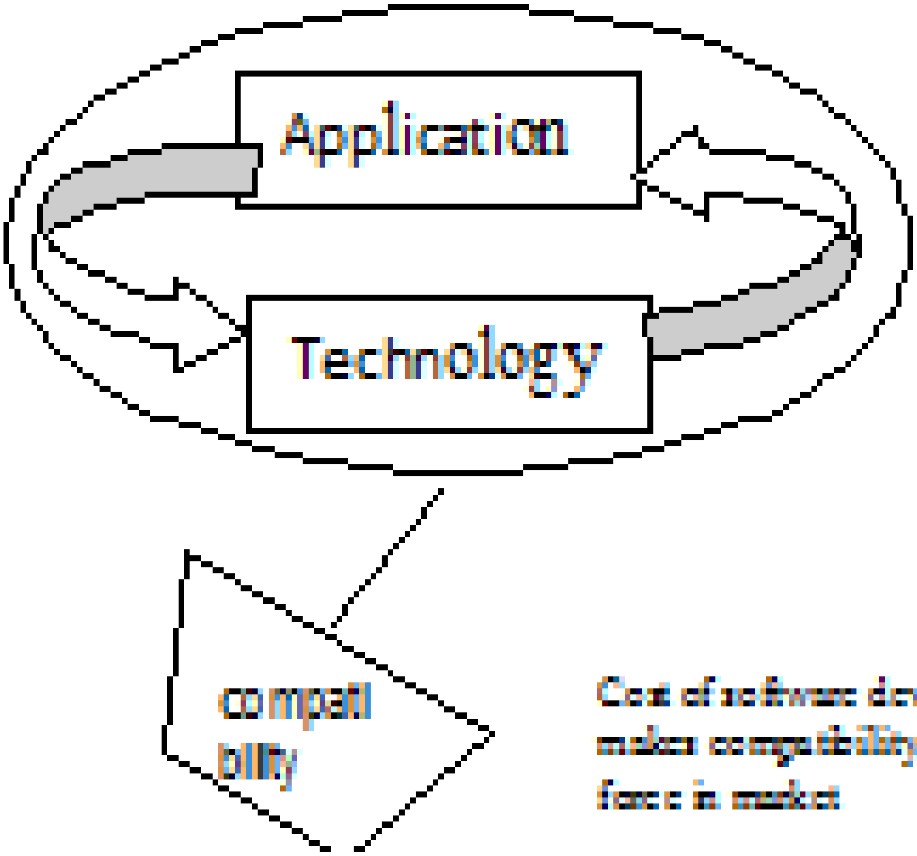
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw$to,   0($2)
lw$t1,   4($2)
sw$t1,   0($2)
sw$t0,   4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Continual change in architecture

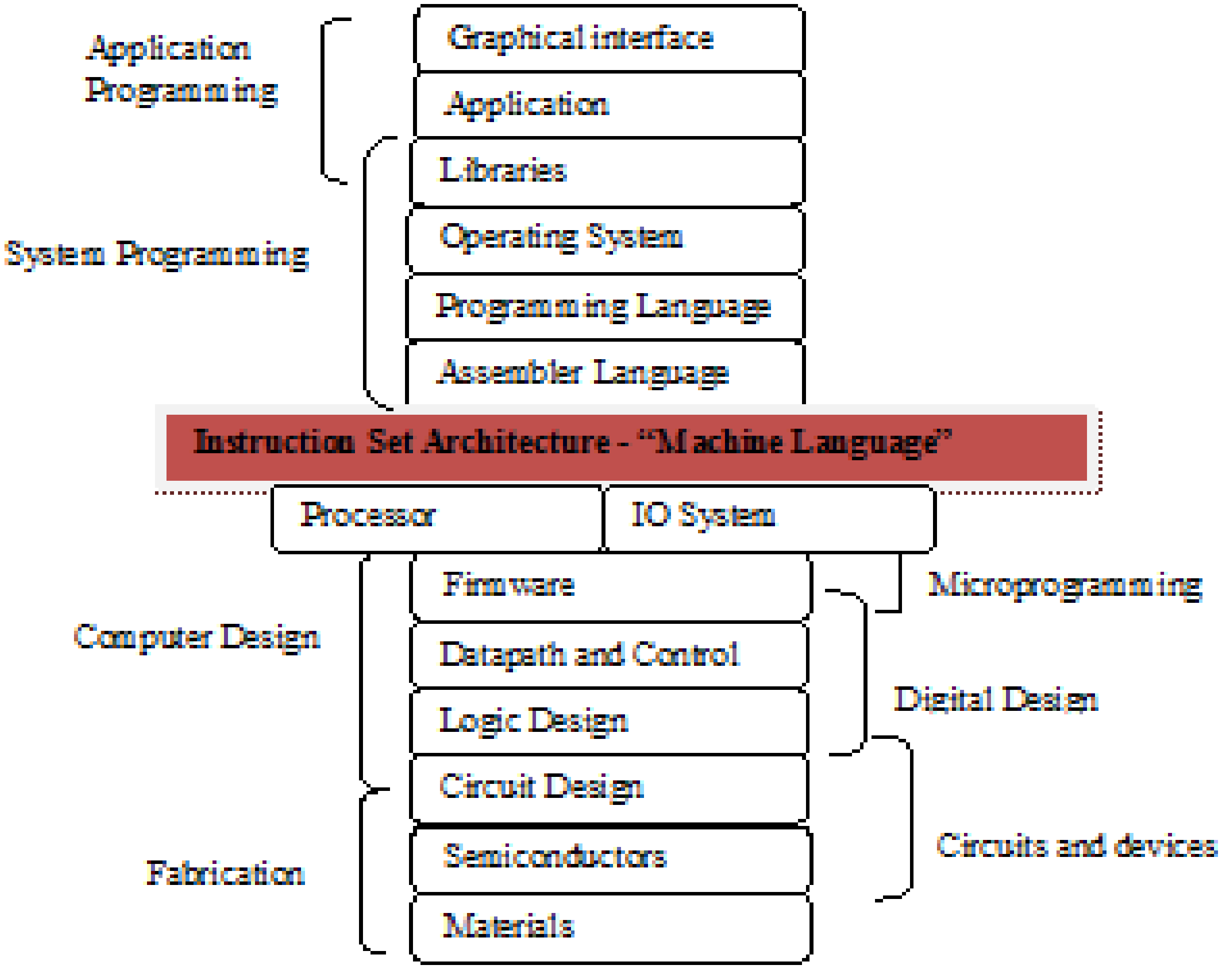
Applications suggest how to improve technology, provide revenue to fund development



Improved technologies make new applications possible

Cost of software development makes compatibility a major force in market

Structure of Computer Architecture



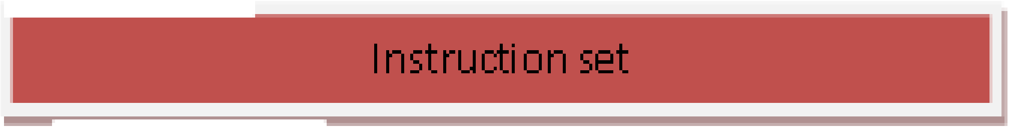
The Instruction Set: A Critical Interface

Computer Architecture =
Instruction Set Architecture +
Machine Organization

Instruction Set Design
- Machine Language
- Compiler View
- "Computer Architecture"
- "Instruction Set Architecture"

"Building Architect"

Software



Instruction set

Hardware

Computer Organization and Design
- Machine Implementation
- Logic Designer's View
- "Processor Architecture"
- "Computer Organization"

"Construction Engineer

Instruction Set Architecture

- Interface between hardware and software
- the attributes of a [computing] system as seen by the programmer, i.e. the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation. it includes,
- **Data Types:** Encoding and representation
- **Memory Model:** Organisation of storage
- **Program Visible Processor State :** General registers, Program counter, Processor status
- **Instruction Set:** Instructions and formats, Instruction set or operation code (opcode) e.g SISC, CISC, RISC, Addressing modes, Data structures
- **System Model:** States, Program exemptions or conditions, Privileges, Interrupts, IO
- **External Interfaces:** IO ,Management

Computer Organization

- Capabilities & Performance Characteristics of Principal Functional Units

  (e.g., Registers, ALU, Shifters, Memory Management, etc.
- Ways in which these components are interconnected

  Datapath - nature of information flows and connection of functional units
  - Control - logic and means by which such information flow is controlled
  - Choreography of functional units to realize the ISA
- Register Transfer Level Description / Microcode

SISC

- First set of computers
- algorithms and hardware for complicated tasks had not yet been developed
- Small and simple instruction
- Simple operations

CISC

- Bigger is better!
- Make the hardware as "smart“
- If you have a great sprawling architecture, that's ok.
- Don't worry about whether or not the system design is neatly partitioned into layers.
- When one little part fails, the whole system dies and we will never find out why.
- **Features: Memory access is directly available to most types of instruction.**
- **Addressing mode are substantial in number.**
- **Instruction formats are of different lengths.**
- **Instructions perform both elementary and complex operations**
- Problems: *lack of maintainability*, *lack of verifiability*, and *brittleness*.
- As CISC machines got more and more complex, they failed considerably more frequently.
- brittle (non-robust) performance in practical computing problems.

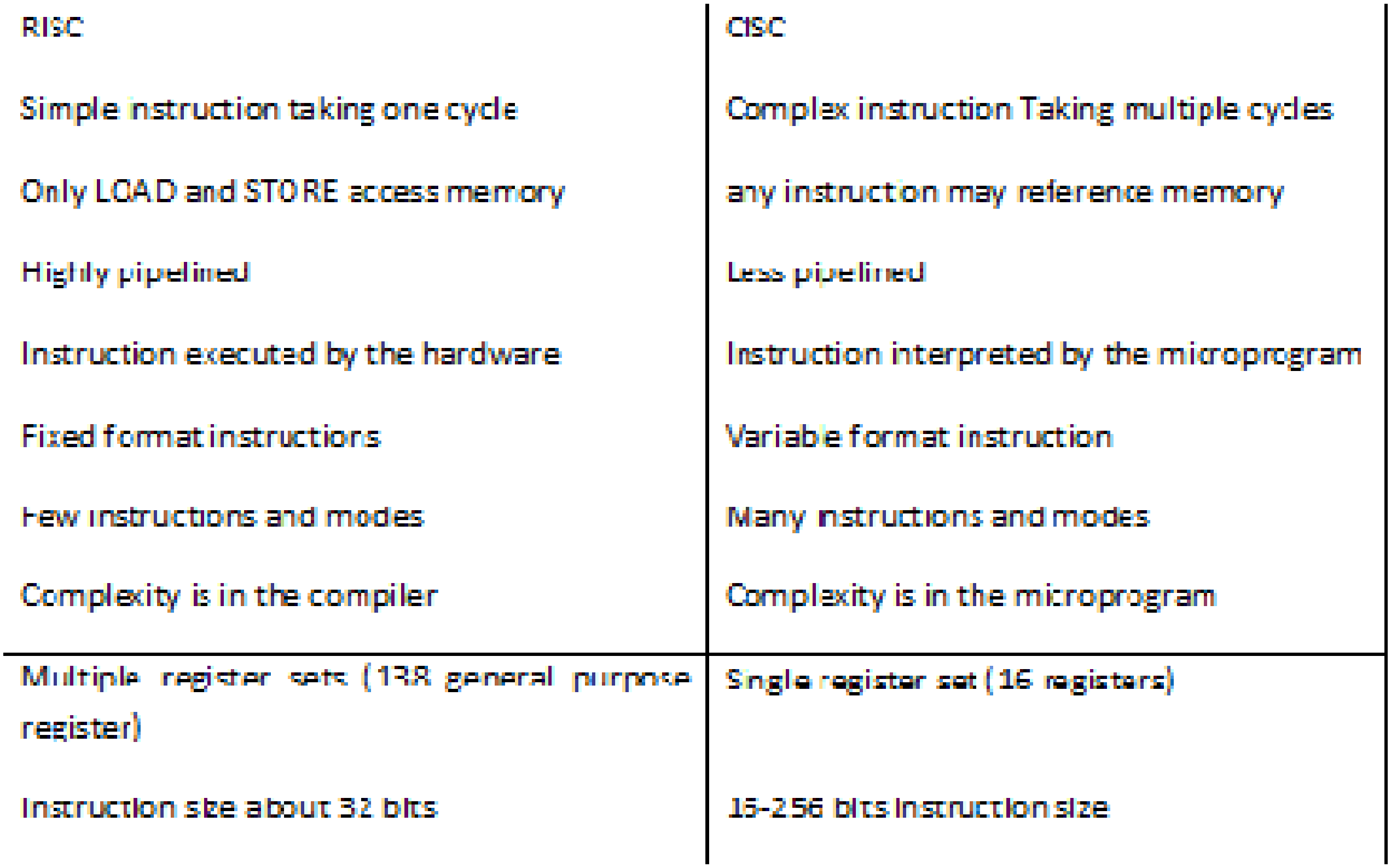
RISC

- Small is beautiful.
- Keep the hardware simple and stupid (KISS design philosophy).
- Hardware and software designers should work together to make the architecture simple and modular.
- Neatly partition the system design into layers,
- Then, take the vast majority of the functionality that CISC implements in hardware, and put it into software (using compiler transformations) instead.
- Make the hardware and compiler robust, so the entire system can perform reliably.
- Strength:
- By keeping the hardware small and modular, the design and fabrication, maintenance, and debugging costs are reduced.
- Economical
- potentially greater profits by shortening both the development and product life cycles.
- greater robustness through modularity and simplicity in hardware and software
- verifiability using software-based testing and proof-of-correctness.
- good economic sense, because development and re-work costs are significantly reduced

RISC cntd

- Memory accesses are restricted to load and store instruction, and data manipulation instructions are register to register.
- –Addressing modes are limited in number.
- –Instruction formats are all of the same length.
- –Instructions perform elementary operations

Comparison of RISC and CISC



RISC	CISC
Simple instruction taking one cycle	Complex instruction Taking multiple cycles
Only LOAD and STORE access memory	any instruction may reference memory
Highly pipelined	Less pipelined
Instruction executed by the hardware	Instruction interpreted by the microprogram
Fixed format instructions	Variable format instruction
Few instructions and modes	Many instructions and modes
Complexity is in the compiler	Complexity is in the microprogram
Multiple register sets (138 general purpose register)	Single register set (16 registers)
Instruction size about 32 bits	16-256 bits instruction size

PARALLELISM

- The architectural technique that allows an overlap of individual machine operations. Multiple operations will execute simultaneously,
- Goal: speed up the execution.
- A parallel computer (or multiple processor system) is a collection of communicating processing elements (processors) that cooperate to solve large computational problems fast by dividing such problems into parallel tasks

Issues

- The concurrency and communication characteristics
- Computing Resources and Computation Allocation :
  processing elements, computing power,
  amount/organization of memory
  - portions of the computation and data are allocated or mapped to each PE.
- Data access, Communication and Synchronization
- • How the processing elements cooperate and communicate.
- • How data is shared/transmitted between processors.
- • Abstractions and primitives for cooperation/ communication and synchronization.
- • The characteristics and performance of parallel system network (System interconnects).

