Computer Architecture

Topics covered:
Course outline and schedule
Introduction

General information

Course :Computer Architecture (CSE 2213)

Instructor : Syeda Shabnam Hasan
Assistant Professor, department of CSE,AUST.

Email : shabnam.cse@aust.edu
Room# : 7A01/L

Acknowledgements: ***These slides contain some materials developed and copyright by Swapna S. Gokhale***

Course Objective

- Describe the general organization and architecture of computers.
- Identify computers' major components and study their functions.
- Introduce hardware design issues of modern computer architectures.
- Build the required skills to read and research the current literature in computer architecture.

Textbooks

•"Computer Organization," by Carl Hamacher, Zvonko Vranesic and Safwat Zaky. Fifth Edition McGraw-Hill, 2002.

.David A. Patterson and John L. Hennessy, *Computer Organization and Design*: *The Hardware/Software Interface*, 3rd Edition, Morgan Kaufmann Publishers Inc.

Course topics

1. Basic structure of computers(Chapter 1): Basic concepts, overall organization.
2. Machine instructions and programs (Chapter 2): fetch/execute cycle, basic addressing modes, instruction sequencing, assembly language and stacks. CISC vs. RISC architectures.
3. Input/Output organization (Chapter 4): I/O device addressing, I/O data transfers, Synchronization, DMA, Interrupts, Channels, Bus transfers, and Interfacing.
4. The Memory System (Chapter 5): Memory hierarchy, Primary memory, Cache memory, virtual memory.
5. Arithmetic (Chapter 3:Patterson): Integer arithmetic and floating-point arithmetic.
6. Basic Processing Unit (Chapter 7): Single-bus CPU, Multiple-bus CPU Hardware control, and Micro programmed control.
7. Pipelining (Chapter 8): Basic concepts, Hazards.

Reading

- Reading the text is imperative.
- Computer architecture especially processor design, changes rapidly.

You really have to keep up with the changes in the industry.

This is especially important for job interviews later.

What will we Learn?

- Computer Architecture
  - The science and art of designing the hardware/software interface and designing, selecting, and interconnecting hardware components to create a computing system that meets functionality requirements, performance, energy consumption, cost, and other specific goals.

Tasks of a computer architect

- Determine which attributes are important for a new computer.
- Design a computer to maximize performance and energy efficiency while staying within cost, power and availability constraints. This task has many aspects:
  a) instruction set design
  b) functional organization
  c) logic design
  d) implementation; which encompass
     i. integrated circuit design
     ii. packaging
     iii. power and cooling
- Optimizing the design.

What is "Computer Architecture" ?

- Computer Architecture =
  Instruction Set Architecture + Computer Organization
- Instruction Set Architecture (ISA)
  - WHAT the computer does (logical view)
- Computer Organization
  - HOW the ISA is implemented (physical view)
- We will study both in this course

Instruction Set Architecture

- Instruction set architecture is the attributes of a computing system as seen by the assembly language programmer or compiler.
  - Instruction Set (what operations can be performed?)
  - Instruction Format (how are instructions specified?)
  - Data storage (where is data located?)
  - Addressing Modes (how is data accessed?)
  - Exceptional Conditions (what happens if something goes wrong?)

Computer Organization

- Computer organization is the view of the computer that is seen by the logic designer. This includes
  - Capabilities & performance characteristics of functional units (e.g., registers, ALU, shifters, etc.).
  - Ways in which these components are interconnected
  - How information flows between components
  - Logic and means by which such information flow is controlled
  - Coordination of functional units

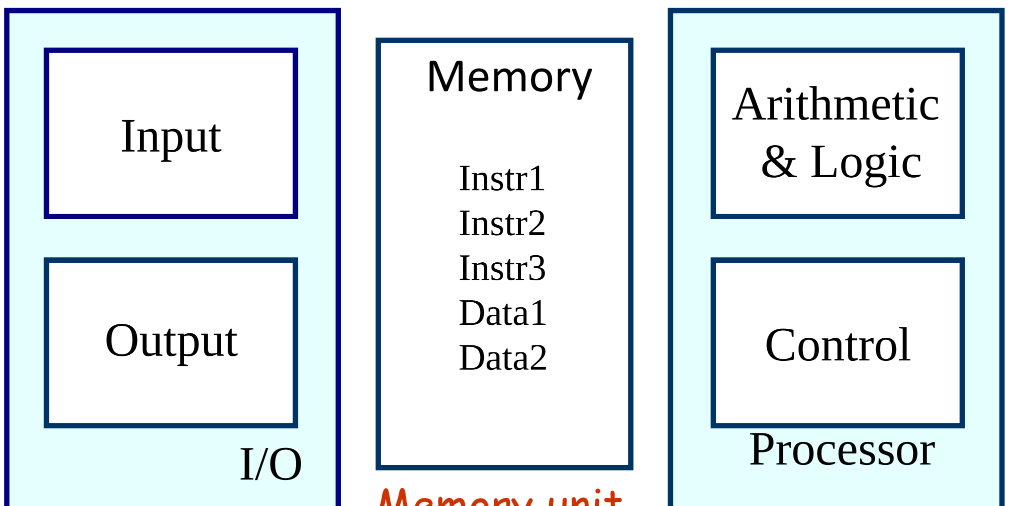
What is a computer?