Goals

- **Maximize performance: Maximize Speedup.**
- **– By minimizing parallelization overheads and balancing workload on processors**
- **Scalability of performance to larger systems/problems.**

Need And Feasibility

- Application demands: More computing cycles/memory needed
- Technology Trends:
- Architecture Trends:
- Economics

Challenging Applications in Applied Science/Engineering

- Astrophysics
- • Atmospheric and Ocean Modeling
- • Bioinformatics
- • Biomolecular simulation: Protein folding
- • Computational Chemistry
- • Computational Fluid Dynamics (CFD)
- • Computational Physics
- • Computer vision and image understanding
- • Data Mining and Data-intensive Computing
- • Engineering analysis (CAD/CAM)
- • Global climate modeling and forecasting
- • Material Sciences
- • Military applications
- • Quantum chemistry
- • VLSI design
- • ….

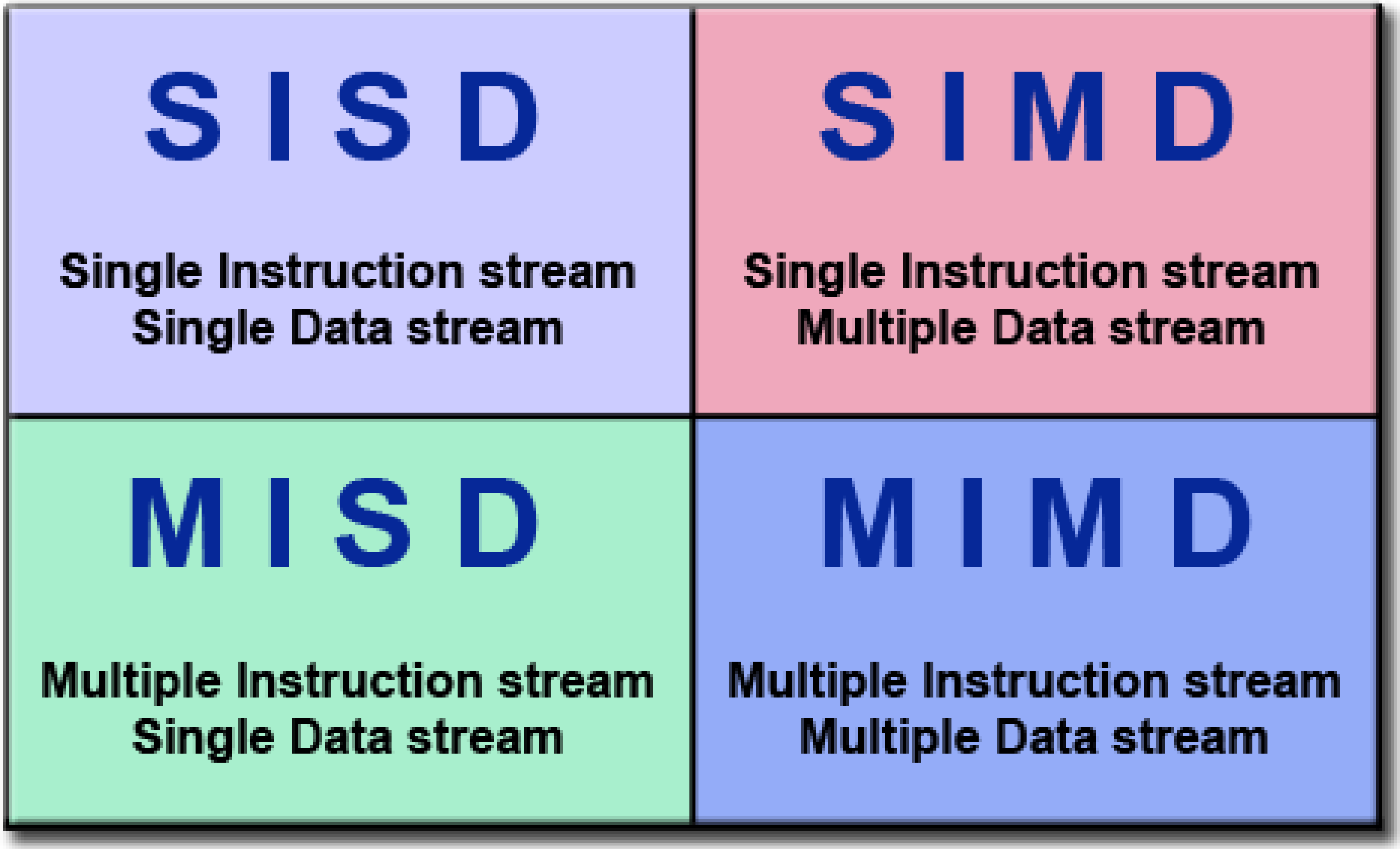
Elements of Parallel Computing

- Computing Problems:
- Parallel Algorithms and Data Structures
- Hardware Resources
- Operating Systems
- Parallelism exploitation possible at:
- Compiler Support
- System Software Support

CLASSIFICATION

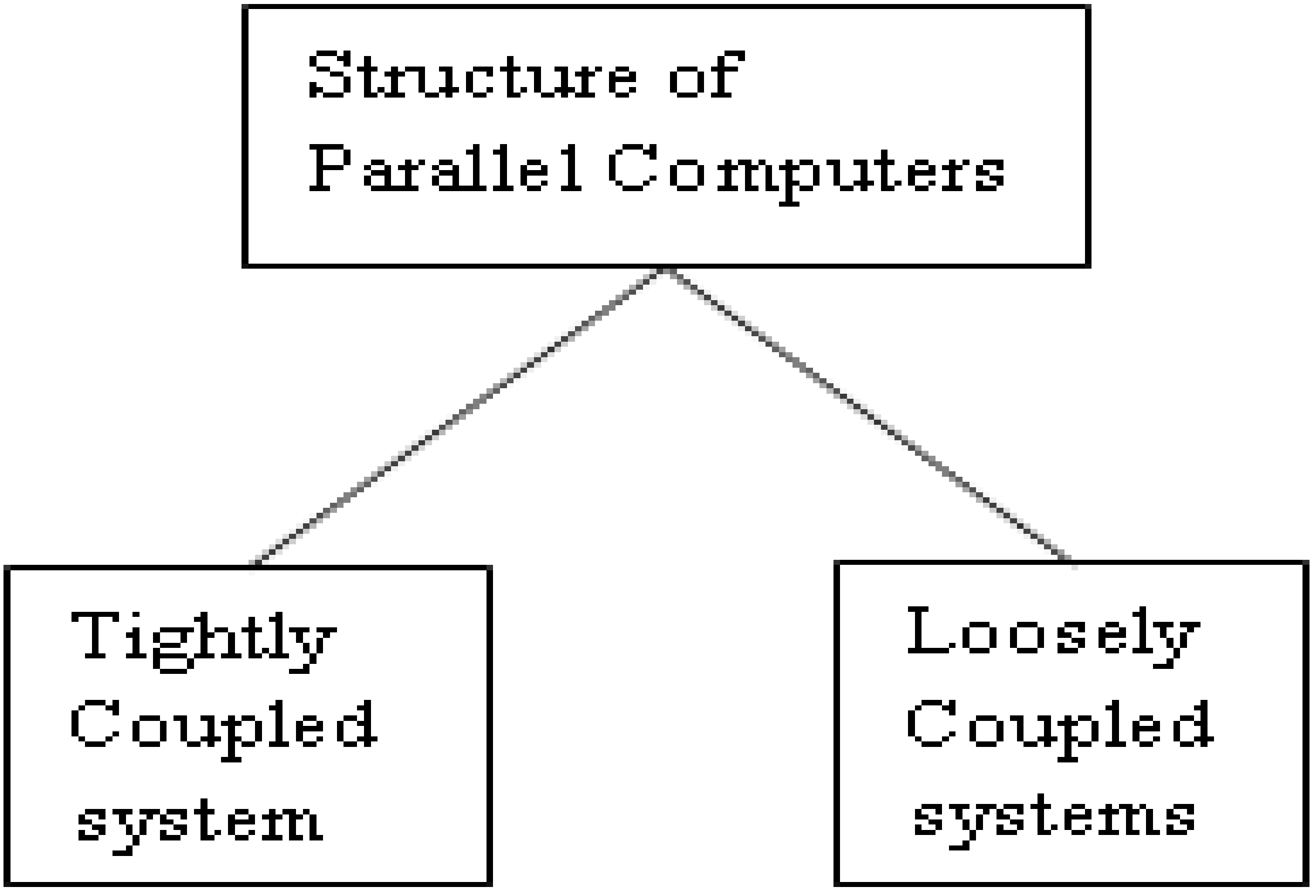
- 1) Classification based on the instruction and data streams : SISD, SIMD, MISD, MIMD
- 2) Classification based on the structure of computers (multiprocessor)
- 3) Classification based on how the memory is accessed
- 4) Classification based on grain size

Flynn’s Taxonomy



SISD Single Instruction stream Single Data stream	SIMD Single Instruction stream Multiple Data stream
MISD Multiple Instruction stream Single Data stream	MIMD Multiple Instruction stream Multiple Data stream

STRUCTURAL CLASSIFICATION

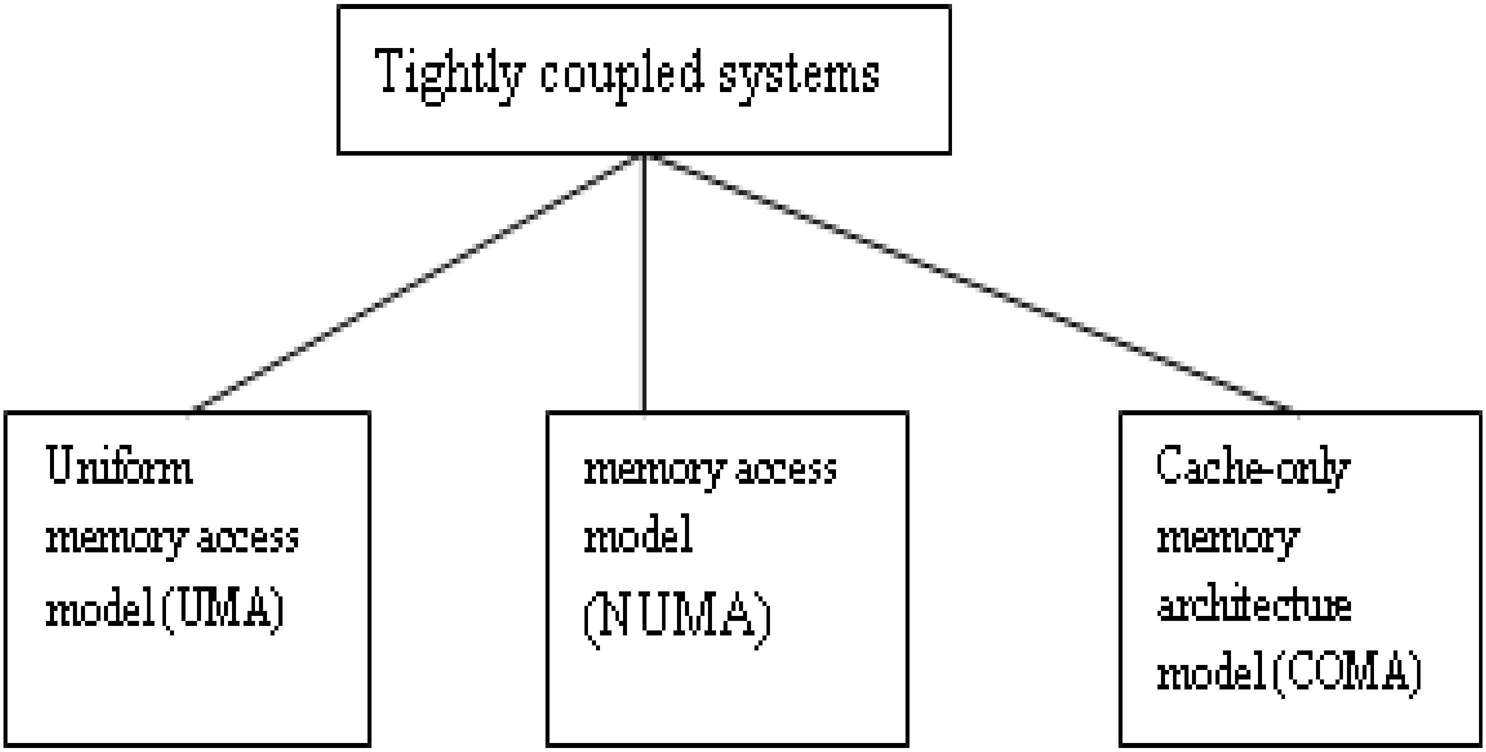


Structure of Parallel Computers

Tightly Coupled system

Loosely Coupled systems

Memory Access



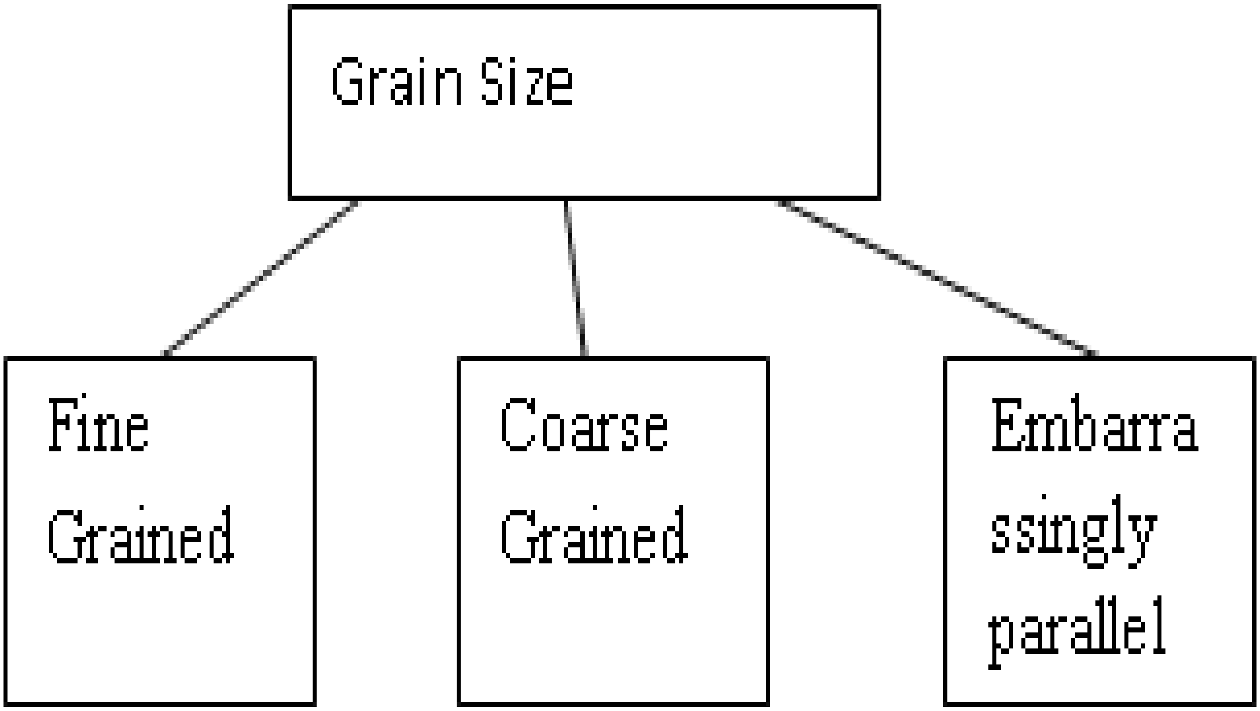
Tightly coupled systems

Uniform memory access model(UMA)

memory access model (NUMA)

Cache-only memory architecture model(COMA)

Grain Size


Grain Size
Fine Grained
Coarse Grained
Embarrassingly parallel

Types of Parallelism

- Bit level parallelism: increasing word size
- Instruction level parallelism: multiple stages
- Data parallelism: loops, similar operation os same/different data sets
- Task parallelism: different operation on same or different data sets

Instruction level parallelism

IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB

Multiprocessing

- coordinated processing of programs by more than one processor
- Classes
- Symmetric (SMP) or tightly coupled:
  - share memory, I/O bus or data path
  - one OS
  - aka "shared everything" system ,<= 16 processors
  - system or user tasks run on any processor, is flexible, better performance

  Massively parallel or loosely coupled
  - > 200 processors ,
  - Each processor has OS, memory,
  - MPP system is known as "shared nothing system".

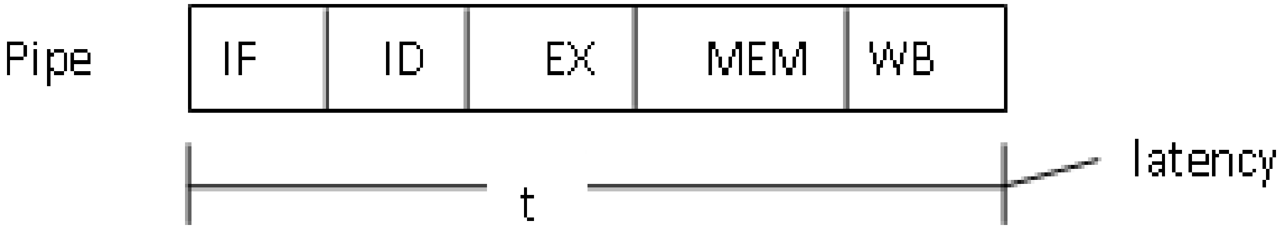
Requirements

- Motherboard support
- Processor support
- Operating system support
- Protocol support

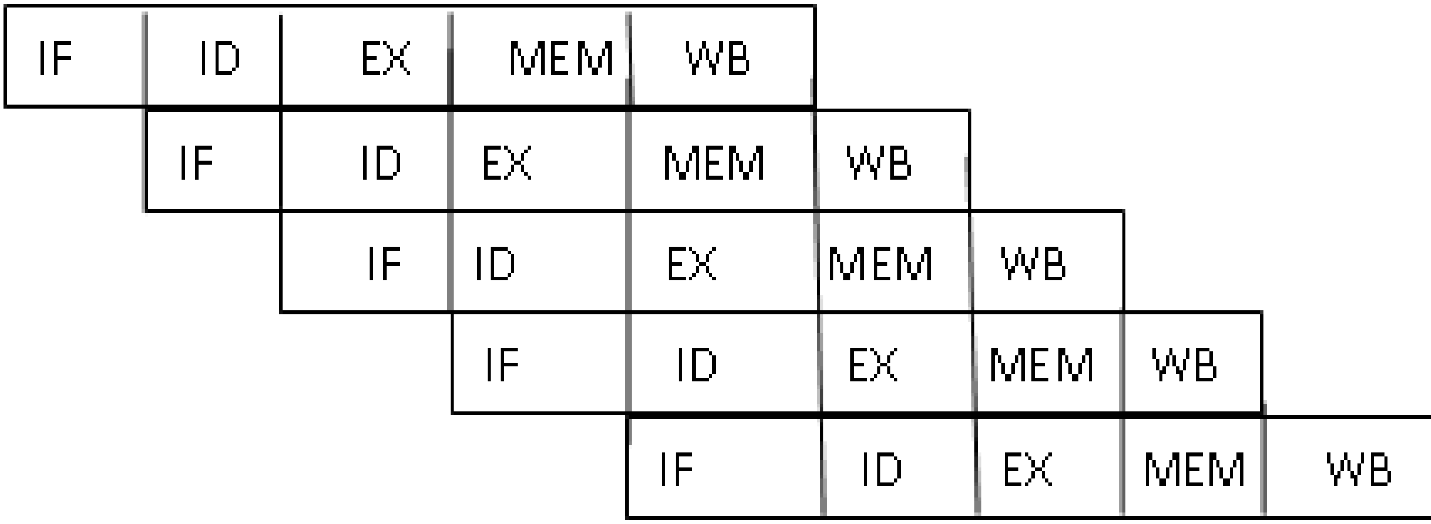
Pipelining

- pipeline is a set of data processing elements connected in series, so that the output of one element is the input of the next
- Computer Related Pipelines
- Instruction pipelines
- Graphic pipelines
- Software pipelines

Pipelining



The classic 5-Stage RISC Pipeline

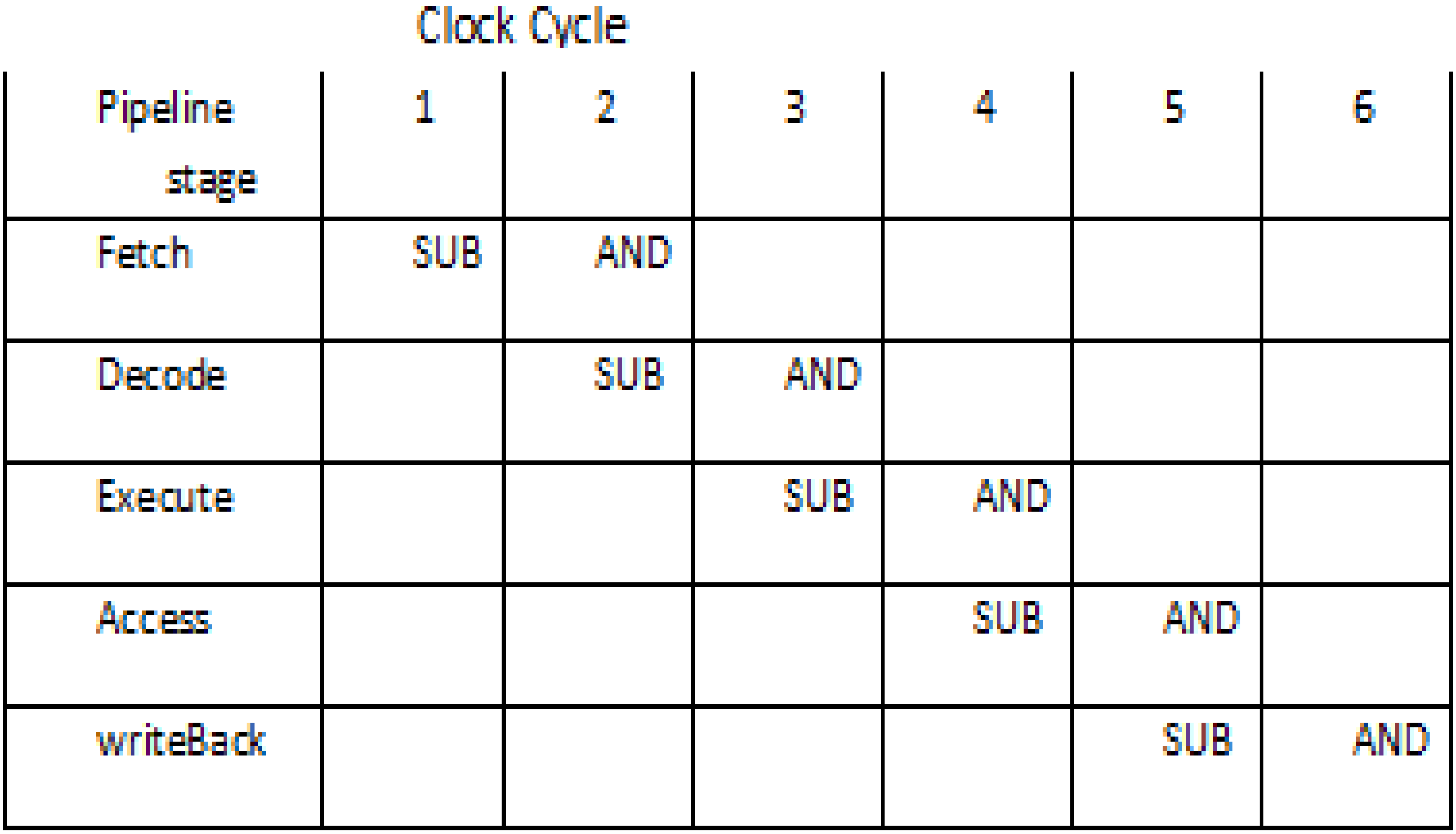


IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB

Hazards

- pipeline would produce wrong answers.
- Structural hazards: two instructions attempt to use the same resources
- Control hazards: delay between the fetching of instruction and decision about changes in control flow (branches and jumps)
- Data hazards: instruction scheduled blindly attempt to use data before the data is available in the register file.

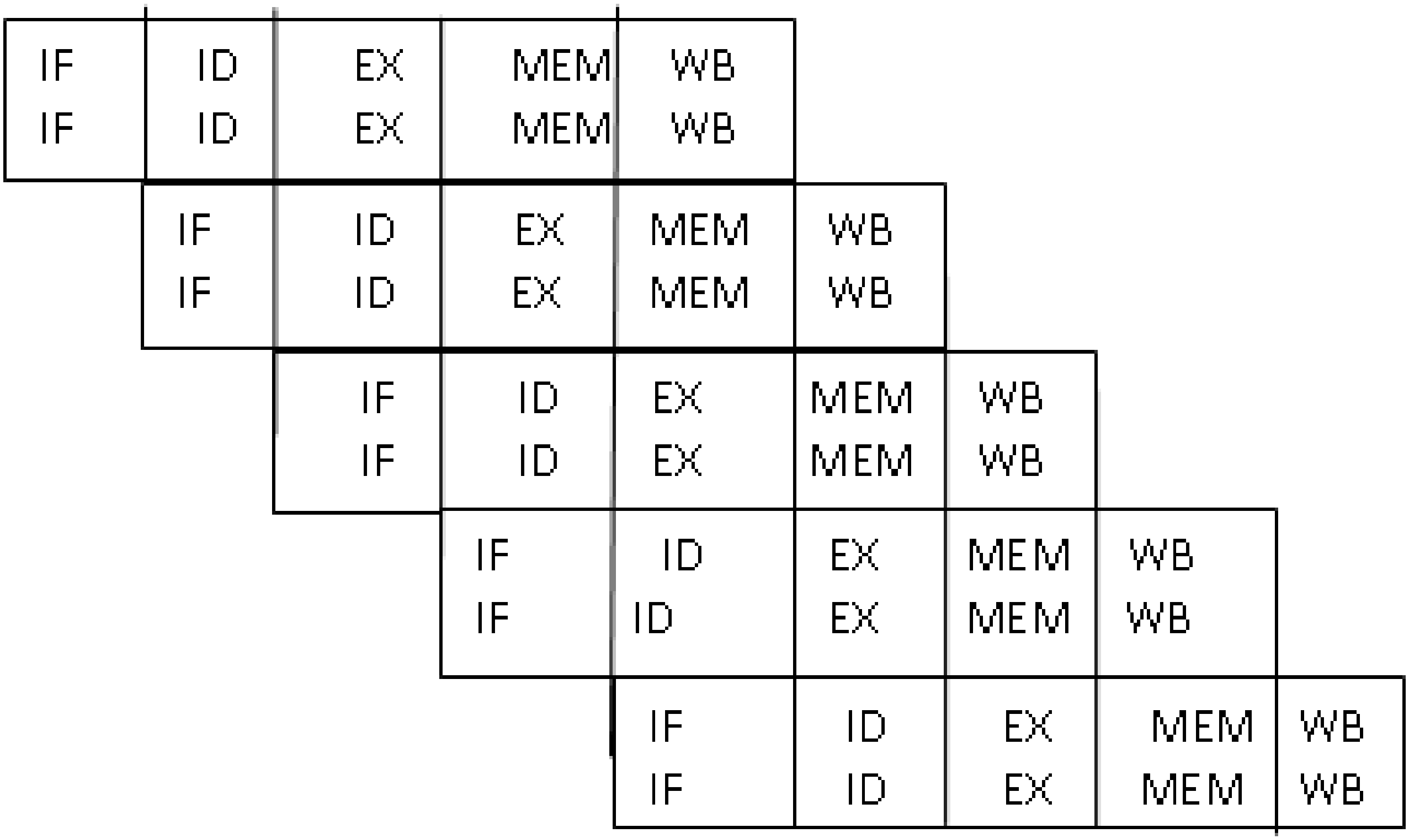
Data hazard



Clock Cycle

Pipeline stage	1	2	3	4	5	6
Fetch	SUB	AND				
Decode		SUB	AND			
Execute			SUB	AND		
Access				SUB	AND	
writeBack					SUB	AND

super scalar processor

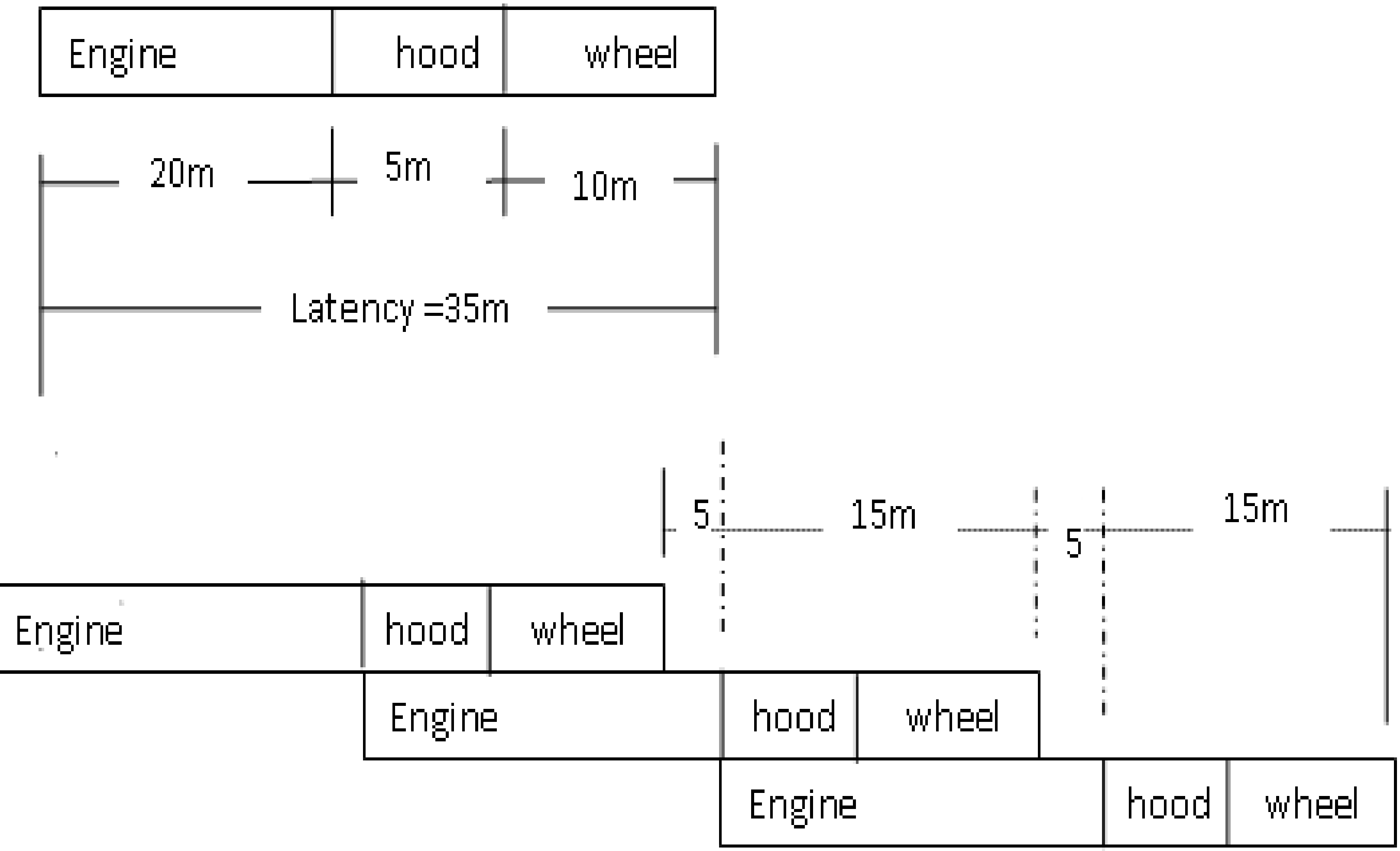


IF IF	ID ID	EX EX	MEM MEM	WB WB				
	IF IF	ID ID	EX EX	MEM MEM	WB WB			
		IF IF	ID ID	EX EX	MEM MEM	WB WB		
			IF IF	ID ID	EX EX	MEM MEM	WB WB	
				IF IF	ID ID	EX EX	MEM MEM	WB WB

Cost drawbacks and benefits

- does not reduce the time
- It only increases the throughput processing a stream of data.
- High pipeline leads to increase of latency
- Pipelined systems require more resources than batch, its stages cannot reuse resources Moreover, pipelining may increase the time it takes for an instruction to finish.