- a computer is a electronic calculating machine that:
  - Accepts digitized input information,
  - Processes the information according to a list of internally stored instructions and
  - Produces the resulting output information.
- The list of instructions is called a computer program, and the internal storage is called computer memory.
- Functions performed by a computer are:
  - Accepting information to be processed as input.
  - Storing a list of instructions to process the information.
  - Processing the information according to the list of instructions.
  - Providing the results of the processing as output.

Basic functional units of a computer

Input unit accepts information:
- Human operators,
- Electromechanical devices (keyboard)
- Other computers

Arithmetic and logic unit(ALU):
- Performs the desired operations on the input information as determined by instructions in the memory



Memory unit Stores information:
- Instructions,
- Data

Output unit sends results of processing:
- To a monitor display,
- To a printer

Control unit coordinates various actions
- Input,
- Output
- Processing

Information in a computer -- *Instructions*

- Instructions are explicit commands that:
  - Transfer information within a computer (e.g., from memory to ALU)
  - Transfer of information between the computer and I/O devices (e.g., from keyboard to computer, or computer to printer)
  - Perform arithmetic and logic operations (e.g., Add two numbers, Perform a logical AND).
- *A sequence of instructions to perform a task is called a program, which is stored in the memory.*
- Processor fetches instructions that make up a program from the memory and performs the operations stated in those instructions.
- What do the instructions operate upon?

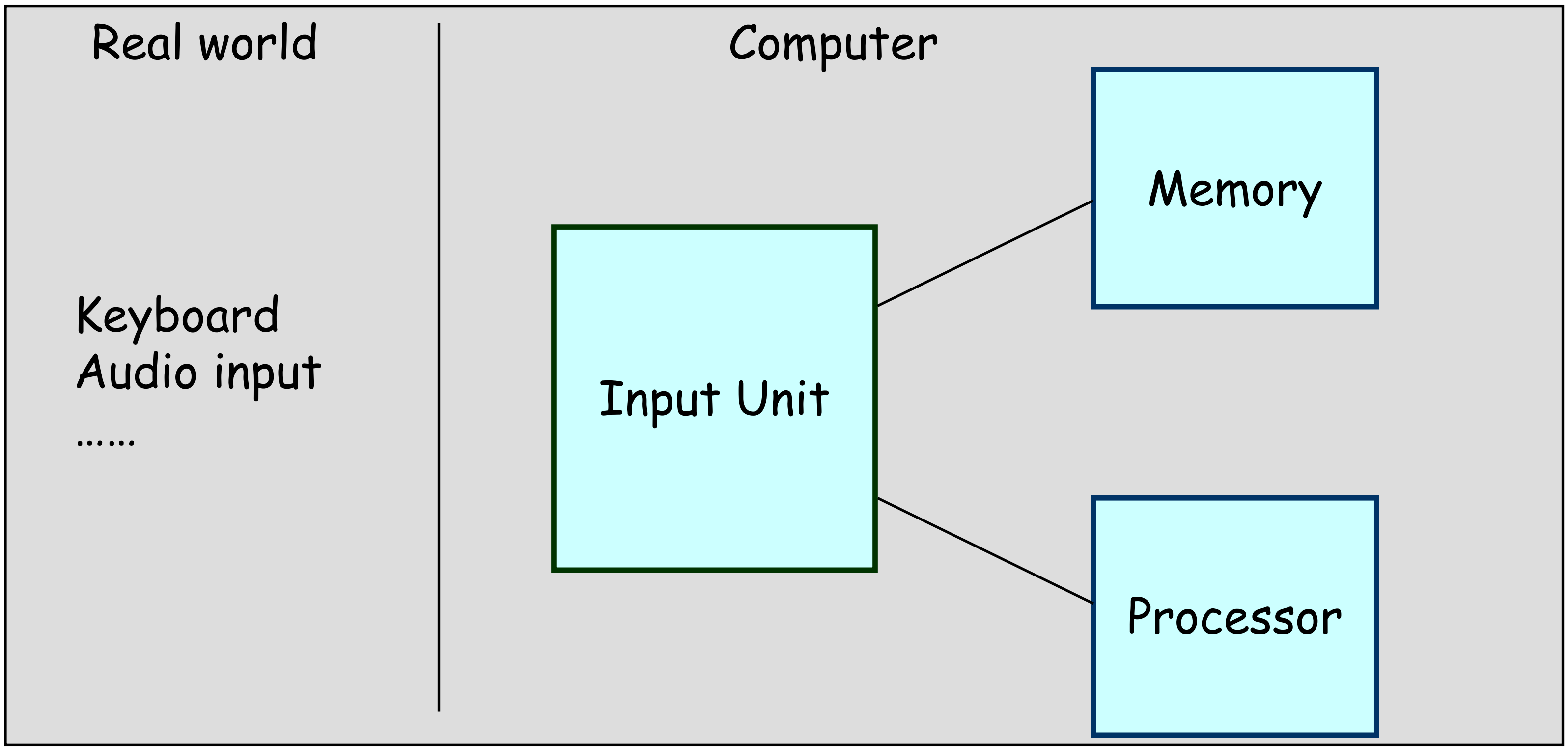
Information in a computer -- **Data**

- Data are the "operands" upon which instructions operate.
- Data could be:
  - Numbers,
  - Encoded characters.
- Data, in a broad sense means any digital information.
- Computers use data that is encoded as a string of binary digits called bits.

Input unit

Binary information must be presented to a computer in a specific format. This task is performed by the input unit:

- Interfaces with input devices.
- Accepts binary information from the input devices.
- Presents this binary information in a format expected by the computer.
- Transfers this information to the memory or processor.



Memory unit

- Memory unit stores instructions and data.
  - Recall, data is represented as a series of bits.
  - The memory contains a large number of semiconductor storage cells each capable of storing one bit of information.
- Processor reads instructions and reads/writes data from/to the memory during the execution of a program.
  - In theory, instructions and data could be fetched one bit at a time.
  - In practice, a group of bits is fetched at a time.
  - Group of bits stored or retrieved at a time is termed as "word"
  - Number of bits in a word is termed as the "word length" of a computer. Typical word lengths range from 16 to 64 bits.
- In order to read/write to and from memory, a processor should know where to look:"Address" is associated with each word location, addresses are numbers that identify successive locations. (Memory address)

Memory unit (contd..)

- Processor reads/writes to/from memory based on the memory address:
  - Access any word location in a short and fixed amount of time based on the address.
  - Random Access Memory (RAM) provides fixed access time independent of the location of the word.
  - Access time is known as "Memory Access Time".
- Memory and processor have to "communicate" with each other in order to read/write information.
  - In order to reduce "communication time", a small amount of RAM (known as Cache) is tightly coupled with the processor.
- Modern computers have three to four levels of RAM units with different speeds and sizes:
  - Fastest, smallest known as Cache
  - Slowest, largest known as Main memory.

Memory unit (contd..)

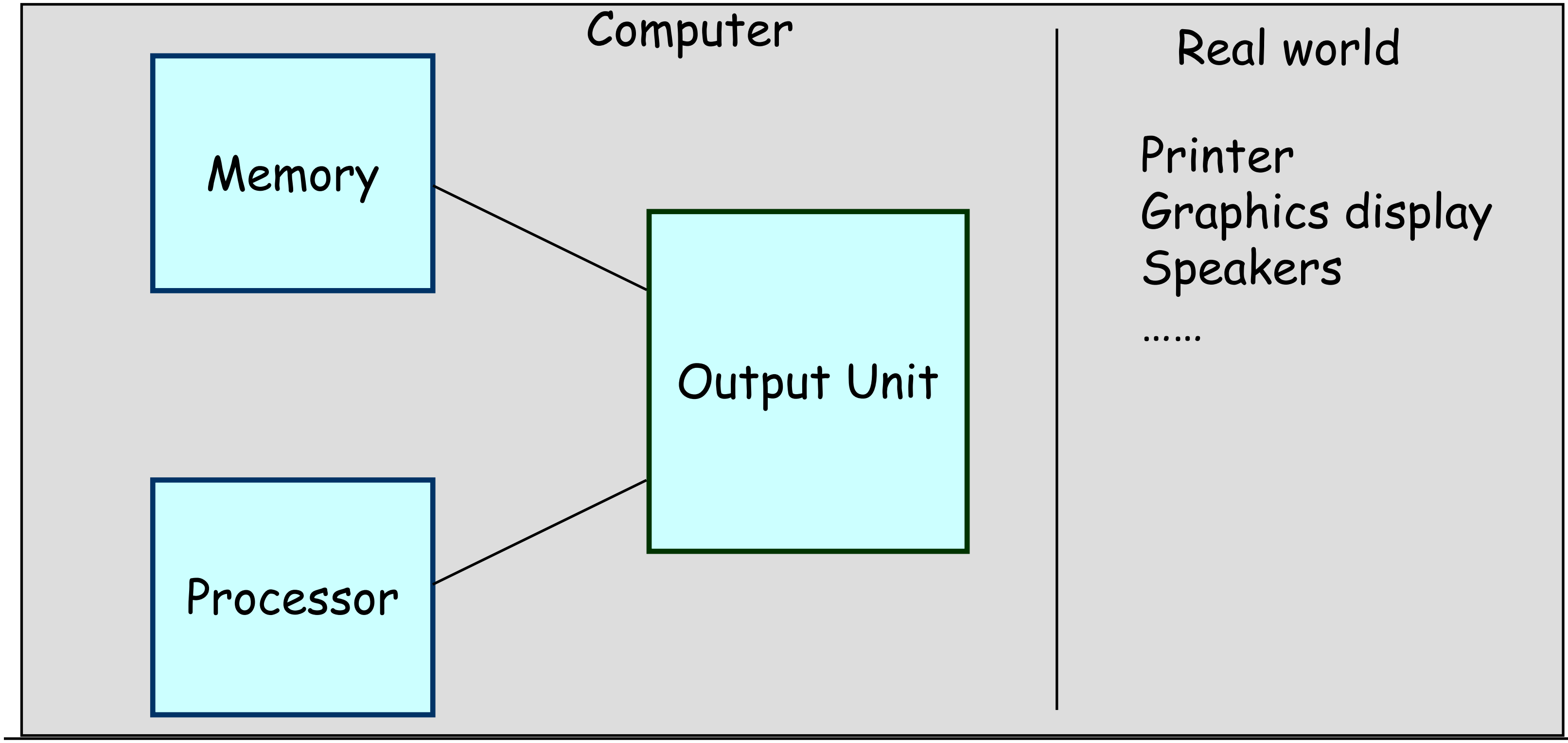
- There are 2 classes of storage called primary and secondary.
- Primary storage of the computer consists of RAM units.
  - Fastest, smallest unit is Cache.
  - Slowest, largest unit is Main Memory.
- Primary storage is insufficient to store large amounts of data and programs.
  - Primary storage can be added, but it is expensive.
- Store large amounts of data on secondary storage devices:
  - Magnetic disks and tapes,
  - Optical disks (CD-ROMS).
  - Access to the data stored in secondary storage in slower, but take advantage of the fact that some information may be accessed infrequently.
- Cost of a memory unit depends on its access time, lesser access time implies higher cost.