- Assembly of a car involves installing the engine, the hood and the wheels. Assume installation of each part takes place as follows: engine 20 mins, hood 5 mins and wheels 10 mins, using pipeline determine how long it takes to assemble 3 cars.


Engine
hood
wheel
20m
5m
10m
Latency =35m
5
15m
5
15m
Engine
hood
wheel
Engine
hood
wheel
Engine
hood
wheel

Superscalar Architecture

- Superscalar is a computer designed to improve the performance of the execution of **scalar instructions.**
- A scalar is a variable that can hold only one atomic value at a time, e.g., an integer or a real.
- A scalar architecture processes one data item at a time

Superscalar Architecture cont..

- In a **superscalar architecture (SSA),** several scalar instructions can be initiated simultaneously and execute independently. Pipelining allows also several instructions to be executed
- at the same time, but they have to be in different pipeline
- stages at any given moment.

- SSA includes all features of pipelining but, in addition, there can be several instructions executing simultaneously in the same pipeline stage.
- SSA introduces therefore a new level of parallelism, called **instruction-level parallelism.**

Implementation

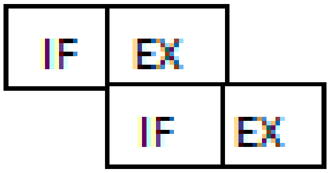
- A SSA processor fetches multiple instructions at a time, and attempts to find nearby instructions that are independent of each other and therefore can be executed in parallel.
- Based on the dependency analysis, the processor may issue and execute instructions in an order that differs from that of the original machine code.
- The processor may eliminate some unnecessary dependencies by the use of additional registers and renaming of register references.

Super-pipelining

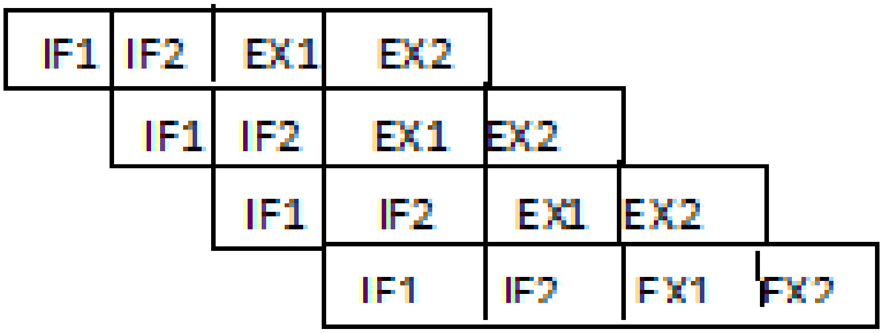
- Super-pipelining is based on dividing the stages of a pipeline into several sub-stages, and thus increasing the number of instructions which are handled by the pipeline at the same time.
- For example, by dividing each stage into two sub-stages, a pipeline can perform at twice the speed in the ideal situation.
- Many pipeline stages may perform tasks that require less than half a clock cycle.
- No duplication of hardware is needed for these stages.

Super-pipelining

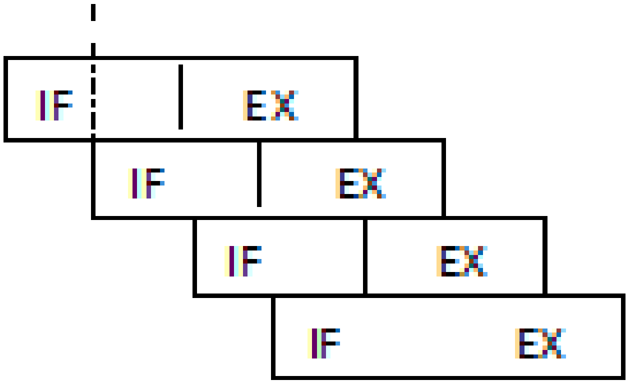
- For a given architecture and the corresponding instruction set there is an optimal number of pipeline stages/sub-stages.
- Increasing the number of stages/sub-stages over this limit
- reduces the overall performance.
  - Not all stages can be divided into (equal-length) sub-stages.
  - Overhead of data buffering between the stages.
  - The hazards will be more difficult to resolved.
  - More complex hardware.
  - Interrupt handling and testing will be more complicated.



2 h/w resources needed



4 h/w resources needed



3 h/w resources needed

Superscalar vs. Superpipeline

Base machine: *4-stage* pipeline

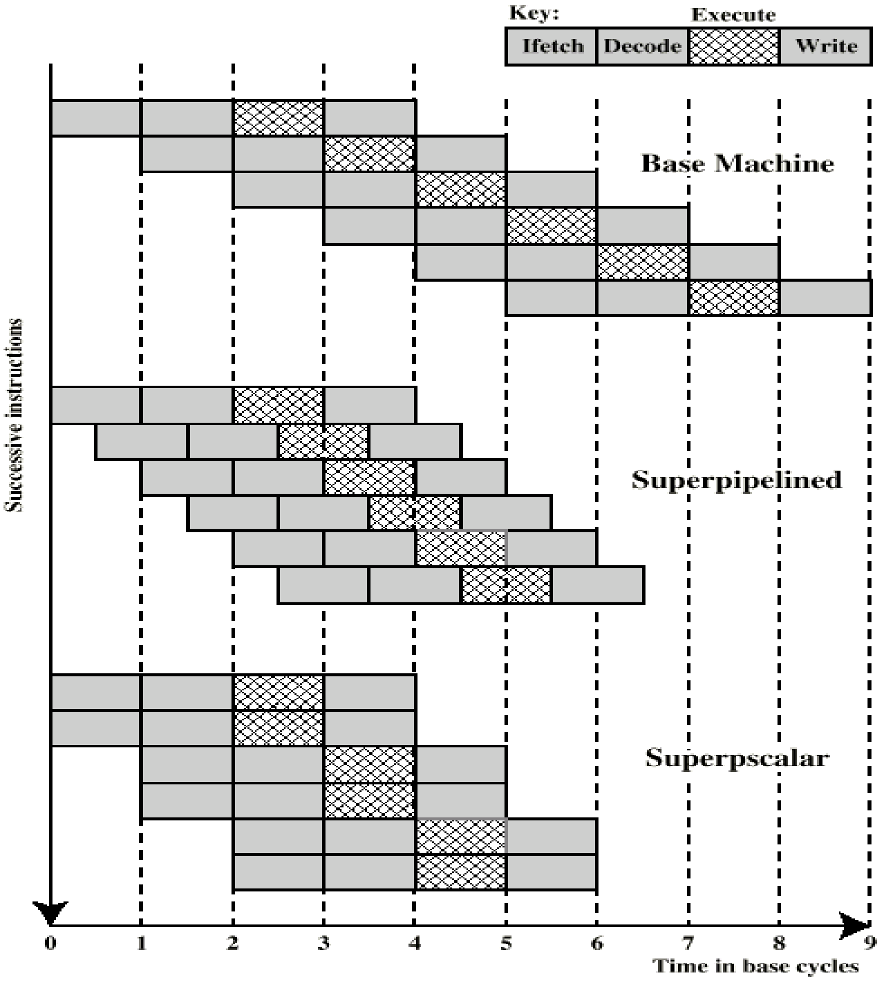
- Instruction fetch
- Operation decode
- Operation execution
- Result write back

Superpipeline of degree *2*

- A sub-stage may take half a clock cycle to finish.

Superscalar of degree *2*

- Two instructions are executed concurrently in each pipeline stage.
- Duplication of hardware is required by definition.



Superscalar Design

- SSA allows several instructions to be issued and completed per clock cycle.
- It consists of a number of pipelines that are working in parallel.

Depending on the number and kind of parallel units available, a certain number of instructions can be executed in parallel.

Parallel Execution Limitation

- The situations which prevent instructions to be executed in parallel by SSA are mainly due to:
  - Resource conflicts.
  - Control (procedural) dependency.
  - Data dependencies.
- They are very similar to those which prevent efficient execution on a pipelined architecture (pipeline hazards):
- Their consequences on SSA are more severe than those on simple pipelines, because the potential of parallelism in SSA is greater and, thus, a larger amount of performance will be lost.

Resource Conflicts

- Several instructions compete for the same hardware
- resource at the same time.
  - e.g., two arithmetic instructions need the same floatingpoint

    unit for execution.

  similar to structural hazards in pipeline.

- They can be solved partly by introducing several hardware units for the same functions.
  - e.g., have two floating-point units.
  - the hardware units can also be pipelined to support several operations at the same time.
  - however, memory units can’t be duplicated.

Procedural Dependency

- The presence of branches creates major problems in assuring the optimal parallelism.
  - cannot execute instructions after a branch in parallel with instructions before a branch.
  - similar to control hazards in pipeline.

- If instructions are of variable length, they cannot be fetched and issued in parallel, since an instruction has to be decoded in order to identify the following one.

  therefore, superscalar techniques are more efficiently applicable to RISCs, with fixed instruction length and format.

Data Conflicts

- Caused by data dependencies between instructions in the program.
  - Similar to date hazards in pipeline.
  - But we have much more data dependencies now, due to the parallel execution of many instructions.
- To address the problem and to increase the degree of parallel execution, SSA provides a great liberty in the order in which instructions are issued and executed.

- Therefore, data dependencies have to be considered and dealt with much more carefully.

Window of Execution

- Due to data dependencies, only some part of the instructions are potential subjects for parallel execution.

- In order to find instructions to be issued in parallel, the processor
- has to select from a sufficiently large instruction sequence.
  - There are usually a lot of data dependencies in a short instruction sequence.

- **Window of execution is defined as the set of instructions that is** considered for execution at a certain moment.

- The number of instructions in the window should be as large as possible. However, it is limited by:
  - Capacity to fetch instructions at a high rate.
  - The problem of branches.
  - The cost of hardware needed to analyze data dependencies.

Instruction vs Machine Parallelism

- Instruction-level parallelism (ILP) the average number of instructions in a program that a processor might be able to execute at the same time.

  Determined by the number of true dependencies and procedural (control) dependencies in relation to the number of other instructions.

- Machine parallelism of a processor the ability of the processor to take advantage of the ILP of the program.
  - Determined by the number of instructions that can be fetched and executed at the same time, i.e., the capacity of the hardware.
- To achieve high performance, we need both ILP and machine parallelism.
  - Ideally, we have the same ILP and machine parallelism. However, this is impossible, since the same computer is used for programs with different ILPs.

Memory Hierarchy

Memory Hierarchy

Memory System

- Every computer system contains various devices for storing data and instructions
- Memory System: Storage devices + algorithms (HW or SW implementation) needed to control or manage the stored information.
- Processors should have instant and uninterrupted access to memory
- Maximum speed of information transfer between memory and processor
- Memories operating at the speed comparable to processor’s speed are relatively costly
  - It is not visible to employ a single memory using one type of technology
  - the stored information is distributed over variety of different memory units with very different physical characteristics

Memory Design

- Goal
- to provide:
  - Adequate storage capacity
  - Acceptable level of performance
  - A reasonable cost

  The use of a number of different memory devices with different cost performance ratios organized to provide a high average performance at a low average cost per bit

  the individual memories form a hierarchy of storage devices

Components of the memory

- Internal processor memory
  - A small set of high-speed registers used for storage of instruction and data
- Main memory (primary memory)
  - A relatively large and fast memory used for program and data storage during computer operation
  - Locations in main memory can be accessed directly and rapidly by the CPU instruction set
  - Semiconductor technology
- Secondary memory (auxiliary memory)
  - Large memory capacity and slower than mm
  - Store system programs, large data files etc which are not continually required by the CPU
  - Information in secondary storage is accessed indirectly via I/O programs
  - Representative technologies used for secondary memory are magnetic and optic disks

Motivation for Memory Hierarchy

- Faster storage technologies are more costly
- o Cost more money per byte
- o Have lower storage capacity
- o Require more power and generate more heat
- • The gap between processing and memory is widening
- o Processors have been getting faster and faster
- o Main memory speed is not improving as dramatically
- • Well-written programs tend to exhibit good locality
- o Across time: repeatedly referencing the same variables
- o Across space: often accessing other variables located nearby
- Want the *speed of fast storage at the cost and capacity of*
- slow storage. Key idea: memory hierarchy!

Why Hierarchical Memory System?

- Economics and performance
- Main memory has less storage capacity, more expensive and slow (i.e. access time required to fetch data or instruction)
- Cache memory is the most expensive, smallest and fastest memory with least access time.