Arithmetic and logic unit (ALU)

- Most computer operations are executed in the Arithmetic and Logic Unit (ALU).
  - Arithmetic operations such as addition, subtraction.
  - Logic operations such as comparison of numbers.
- In order to execute an instruction, operands need to be brought into the ALU from the memory.
  - Operands are stored in general purpose registers available in the ALU.
  - Access times of general purpose registers are faster than the cache.
- Results of the operations are stored back in the memory or retained in the processor for immediate use.

Output unit

- Computers represent information in a specific binary form. Output units:
  - Interface with output devices.
  - Accept processed results provided by the computer in specific binary form.
  - Convert the information in binary form to a form understood by an output device and send processed results to the outside world.

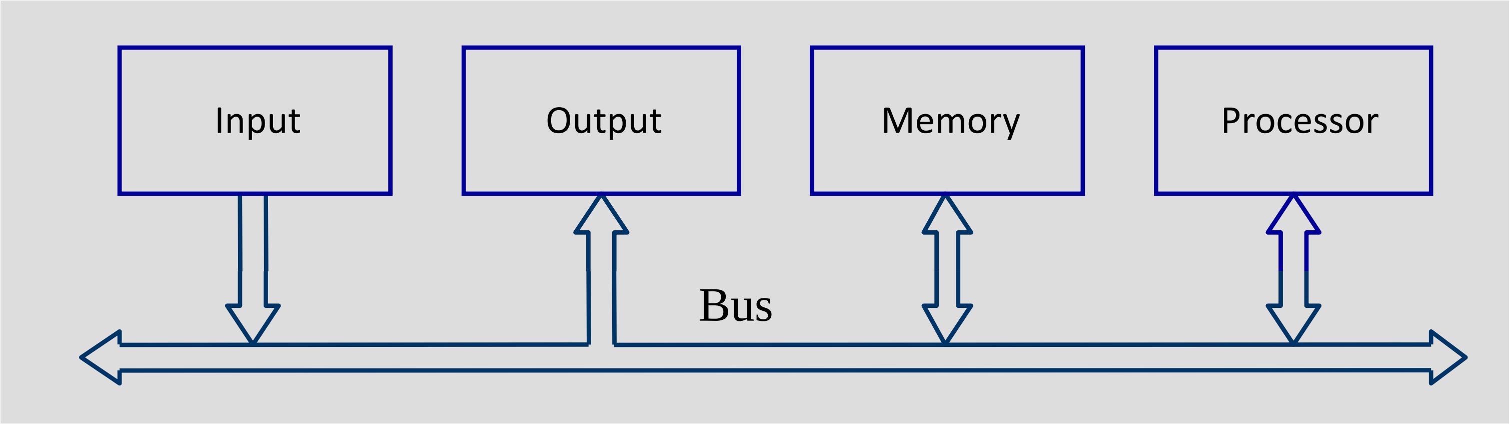


Control unit

- Operation of a computer can be summarized as:
  - Accepts information from the input units (Input unit).
  - Stores the information (Memory).
  - Processes the information (ALU).
  - Provides processed results through the output units (Output unit).
- Operations of Input unit, Memory, ALU and Output unit are coordinated by Control unit.
- Instructions control "what" operations take place (e.g. data transfer, processing).
- Control unit generates timing signals which determines "when" a particular operation takes place.

How are the functional units connected?

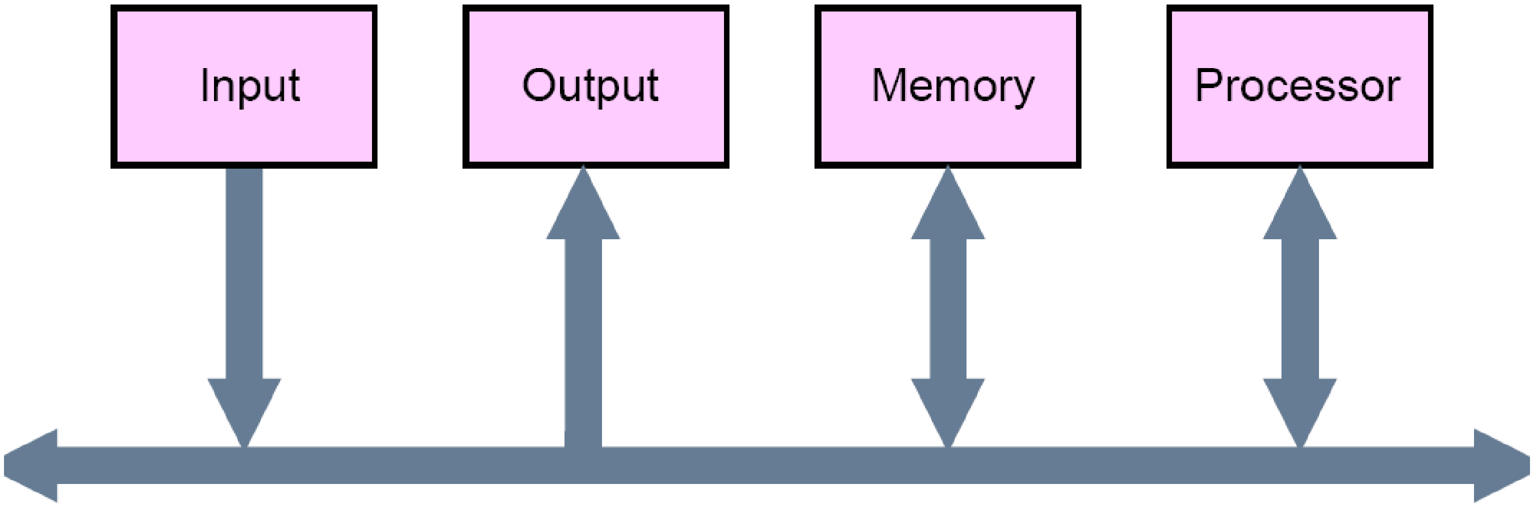
•For a computer to achieve its operation, the functional units need to communicate with each other.
•In order to communicate, they need to be connected.