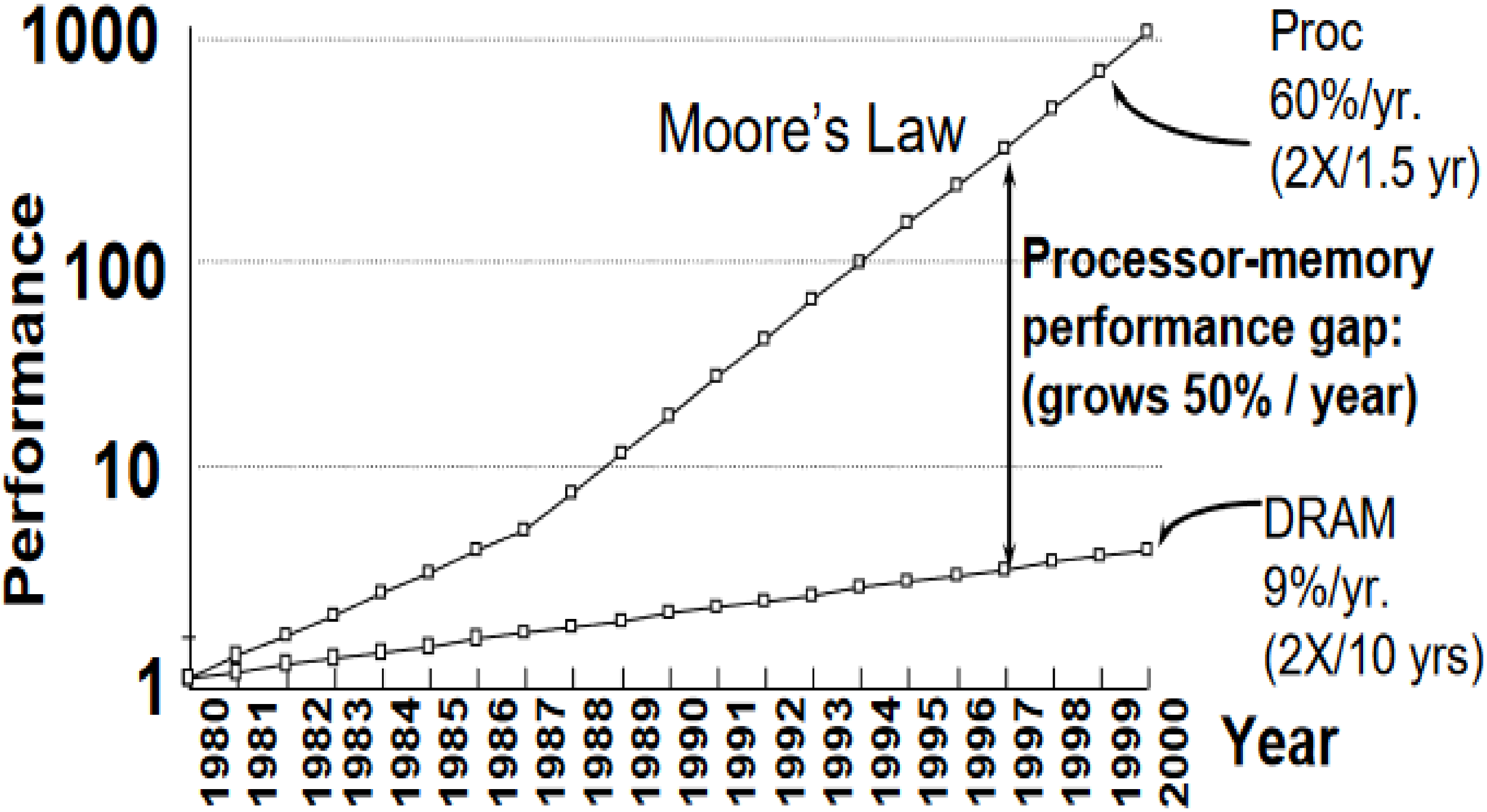
Simple Three-Level Hierarchy

- Registers
- o Usually reside directly on the processor chip
- o Essentially no latency, referenced directly in instructions
- o Low capacity (e.g., 32-512 bytes)
- Main memory
- o Around 100 times slower than a clock cycle
- o Constant access time for any memory location
- o Modest capacity (e.g., 512 MB-8GB)
- Disk
- o Around 100,000 times slower than main memory
- o Faster when accessing many bytes in a row
- o High capacity (e.g., 200 GB - TeraBytes)

Widening Processor/Memory Gap

- Gap in speed increasing from 1986 to 2000
- o CPU speed improved ~55% per year
- o Main memory speed improved only ~10% per year
- Main memory as major performance bottleneck
- o Many programs stall waiting for reads and writes to finish
- Changes in the memory hierarchy
- o Increasing the number of registers
- – 8 integer registers in the x86 vs. 128 in the Itanium
- o Adding caches between registers and main memory
- – On-chip level-1 cache and off-chip level-2 cache

Widening Processor/Memory Gap

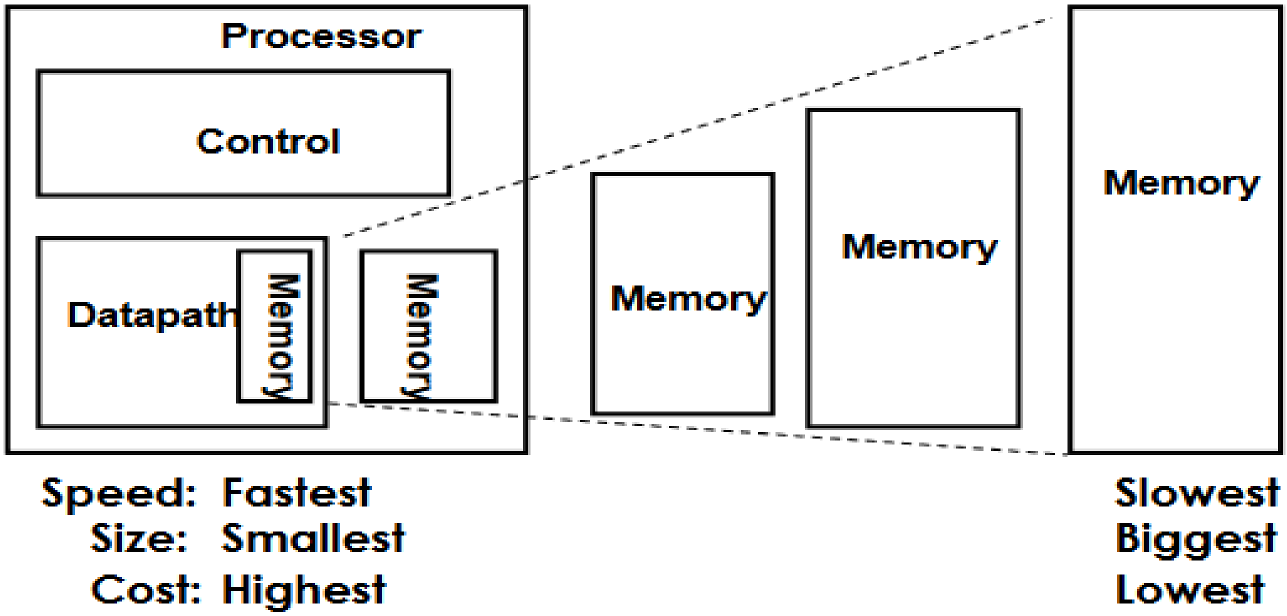


Solution: Memory Hierarchy

An Illusion of a large, fast, cheap memory

- Fact: Large memories slow, fast memories small
- How to achieve: hierarchy, parallelism

An expanded view of memory system:



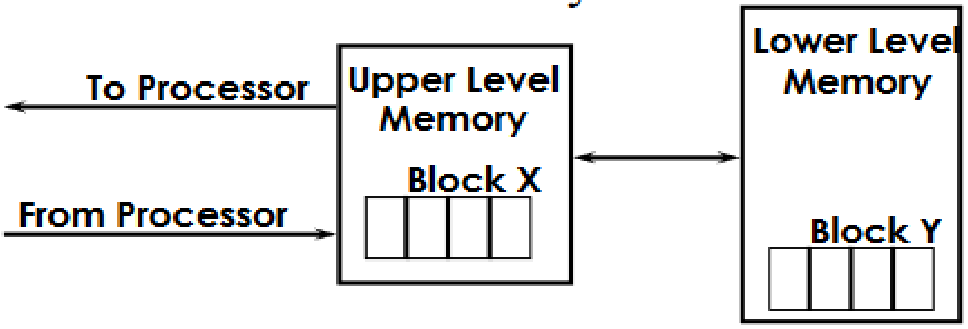
Memory Hierarchy: Principles

At any given time, data is copied between only two adjacent levels:

- Upper level: the one closer to the processor
  - Smaller, faster, uses more expensive technology
- Lower level: the one away from the processor
  - Bigger, slower, uses less expensive technology

Block: basic unit of information transfer

- Minimum unit of information that can either be present or not present in a level of the hierarchy



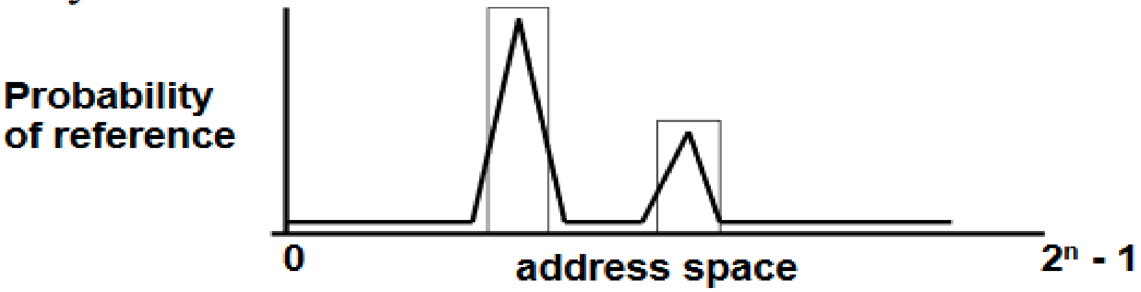
Why Hierarchy works?

Principle of Locality:

- Program access a relatively small portion of the address space at any instant of time
- 90/10 rule: 10% of code executed 90% of time

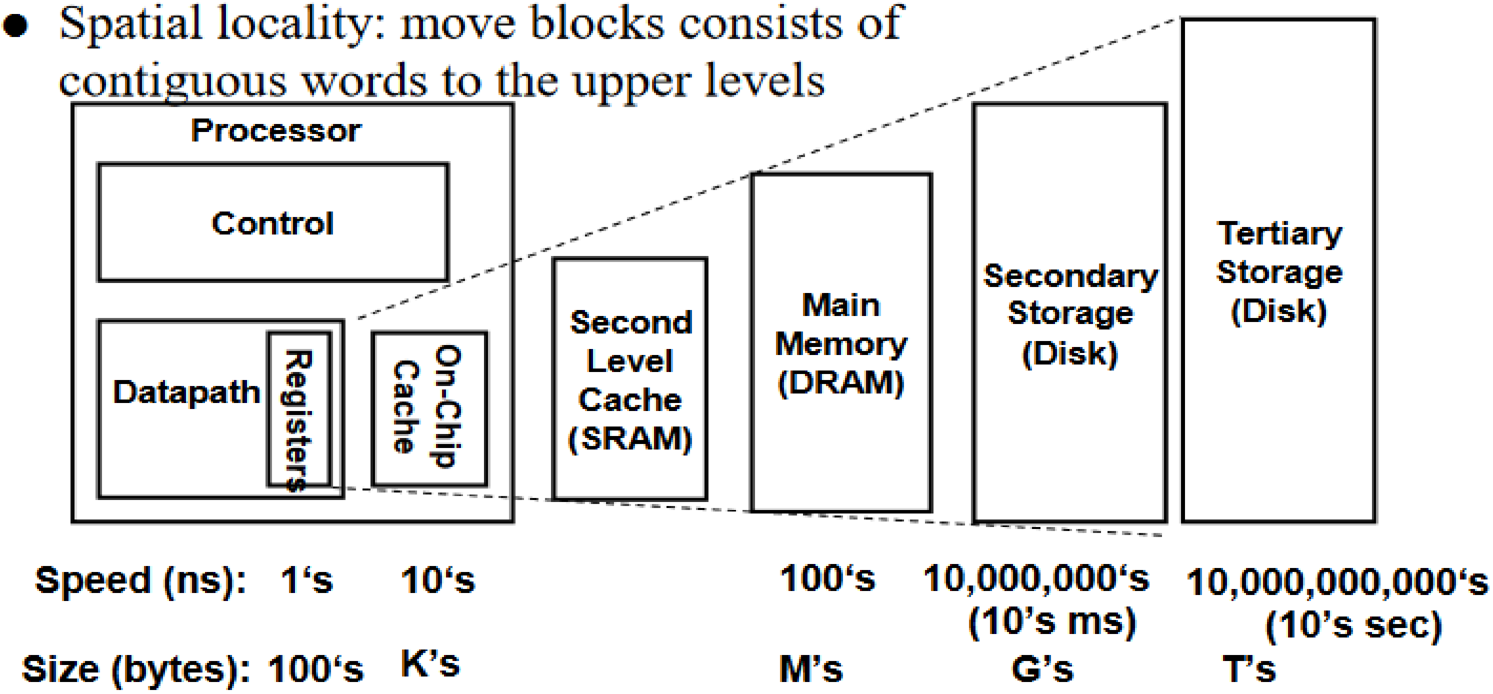
Two types of locality:

- Temporal locality: if an item is referenced, it will tend to be referenced again soon
- Spatial locality: if an item is referenced, items whose addresses are close by tend to be referenced soon

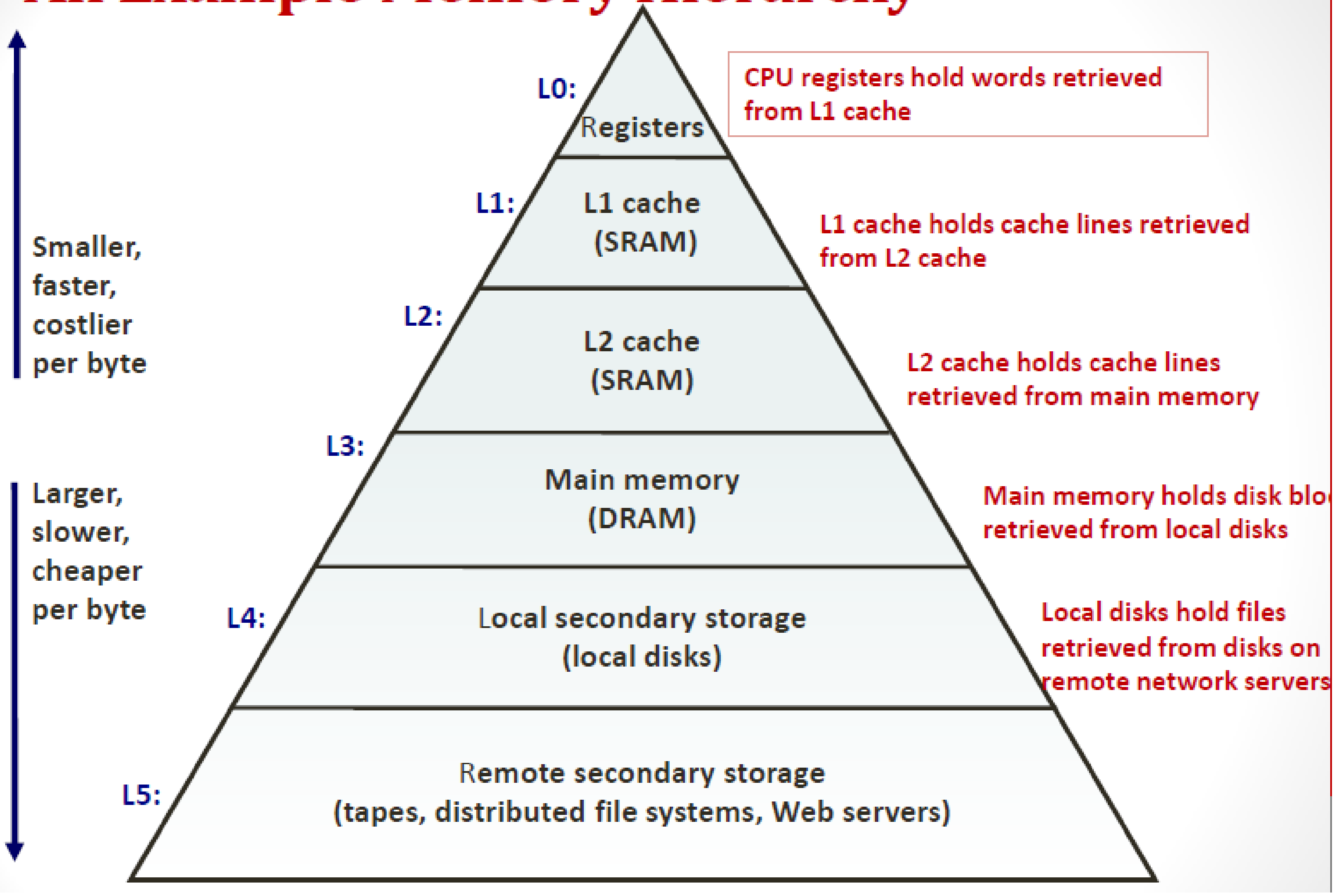


How Hierarchy Workss?

- Temporal locality: keep most recently accessed data items closer to the processor
- Spatial locality: move blocks consists of contiguous words to the upper levels


Processor
Control
Datapath
Registers
On-Chip Cache
Second Level Cache (SRAM)
Main Memory (DRAM)
Secondary Storage (Disk)
Tertiary Storage (Disk)
Speed (ns): 1's 10's 100's 10,000,000's (10's ms) 10,000,000,000's (10's sec)
Size (bytes): 100's K's M's G's T's

An Example Memory Hierarchy



Smaller, faster, costlier per byte

Larger, slower, cheaper per byte

L0: Registers

L1: L1 cache (SRAM)

L2: L2 cache (SRAM)

L3: Main memory (DRAM)

L4: Local secondary storage (local disks)

L5: Remote secondary storage (tapes, distributed file systems, Web servers)

CPU registers hold words retrieved from L1 cache

L1 cache holds cache lines retrieved from L2 cache

L2 cache holds cache lines retrieved from main memory

Main memory holds disk blo
retrieved from local disks

Local disks hold files retrieved from disks on remote network servers

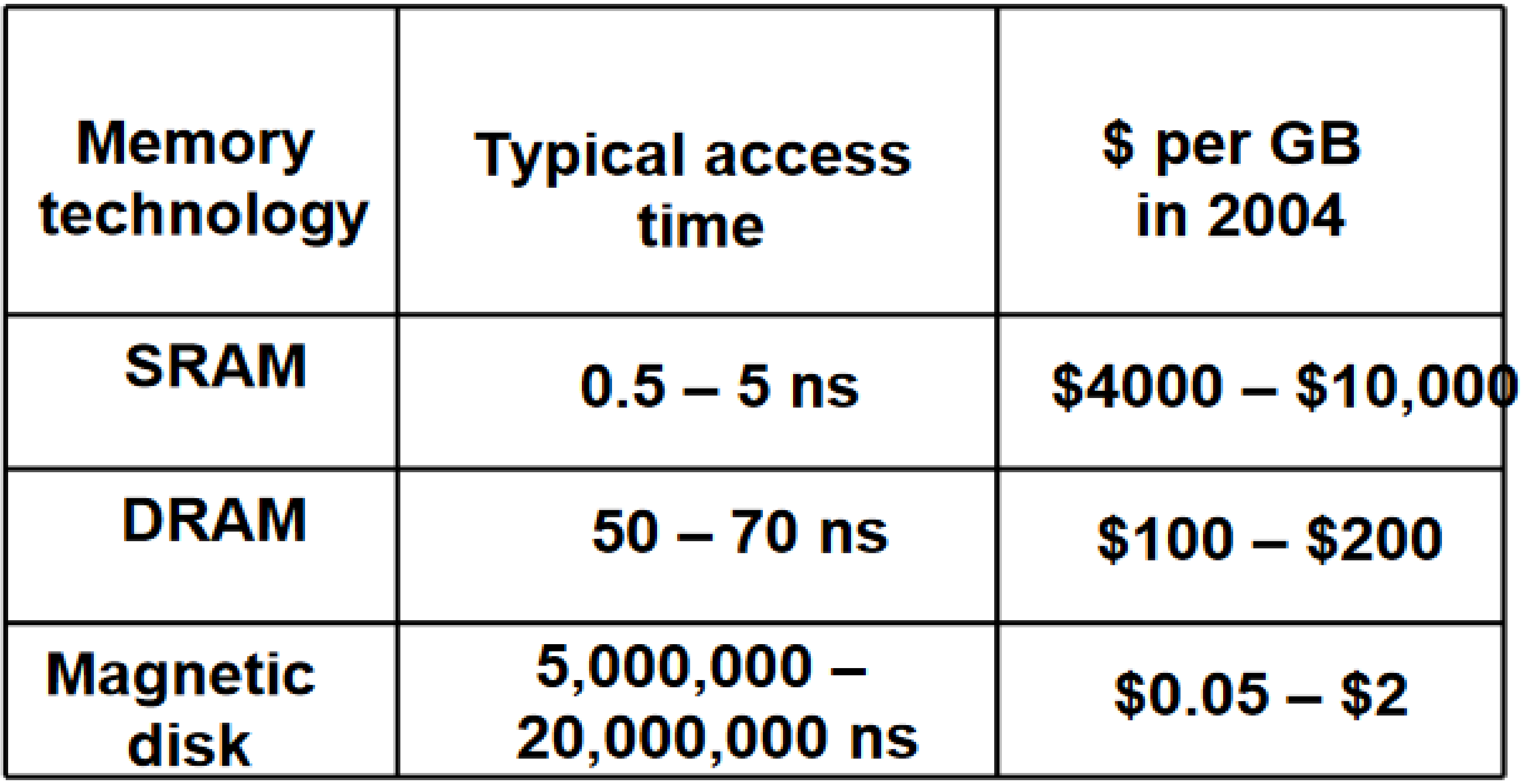
How is the Hierarchy Managed?

- Registers : managed by the compiler
- Cache : managed by the hardware
- Main memory: operating system
- Secondary memory: hardware and Operating System (virtual memory)
  - Programmer (files)

Memory Hierarchy Technology

- Random access:
- −Access time same for all locations
- −DRAM:Dynamic Random Access Memory
  - High density, low power, cheap, slow
  - Dynamic: need to be refreshed regularly
  - Addresses in 2 halves (memory as a 2D matrix):
    - −RAS/CAS (Row/Column Access Strobe)
  - Use for main memory
  - −SRAM:Static Random Access Memory
  - Low density, high power, expensive, fast
  - Static: content will last (forever until lose power)
  - Address not divided
  - Use for caches

Comparisons of Various Technologies



Memory technology	Typical access time	$ per GB in 2004
SRAM	0.5 – 5 ns	$4000 – $10,000
DRAM	50 – 70 ns	$100 – $200
Magnetic disk	5,000,000 – 20,000,000 ns	$0.05 – $2

Memory Hierarchy Technology

- Performance of main memory:
- −Latency: related directly to Cache Miss Penalty
  - Access Time: time between request and word arrives
  - Cycle Time: time between requests

  −Bandwidth: Large Block Miss Penalty (interleavedmemory, L2)

  Non-so-random access technology:

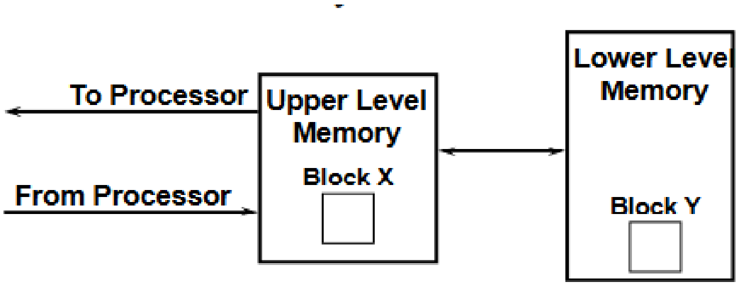
  −Access time varies from location to location and from time to time, e.g., disk, CDROM

  Sequential access technology: access time linear in location (e.g., tape)

Memory Hierarchy: Terminology

- Hit: data appears in upper level (Block X)
  - −Hit rate: fraction of memory access found in the upper level
  - −Hit time: time to access the upper level

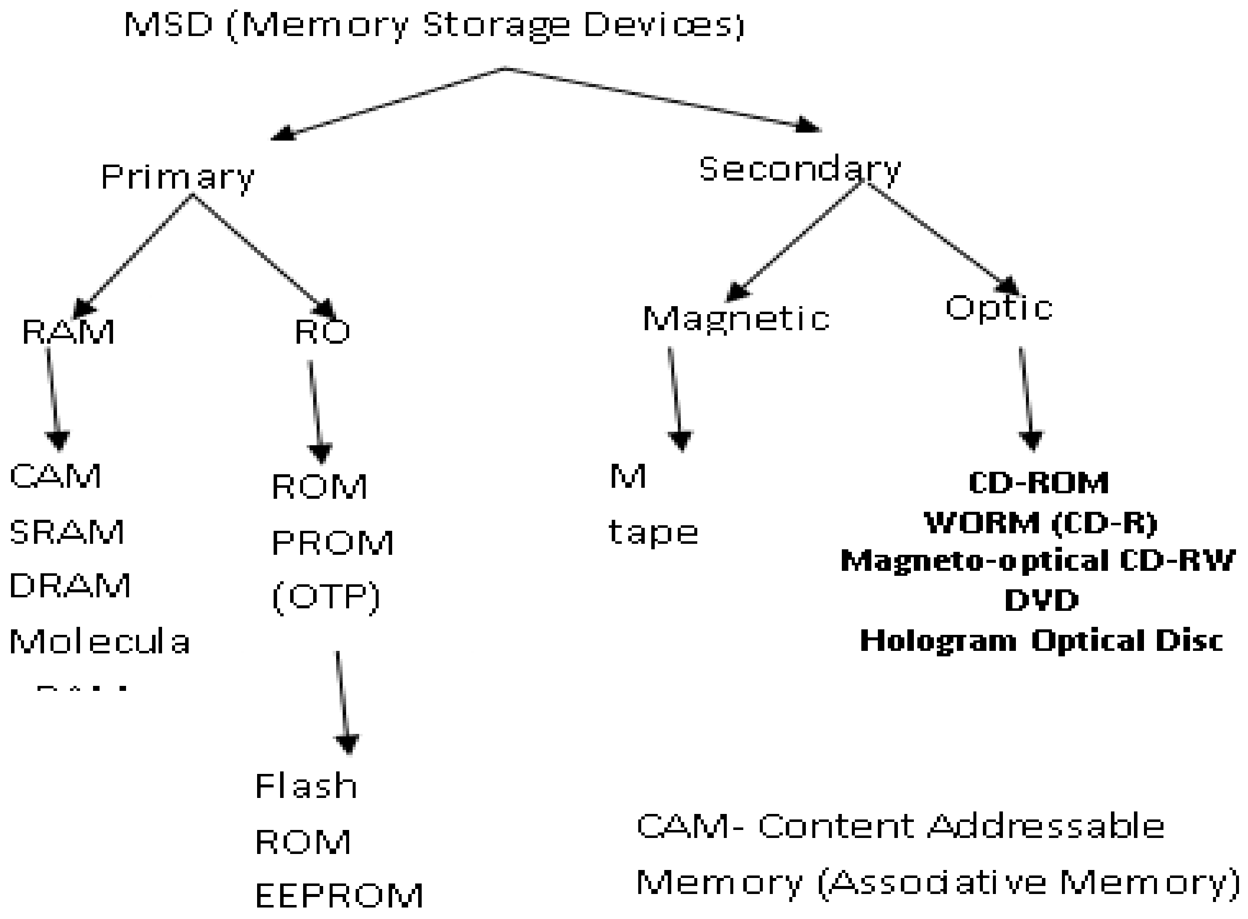
    RAM access time + Time to determine hit/miss
  - Miss: data needs to be retrieved from a block in the lower level (Block Y)
  - −Miss Rate= 1-(Hit Rate)
  - −Miss Penalty: time to replace a block in the upper level + time to deliver the block to the processor (latency + transmit time)
  - Hit Time << Miss Penalt



4 Questions for Hierarchy Design

- Q1: Where can a block be placed in the upper level? =>block placement
- Q2: How is a block found if it is in the upper level? =>block identification
- Q3: Which block should be replaced on a miss? =>block replacement
- Q4: What happens on a write?
- =>write strategy

Types of Memory



MSD (Memory Storage Devices)

Primary

Secondary

RAM

RO

Magnetic

Optic

CAM
SRAM
DRAM
Molecula

ROM
PROM
(OTP)

Flash
ROM
EEPROM

M
tape

CD-ROM
WORM (CD-R)
Magneto-optical CD-RW
DVD
Hologram Optical Disc

CAM- Content Addressable Memory (Associative Memory)

Types

- ROM: Read Only memory, permanent storage (boot code, embedded code)
- SRAM: Static Random Access memory, cache and high speed access
- DRAM: dynamic RAM (Main Memory)
- SDRAM - synchronous dynamic RAM
- DDR: Double Data Rate memory
- DDR-SDRAM - double-data-rate SDRAM
- MDRAM – multi-bank DRAM
- ESDRAM - cache-enhanced DRAM
- EPROM: Electrically Programmable ROM, replace ROM when reprogramming required
- EEPROM: Electrically Erasable Programmable ROM. Alternative to EPROM, limited but regular reprogramming.

Memory Design

- Goal
- to provide:
  - Adequate storage capacity
  - Acceptable level of performance
  - A reasonable cost

  The use of a number of different memory devices with different cost performance ratios organized to provide a high average performance at a low average cost per bit

  the individual memories form a hierarchy of storage devices

Components of the memory

- Internal processor memory
  - A small set of high-speed registers used for storage of instruction and data
- Main memory (primary memory)
  - A relatively large and fast memory used for program and data storage during computer operation
  - Locations in main memory can be accessed directly and rapidly by the CPU instruction set
  - Semiconductor technology
- Secondary memory (auxiliary memory)
  - Large memory capacity and slower than mm
  - Store system programs, large data files etc which are not continually required by the CPU
  - Information in secondary storage is accessed indirectly via I/O programs
  - Representative technologies used for secondary memory are magnetic and optic disks

Motivation for Memory Hierarchy

- Faster storage technologies are more costly
- o Cost more money per byte
- o Have lower storage capacity
- o Require more power and generate more heat
- • The gap between processing and memory is widening
- o Processors have been getting faster and faster
- o Main memory speed is not improving as dramatically
- • Well-written programs tend to exhibit good locality
- o Across time: repeatedly referencing the same variables
- o Across space: often accessing other variables located nearby
- Want the *speed of fast storage at the cost and capacity of*
- slow storage. Key idea: memory hierarchy!