•Functional units may be connected by a group of parallel wires.
•The group of parallel wires is called a ***bus***.
•Each wire in a bus can transfer one bit of information.
•The number of parallel wires in a bus is equal to the word length of a computer

Bus Structures

- A group of lines that serves a connecting path for several devices is called a bus
  - In addition to the lines that carry the data, the bus must have lines for address and control purposes
  - The simplest way to interconnect functional units is to use a single bus, as shown below (Single bus structure)



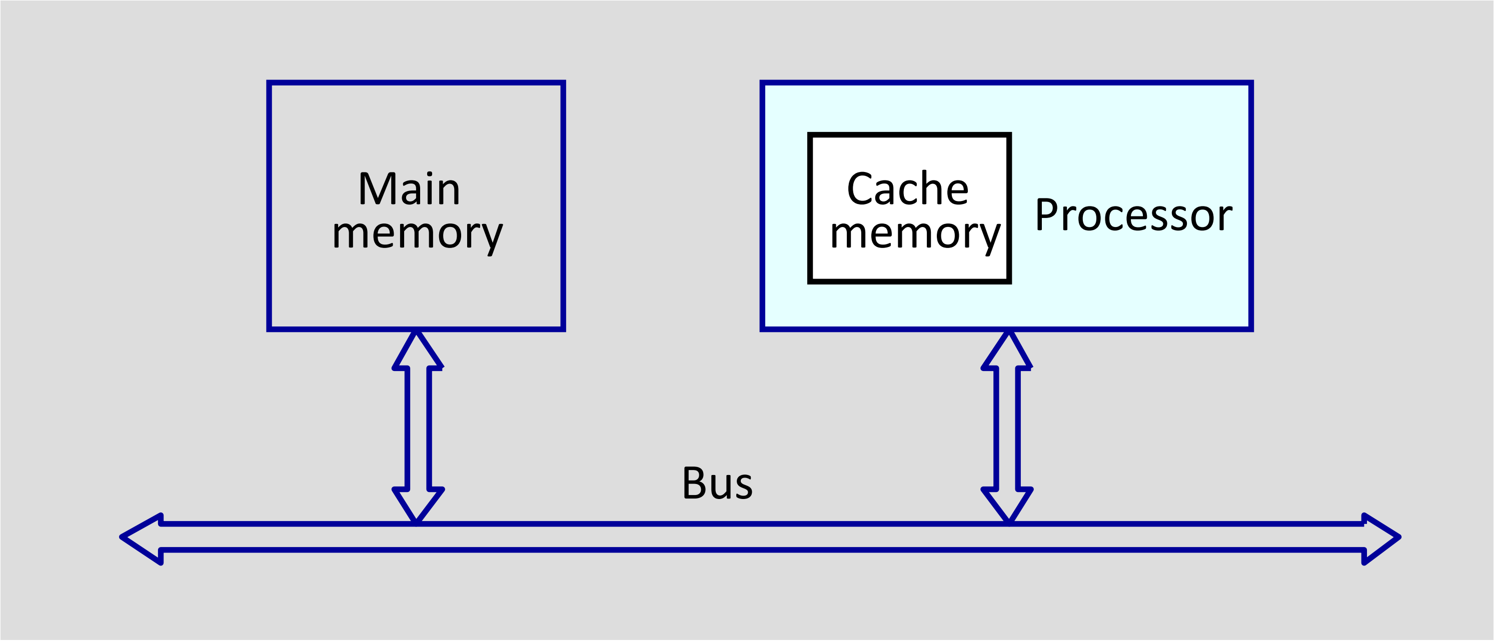
Drawbacks & advantages of the Single Bus Structure

- The devices connected to a bus vary widely in their speed of operation
  - Some devices are relatively slow, such as printer and keyboard
  - Some devices are considerably fast, such as optical disks
  - Memory and processor units operate are the fastest parts of a computer
- Efficient transfer mechanism thus is needed to cope with this problem
  - A common approach is to include buffer registers with the devices to hold the information during transfers

Advantages of the Single Bus Structure:

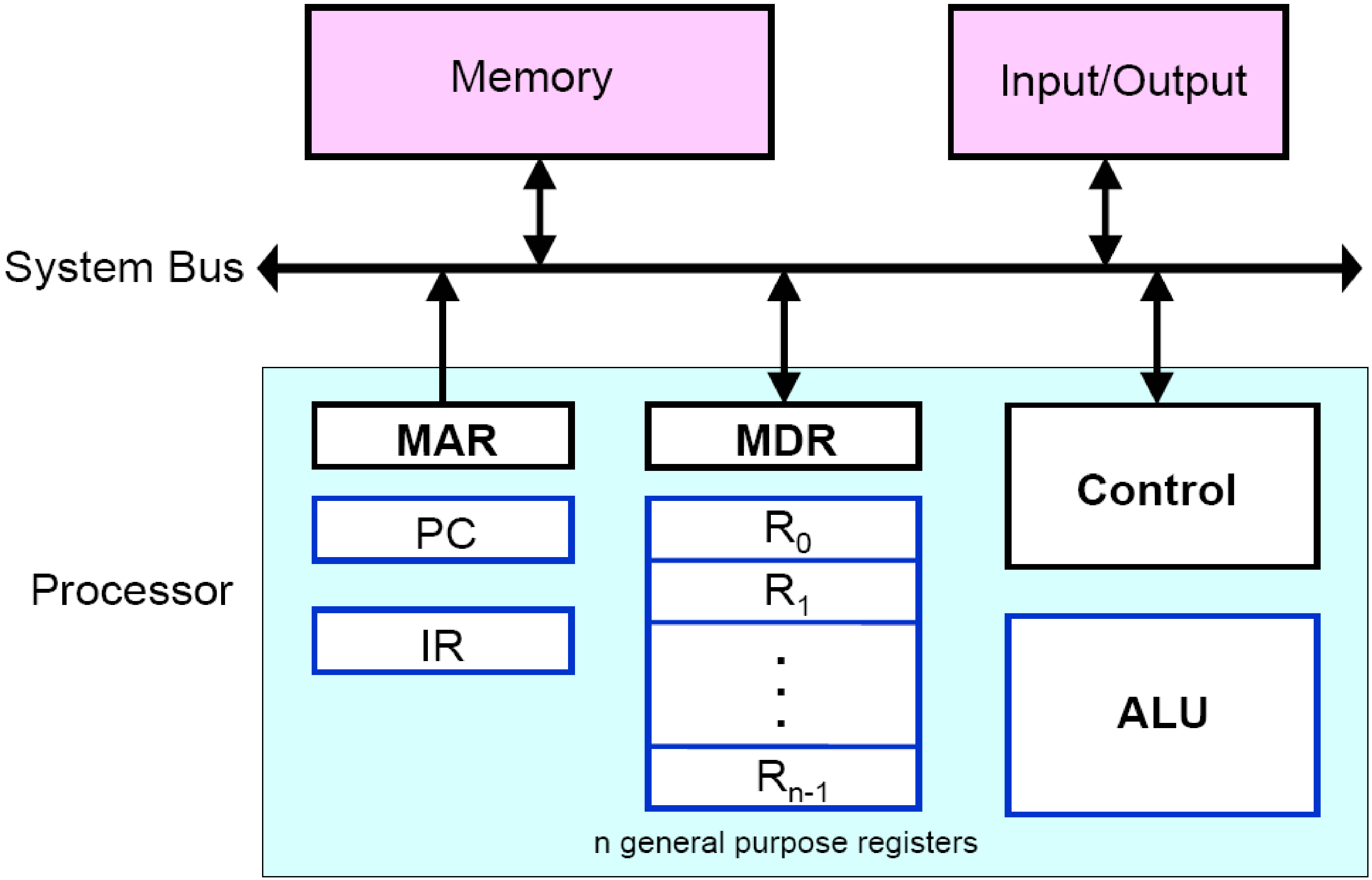
- Low cost
- Flexibility for attaching peripheral devices

Organization of cache and main memory



Why is the access time of the cache memory lesser than the access time of the main memory?

Computer Components: Top-Level View



Basic Operational Concepts

Review

- Activity in a computer is governed by instructions.
- To perform a task, an appropriate program consisting of a list of instructions is stored in the memory.
- **A Program = A sequence of instructions : Assembly language or Machine language instructions**
- Individual instructions are brought from the memory into the processor, which executes the specified operations.
- Data to be used as operands are also stored in the memory.

A Typical Instruction

- MOV LOCA, R0
- General format:

Instruction = Operation source_operand destination_operand

- Moves the operand at memory location LOCA to the operand in a register R0 in the processor.
- Simply: Moves the contents of Memory Location LOCA to the processor register R0
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction that Moves data **from Memory to Register** is called **LOAD** instruction (e.g., MOV LOCA, R0)
- Instruction that moves data from Register to Memory is called **STORE** instruction (e.g., MOV R0, LOCA)

Another Typical Instruction

- ADD LOCA, R0
- General format:

Instruction = Operation Source_operand Destination_operand

- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- **Place the sum into register R0.**
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

LOAD and Store Instructions to Transfer From/To Memory To/From Registers

Summary:

- MOV LOCA, R1 = means => Bring the content of memory location A into Register R1
- MOV R2, LOCB = means => save the value of register R2 in memory location B
- ADD R1, R0 == means => R0 ← [R0] + [R1] (Add the contents of both the registers R0 and R1 and store into register R0
  - For ADD, whose contents will be overwritten? (R0)
- **Load and Store Instructions**
- **LOAD** **LOCA, R1** equivalent to **MOV LOCA, R1**
- **STORE** **R2, LOCB** equivalent to **MOV R2, LOCB**

Examples of a Few Registers:

- **Instruction register (IR):** Holds the instruction that is currently executing by the CPU
- **Program counter register (PC):** Points to (i.e., holds the address of) the next instruction that will be fetched from the memory to be executed by the CPU
- **General-purpose registers (R_0 – R_{n-1}):** generally holds the operands for executing the instructions of current program
- **Memory address register (MAR):** Holds the memory address to be read. A read signal from the CPU to the memory module reads the **word** address held by the MAR register
- **Memory data register (MDR):** Contains the data to be written into or read out of the addressed location i.e Facilitates the transfer of operands/data to/from Memory from/to the CPU.

Executing a Program ... Basic Operating Steps

- Programs reside in the main memory (RAM) through input devices
- PC register's value is set to the first instruction

Repeat the following Steps Until the "END" instruction is executed

- **<u>Instruction fetch: -</u>**The contents of PC are transferred to MAR

-A Read signal is sent by CU to the memory

-The Memory module reads out the location addressed by MAR register. The contents of that location is loaded into (returned by) MDR

-The contents of MDR are transferred to IR register

- **<u>Decode and execute</u>** -At this point, the instruction is ready to be decoded and executed. Instruction in the IR is examined (decoded) to determine which operation is to be performed.

- Get operands for ALU: Fetch the operands from the memory or registers.

Executing a Program … Basic Operating Steps…

- -The operand may already in a General-purpose register
- -Or, may be fetched from Memory (send address to MAR – send Read signal to Memory module – Wait for MFC signal (WMFC) from Memory – Get the operand/data from MDR)

- Perform operation in ALU
- Store the result back
  - Store in a general-purpose register
  - Or, store into memory (send the write address to MAR, and send result to MDR – Write signal to Memory – WMFC)
  - WMFC = Wait for Memory Function Complete Signal
- Meanwhile, PC is incremented to the next instruction
- **Some Examples:** Add R_0, R_1; Add (R_0), R_1; Add 50(R_0), R_1;

Interrupt

- Normal execution of programs may be interrupted if some device requires urgent servicing
  - To deal with the situation immediately, the normal execution of the current program must be interrupted

- Procedure of interrupt operation
  - The device raises an interrupt signal
  - The processor provides the requested service by executing an appropriate interrupt-service routine
  - The state of the processor is first saved before servicing the interrupt
    - Normally, the contents of the PC, the general registers, and some control information are stored in memory
  - When the interrupt-service routine is completed, the state of the processor is restored so that the interrupted program may continue

Classes of Interrupts

- Program
  - Generated by some condition that occurs as a result of an instruction execution such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space
- Timer
  - Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis
- I/O
  - Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions
- Hardware failure
  - Generated by a failure such as power failure

Software

- In order for a user to enter and run an application program, the computer must already contain some system software in its memory

- System software is a collection of programs that are executed as needed to perform functions such as
  - Receiving and interpreting user commands
  - Running standard application programs such as word processors, etc, or games
  - Managing the storage and retrieval of files in secondary storage devices
  - Controlling I/O units to receive input information and produce output results

Software

- Translating programs from source form prepared by the user into object form consisting of machine instructions
- Linking and running user-written application programs with existing standard library routines, such as numerical computation packages
- System software is thus responsible for the coordination of all activities in a computing system

Operating System

- Operating system (OS)
  - This is a large program, or actually a collection of routines, that is used to control the sharing of and interaction among various computer units as they perform application programs

- The OS routines perform the tasks required to assign computer resource to individual application programs
  - These tasks include assigning memory and magnetic disk space to program and data files, moving data between memory and disk units, and handling I/O operations

Performance

- The most important measure of a computer is how quickly it can execute programs i.e., Runtime of programs. The speed with which a computer executes programs is affected by the design of its hardware and its machine language instructions. Because programs are usually written in a high-level language, performance is also affected by the compiler that translates programs into machine languages.

- For best performance, the following factors must be considered
  - Compiler
  - Instruction set
  - Hardware design

Performance

- Three factors affect performance:
- **Hardware design** (e.g., CPU clock rate)
  - 1GHz CPU => 1 Billion Hz => 10^9 clock cycles/sec (Hz=cycles/sec)
    - 1 basic operation (e.g., integer addition) possible in 1 cycle => 1 billion basic operations (10^9 integer additions!) possible in 1 sec!!! WOW!!!
  - 1Mhz => 1 Million Hz => 10^6 clock cycles/sec
- **Instruction set architecture (ISA)** (e.g., CISC or RISC ISA?)
  - CISC => instructions complex, more capable, but runs slower
  - RISC => instructions Simple, runs faster, but less capable
- **Compiler** (how efficient your compiler to optimize your code for pipelining…etc?)