Why Hierarchical Memory System?

- Economics and performance
- Main memory has less storage capacity, more expensive and slow (i.e. access time required to fetch data or instruction)
- Cache memory is the most expensive, smallest and fastest memory with least access time.

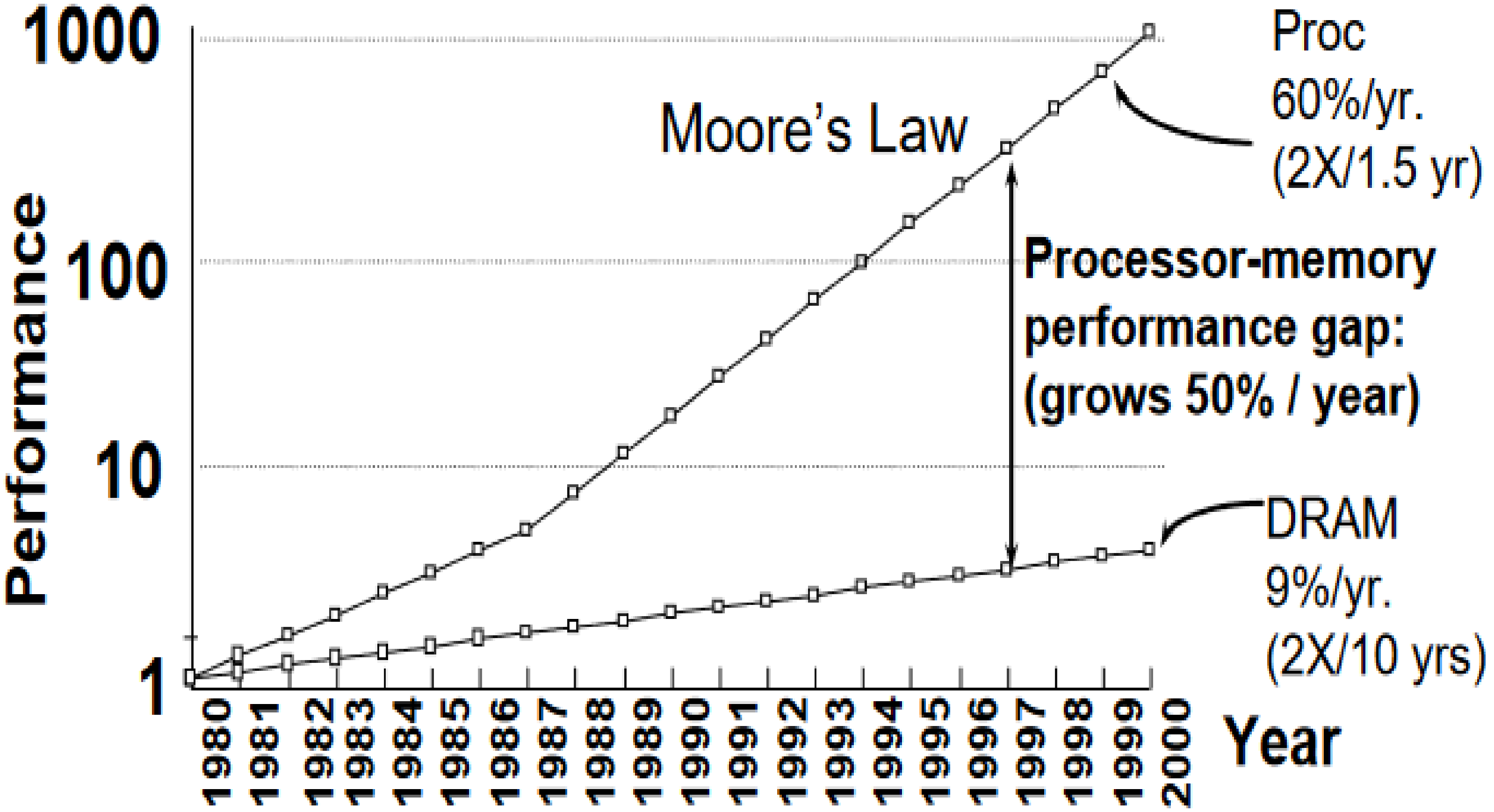
Simple Three-Level Hierarchy

- Registers
- o Usually reside directly on the processor chip
- o Essentially no latency, referenced directly in instructions
- o Low capacity (e.g., 32-512 bytes)
- Main memory
- o Around 100 times slower than a clock cycle
- o Constant access time for any memory location
- o Modest capacity (e.g., 512 MB-8GB)
- Disk
- o Around 100,000 times slower than main memory
- o Faster when accessing many bytes in a row
- o High capacity (e.g., 200 GB - TeraBytes)

Widening Processor/Memory Gap

- Gap in speed increasing from 1986 to 2000
- o CPU speed improved ~55% per year
- o Main memory speed improved only ~10% per year
- Main memory as major performance bottleneck
- o Many programs stall waiting for reads and writes to finish
- Changes in the memory hierarchy
- o Increasing the number of registers
- – 8 integer registers in the x86 vs. 128 in the Itanium
- o Adding caches between registers and main memory
- – On-chip level-1 cache and off-chip level-2 cache

Widening Processor/Memory Gap

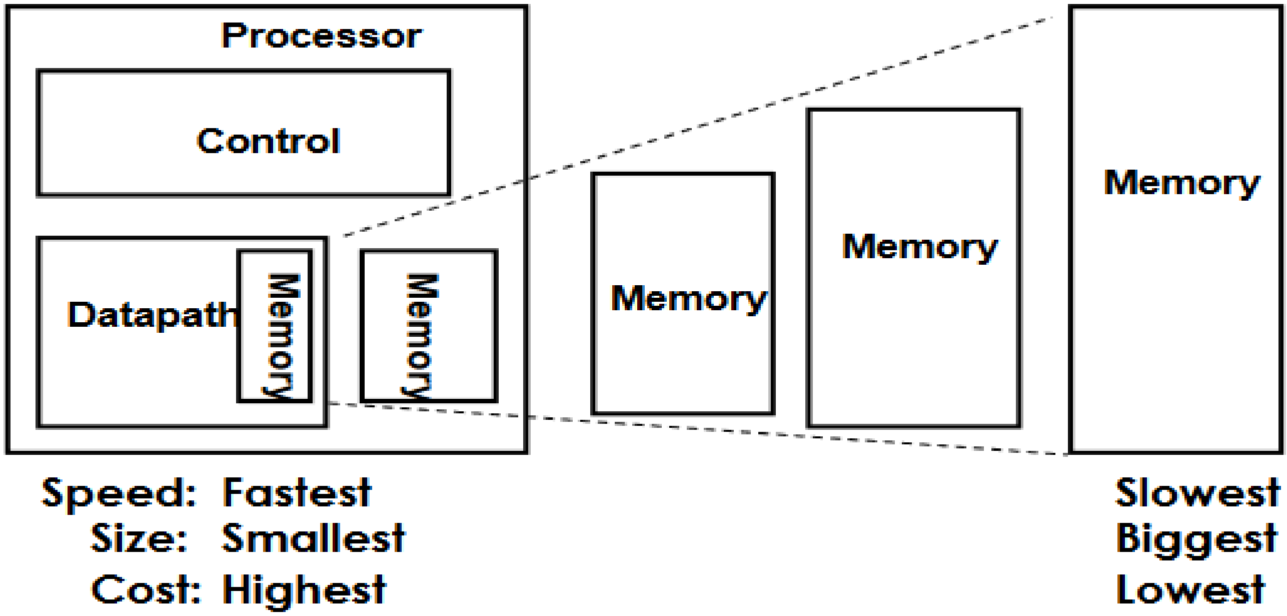


Solution: Memory Hierarchy

An Illusion of a large, fast, cheap memory

- Fact: Large memories slow, fast memories small
- How to achieve: hierarchy, parallelism

An expanded view of memory system:



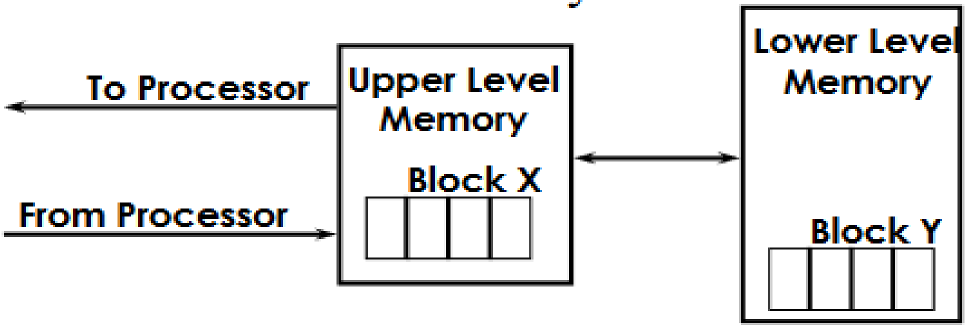
Memory Hierarchy: Principles

At any given time, data is copied between only two adjacent levels:

- Upper level: the one closer to the processor
  - Smaller, faster, uses more expensive technology
- Lower level: the one away from the processor
  - Bigger, slower, uses less expensive technology

Block: basic unit of information transfer

- Minimum unit of information that can either be present or not present in a level of the hierarchy



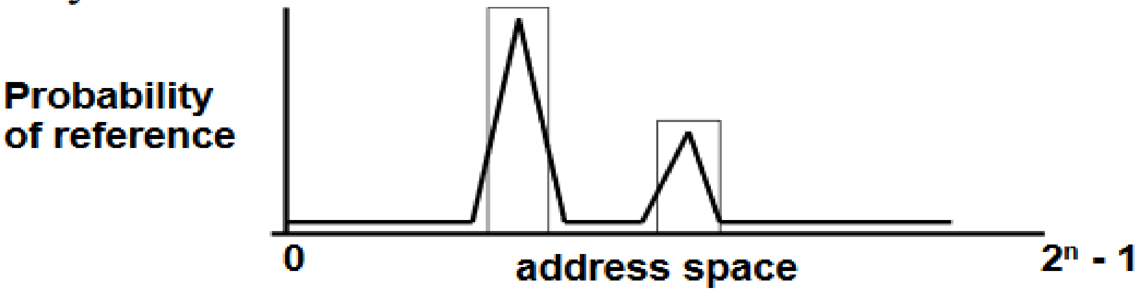
Why Hierarchy works?

Principle of Locality:

- Program access a relatively small portion of the address space at any instant of time
- 90/10 rule: 10% of code executed 90% of time

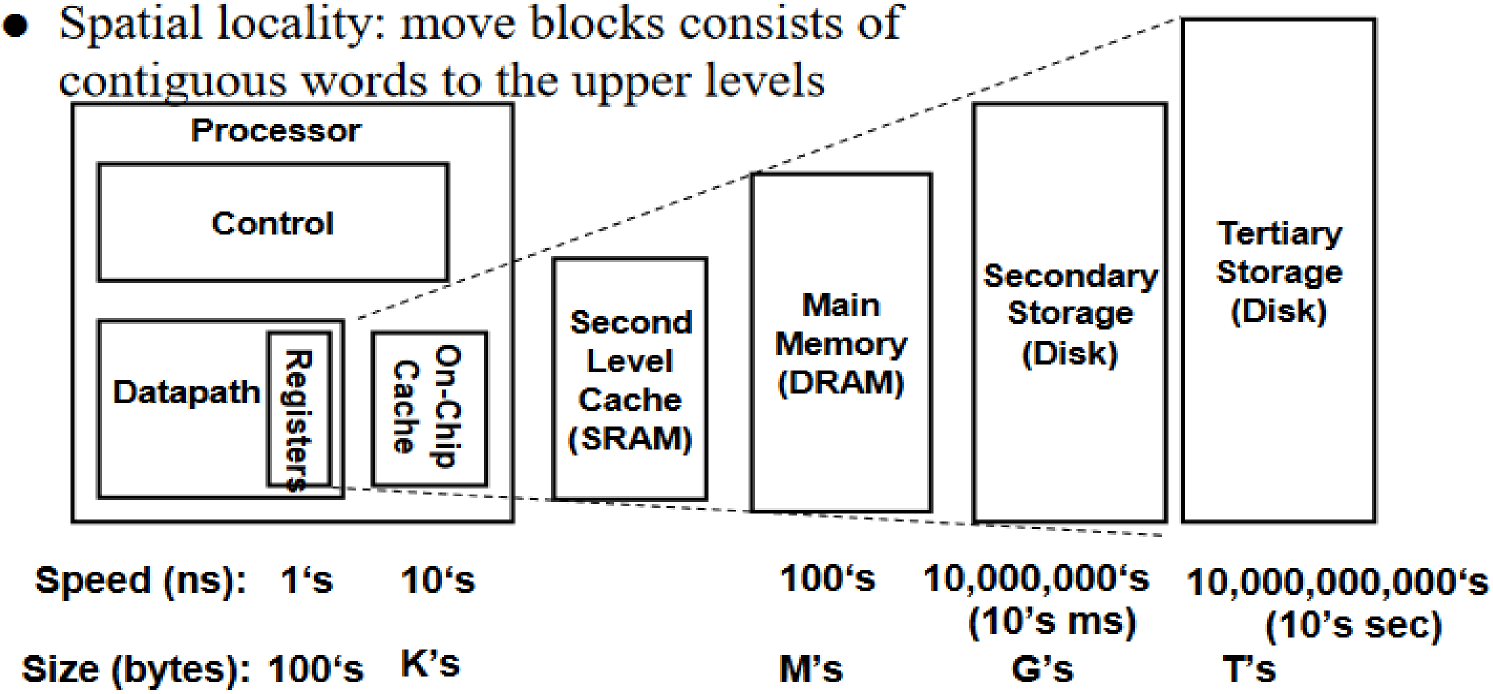
Two types of locality:

- Temporal locality: if an item is referenced, it will tend to be referenced again soon
- Spatial locality: if an item is referenced, items whose addresses are close by tend to be referenced soon

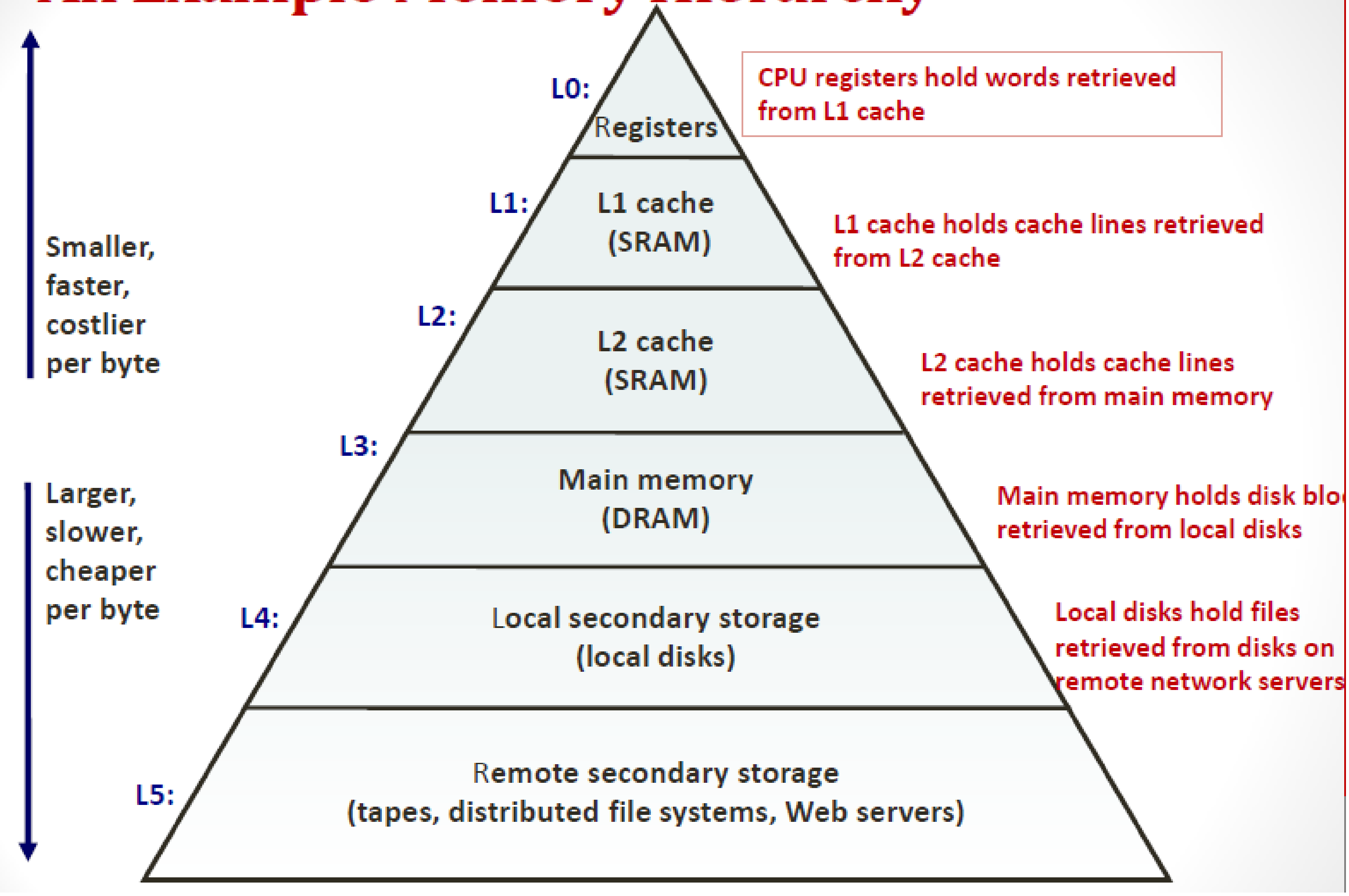


How Hierarchy Workss?

- Temporal locality: keep most recently accessed data items closer to the processor
- Spatial locality: move blocks consists of contiguous words to the upper levels


Processor
Control
Datapath
Registers
On-Chip Cache
Second Level Cache (SRAM)
Main Memory (DRAM)
Secondary Storage (Disk)
Tertiary Storage (Disk)
Speed (ns): 1‘s 10‘s 100‘s 10,000,000‘s (10’s ms) 10,000,000,000‘s (10’s sec)
Size (bytes): 100‘s K’s M’s G’s T’s

An Example Memory Hierarchy



Smaller,
faster,
costlier
per byte

Larger,
slower,
cheaper
per byte

L0: Registers

CPU registers hold words retrieved from L1 cache

L1: L1 cache (SRAM)

L1 cache holds cache lines retrieved from L2 cache

L2: L2 cache (SRAM)

L2 cache holds cache lines retrieved from main memory

L3: Main memory (DRAM)

Main memory holds disk blo
retrieved from local disks

L4: Local secondary storage (local disks)

Local disks hold files retrieved from disks on remote network servers

L5: Remote secondary storage (tapes, distributed file systems, Web servers)

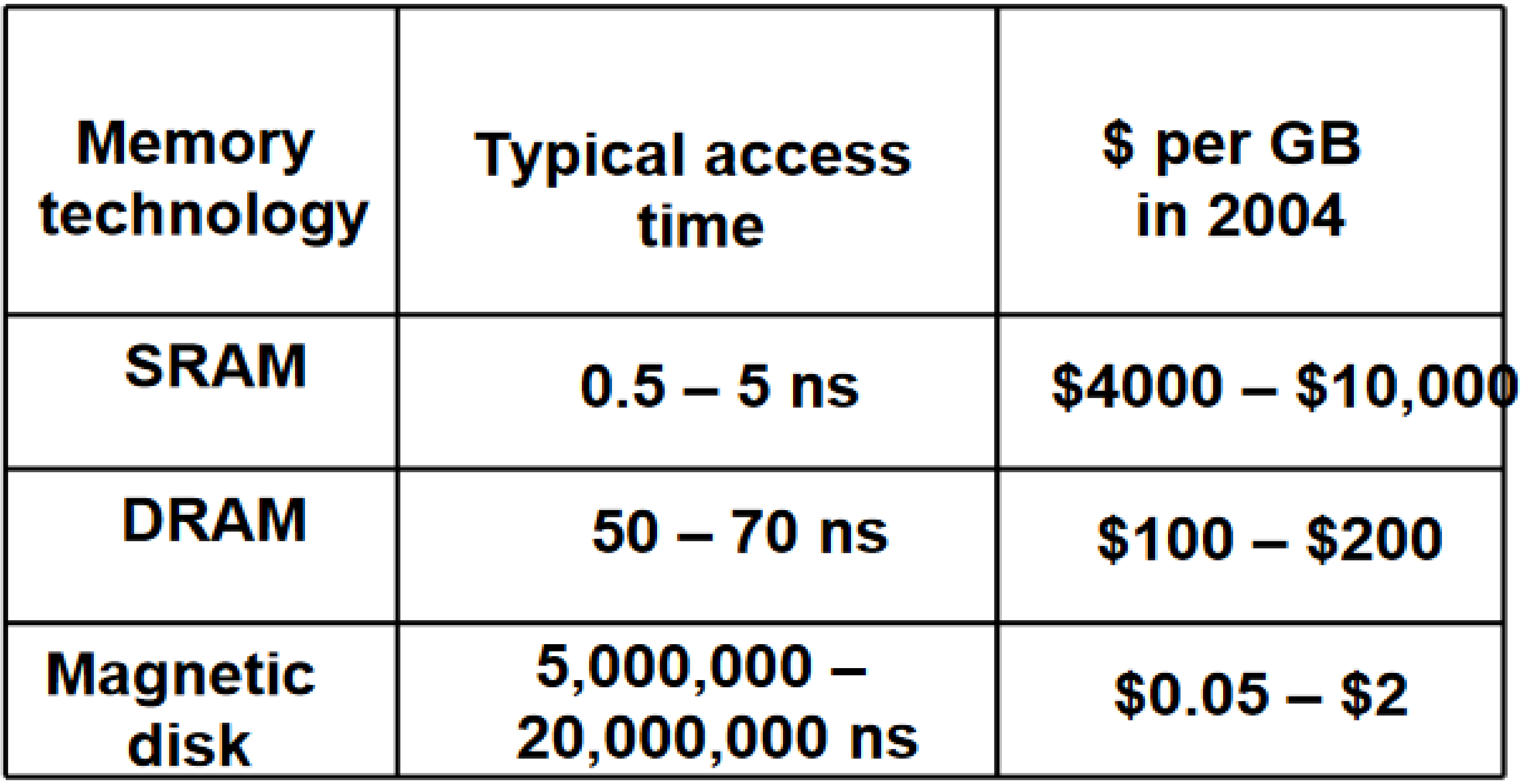
How is the Hierarchy Managed?

- Registers : managed by the compiler
- Cache : managed by the hardware
- Main memory: operating system
- Secondary memory: hardware and Operating System (virtual memory)
  - Programmer (files)

Memory Hierarchy Technology

- Random access:
- −Access time same for all locations
- −DRAM:Dynamic Random Access Memory
  - High density, low power, cheap, slow
  - Dynamic: need to be refreshed regularly
  - Addresses in 2 halves (memory as a 2D matrix):
    - −RAS/CAS (Row/Column Access Strobe)
  - Use for main memory
  - −SRAM:Static Random Access Memory
  - Low density, high power, expensive, fast
  - Static: content will last (forever until lose power)
  - Address not divided
  - Use for caches

Comparisons of Various Technologies



Memory technology	Typical access time	$ per GB in 2004
SRAM	0.5 – 5 ns	$4000 – $10,000
DRAM	50 – 70 ns	$100 – $200
Magnetic disk	5,000,000 – 20,000,000 ns	$0.05 – $2

Memory Hierarchy Technology

- Performance of main memory:
- −Latency: related directly to Cache Miss Penalty
  - Access Time: time between request and word arrives
  - Cycle Time: time between requests

  −Bandwidth: Large Block Miss Penalty (interleavedmemory, L2)

  Non-so-random access technology:

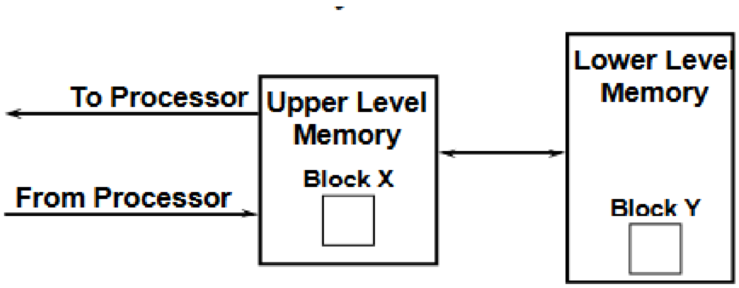
  −Access time varies from location to location and from time to time, e.g., disk, CDROM

  □Sequential access technology: access time linear in location (e.g., tape)

Memory Hierarchy: Terminology

- Hit: data appears in upper level (Block X)
  - −Hit rate: fraction of memory access found in the upper level
  - −Hit time: time to access the upper level

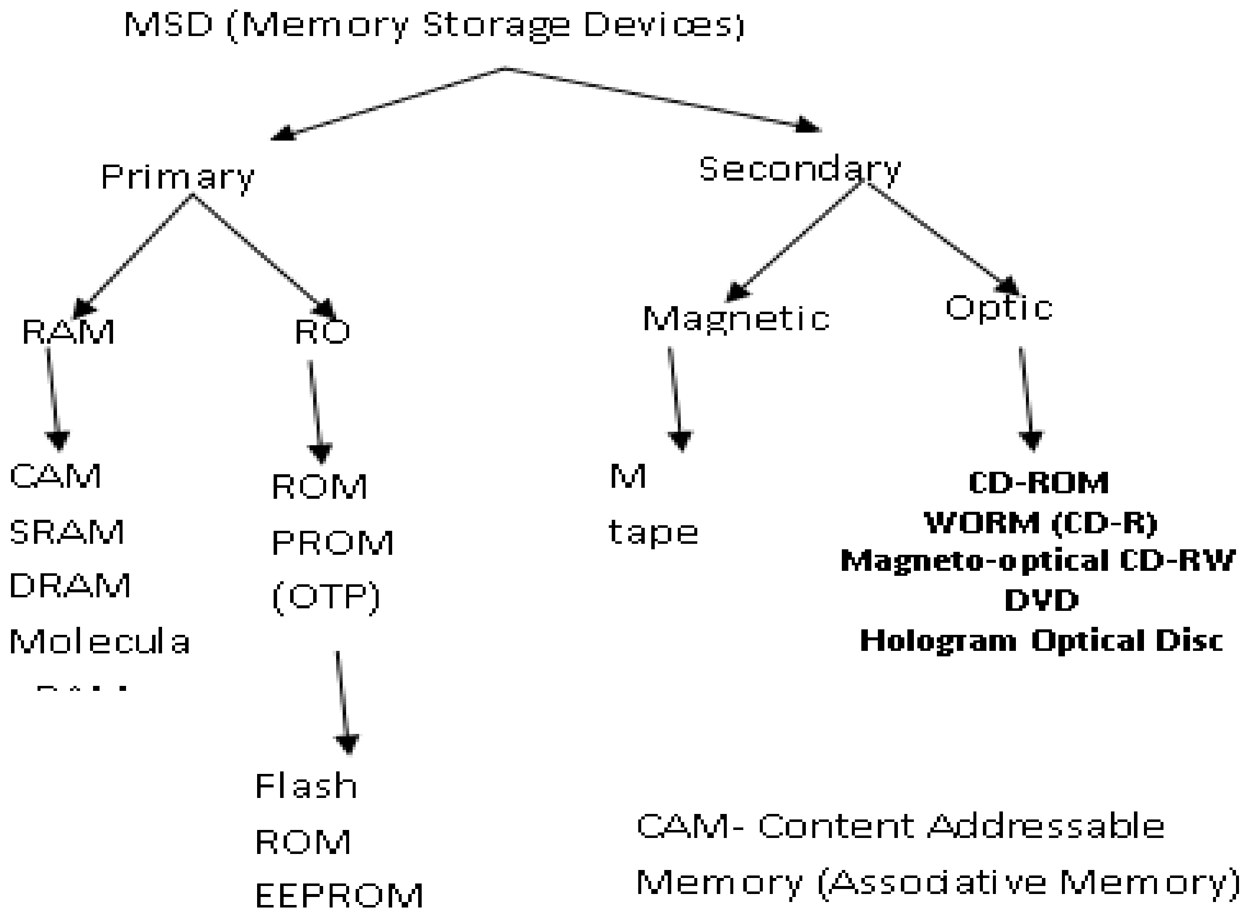
    RAM access time + Time to determine hit/miss
  - Miss: data needs to be retrieved from a block in the lower level (Block Y)
  - −Miss Rate= 1-(Hit Rate)
  - −Miss Penalty: time to replace a block in the upper level + time to deliver the block to the processor (latency + transmit time)
  - Hit Time << Miss Penalt



4 Questions for Hierarchy Design

- Q1: Where can a block be placed in the upper level? =>block placement
- Q2: How is a block found if it is in the upper level? =>block identification
- Q3: Which block should be replaced on a miss? =>block replacement
- Q4: What happens on a write?
- =>write strategy

Types of Memory



MSD (Memory Storage Devices)

Primary

RAM

CAM
SRAM
DRAM
Molecula

RO

ROM
PROM
(OTP)

Flash
ROM
EEPROM

Secondary

Magnetic

M
tape

Optic

CD-ROM
WORM (CD-R)
Magneto-optical CD-RW
DVD
Hologram Optical Disc

CAM- Content Addressable
Memory (Associative Memory)

Types

- ROM: Read Only memory, permanent storage (boot code, embedded code)
- SRAM: Static Random Access memory, cache and high speed access
- DRAM: dynamic RAM (Main Memory)
- SDRAM - synchronous dynamic RAM
- DDR: Double Data Rate memory
- DDR-SDRAM - double-data-rate SDRAM
- MDRAM – multi-bank DRAM
- ESDRAM - cache-enhanced DRAM
- EPROM: Electrically Programmable ROM, replace ROM when reprogramming required
- EEPROM: Electrically Erasable Programmable ROM. Alternative to EPROM, limited but regular reprogramming.