Performance

- Processor circuits are controlled by a timing signal called a clock
  - The clock defines regular time intervals, called clock cycles
- To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps, such that each step can be completed in one clock cycle
- Let the length P of one clock cycle, its inverse is the clock rate, R=1/P

Processor Clock

- Clock, clock cycle, and clock rate
  - Clock Rate = 1 GHz = 10^9 Hz = 10^9 cycles/second or 10^9 clock pulses per second !!! WOW!!! It also means it has a Clock Cycle of $1/10^9 = 10^{-9}$ sec = 1 ns (nano-second).
  - 4GHz CPU => 4×10^9 cy/sec => 1 clock cycle = 0.25 ns
  - 500 MHz => 500×10^6 cycles/sec => 2 ns clock pulses
  - 1 MHz = 10^6 cycles/sec; 1KHz=10^3 cycles/sec
  - 1GHz=1000MHz, 1MHz=1000KHz, 1KHz=1000Hz
  - Hz (Hertz) – cycles per second (clock cycles / second)

Basic Performance Equation

$$T = \frac{N \times S}{R}$$

- T – processor time required to execute a program that may have been prepared in high-level language
- N – ***Dynamic Instruction Count***. It is the number of actual machine language instructions needed to complete the execution (note: A single 1-line loop may execute more than a billion times !!!)
- S – average number of basic steps (or, clock cycles) needed to execute one machine instruction. Each basic step completes in one clock cycle. Unit: cycles/instruction
- R – clock rate: cycles/sec
- Note: **these are not independent to each other**
- **How to improve T?**

  - reduce N x S, Increase R

Basic Performance Equation

- *T*–program execution time. Unit: second
- *N* – **Unit: instructions**
- *S* – Unit: **cycles/instructions**
- *R*–clock rate: **cycles/second**

Example: A program with dynamic instruction count (*N*) of 1000 instructions, each instruction taking 5 cycles on average (*S*=5 cycles/instruction) and running at a speed of 1KHZ (R = 10^3 Or 1000 cycles/second), what will be the program execution time T?

Ans: $T = \dfrac{1000 \text{ instructions} \times 5 \text{cycles/ instruction}}{1000 \text{ cycles/sec}}$

$= 5$ sec

Overview

- **The execution time** ***T*** of a program that has a **dynamic instruction count** ***N*** is given by:

$$T = \frac{N \times S}{R}$$

unit: **second**, because $\frac{instructions \times cycles/instruction}{cycles/second}$

Here ***S*** **is the average number of clock cycles it takes to fetch and execute one instruction**, and ***R*** **is the clock rate.** (The dynamic instruction count *N* is computed considering loops, repeated function calls, recursion, etc!)

- **Instruction throughput** is defined as the number of instructions executed per second.

$$P_s = \frac{R}{S}$$

unit: **instructions / second**, because: $\frac{cycles/second}{cycles/instruction}$

Performance Improvement

- Pipelining and superscalar operation
  - Pipelining: by overlapping the execution of successive instructions
  - Superscalar: different instructions are concurrently executed with multiple instruction pipelines. This means that multiple functional units are needed
- Clock rate improvement
- Improving the integrated-circuit technology makes logic circuits faster, which reduces the time needed to complete a basic step

Performance Improvement

- Reducing amount of processing done in one basic step also makes it possible to reduce the clock period, P.
- However, if the actions that have to be performed by an instruction remain the same, the number of basic steps needed may increase
- Reduce the number of basic steps to execute
  - Reduced instruction set computers (RISC) and complex instruction set computers (CISC)

Improving Performance: Effect of Instruction Set Architectures (ISA), e.g., CISC and RISC ISA

- **Reduced Instruction Set Computers (RISC): simpler instructions** => N↑, S↓, Better than CISC, because Pipelining is more effective for RISC!!

- **Complex Instruction Set Computers (CISC):**

Complex instructions => N↓, S↑ , Not Good, As not suitable for Pipelining!! Instructions complex, more capable => the program gets smaller in size (reduced *N*), but complex instructions increase *S* and hampers/stalls pipeline. Example of CISC: Intel processors

- So, *A* key consideration is the use of **Pipelining**

- *S* is close to 1, means the number of cycles per instruction is nearly ideal / small (close to 1) (e.g. RISC processors)
- **RISC is Better**, because easier to implement efficient pipelining with simpler instruction sets. (example of RISC architecture: ARM processors

Performance Measurement

- *T* is difficult to compute. Also, *T* has inappropriate unit (second) for commercial use.
- Measure computer performance using benchmark programs (a set of sample programs, e.g., word processing programs, games, media (audio/video) playback , I/O intensive programs, etc ...).
- **System Performance Evaluation Corporation (SPEC)** selects and publishes representative application programs for different application domains, together with test results for many commercially available computers.
- Reference computer: A previous, renowned computer system, picked by SPEC

$$SPEC\ rating = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

$$SPEC\ rating = \left(\prod_{i=1}^{n} SPEC_i\right)^{\frac{1}{n}}$$

Multiprocessors and Multi-computers

- Multiprocessor computer
  - Good for Executing several different application tasks in parallel
  - Good for Executing subtasks of a single large task in parallel
  - All processors have access to all of the memory – shared-memory multiprocessor. Example: Some **commercial server computers using two/four processors**. www.cpubenchmark.net/multi_cpu.html
  - Cost– processors, memory units, complex interconnection networks
- Multicomputers
  - Each computer only have access to its own memory
  - Example: **a Network of computers, such as a LAN (Local Area Network), WAN (wide area network) or MAN (metropolitan area network)** etc.
  - Exchange message via a communication network – message-passing multi-computers