ECE 341

Lecture # 8

Instructor: Zeshan Chishti
zeshan@pdx.edu

October 22, 2014

Portland State University

Lecture Topics

- Basic Organization and Operation of Computers
  - Functional units of a computer
  - Computer performance
- Basic Processing Unit
  - Fundamental Concepts
    - Instruction Processing Steps
    - Basic Processing Hardware
    - RISC Processors

- Reference:
  - Chapter 1: Section 1.2 and 1.6
  - Chapter 5: Section 5.1

Information Processed by a Computer

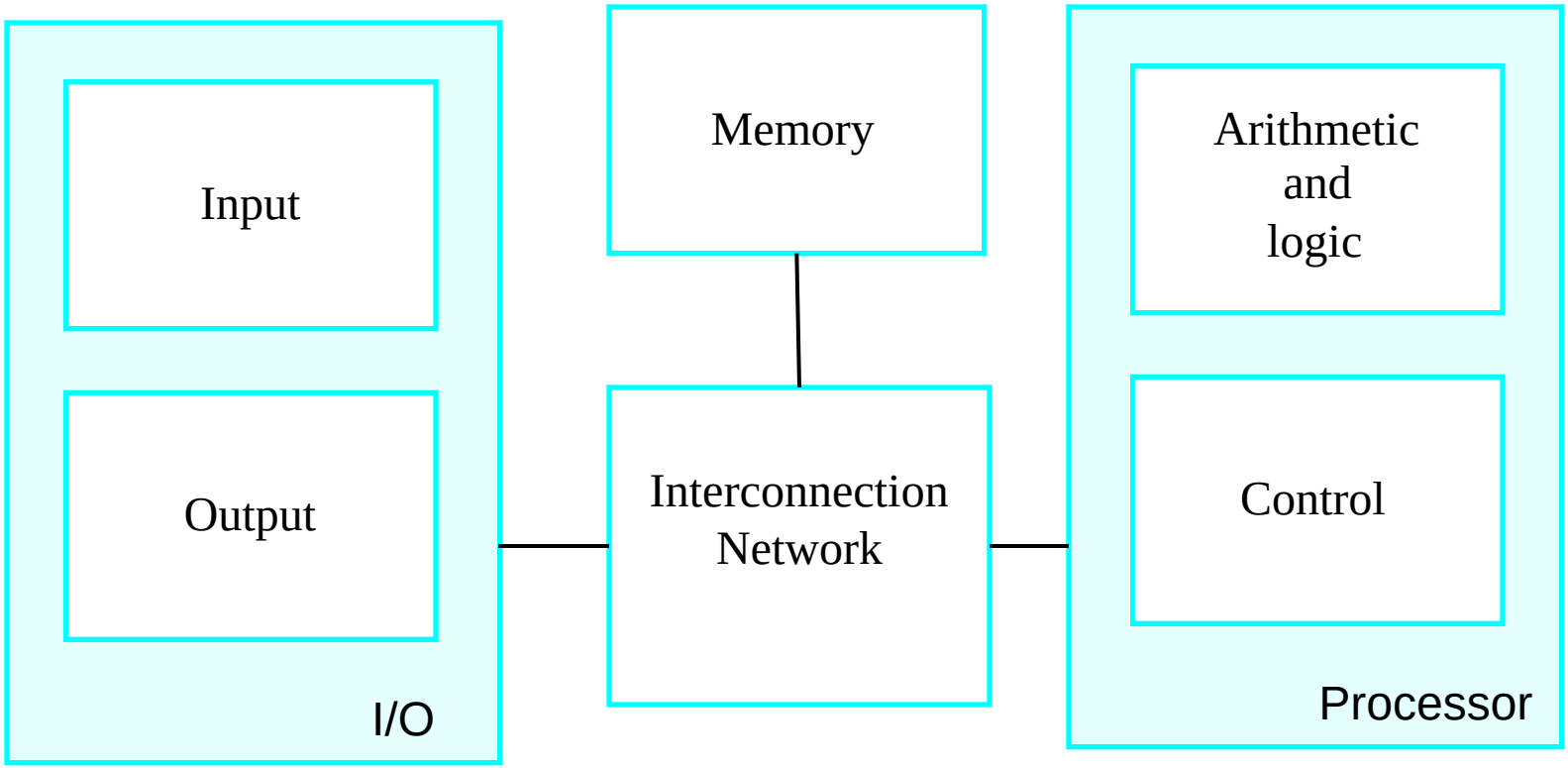
- Two types of information: (i) Instructions, (ii) Data

- **Instructions** govern the activity of a computer:
  - Specify the arithmetic and logic operations to be performed
  - Control information transfer between (and within) different units of a computer
- A **program** is a list of instructions which work together to perform a task

- **Data** is the numbers and characters used as operands by the instructions

- Both instructions and data encoded as a string of binary digits (or *bits*)

Functional Units



Basic Functional Units of a Computer

Input and Output Units (I/O)

- I/O units are an interface for the operator to interact with the computer

- Computer *accepts* coded information through *input* units
- Examples
  - Keyboard, mouse, microphone, video camera

- Computer *sends* processed results to the outside world via *output* units
- Examples
  - Printer, Graphic display

- Some units provide both input and output capability
  - Touch screens in mobile phones

Memory Unit

- Stores programs and data
- Two classes of storage
- Primary Memory (or Main memory)
  - Fast and volatile
  - Programs must be stored in main memory while being executed
  - Read or written at *word* granularities
  - Each memory access requires an *address* and a *control command* (read/write)
  - A faster memory, called ***cache***, keeps a subset of main memory which is currently being used by the processor
- Secondary Storage
  - Less expensive and non-volatile
  - Orders of magnitude slower than primary memory

ALU and Control Units

- Arithmetic/Logic Unit (ALU) and Control Unit together form the ***processor***

- **ALU** executes most of the computer operations
  - Addition, subtraction, multiplication etc.
- ALU operands need to be brought from the memory to the processor
- ALU results stored in memory or kept in ***registers*** for immediate re-use

- **Control unit** co-ordinates the operations of other units
  - *Nerve center* of the computer
  - Generates timing signals, sends to other units to control data transfer & operations
  - Not a single centralized unit, physically distributed throughout the computer by means of large sets of control wires

Putting Everything Together

- Program & data information accepted through *input* units and stored in *memory*
- Information fetched from *memory* into the processor under program control
- *ALU* processes the information and generates results
- Processed information leaves the computer through an *output* unit
- All the above activities are coordinated by the *control* unit

Performance

- The traditional goodness metric of computers is *performance*
- Performance is measured in terms of a program's execution time
- Basic performance equation:

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

  - *T*: Time required to execute a program
  - *N*: Number of instructions executed to complete a program
  - *CPI*: Average number of clock cycles per instruction
  - *R*: Clock rate

Performance Equation Parameters

- **Clock rate (R)**
  - Instruction's execution divided into multiple steps, each step 1 *clock cycle* long
  - Clock rate is equal to number of clock cycles per second (Hertz or Hz)
  - Clock period is equal to the length of clock cycle (clock rate = 1 / clock period)
- **Cycles per Instruction (CPI)**
  - Avg. no. of clock cycles required to execute an instruction
    - For ALU instructions, CPI depends on circuit speed and logic complexity
    - For memory instructions, CPI depends on memory access latency
- **Number of Instructions (N)**
  - Count of *dynamic* instructions required to complete a program
    - Depends on the instruction set architecture
  - Not to be confused with code size (no. of instructions in a source program)
    - For example, 8-instruction loop executing 5 times counted as 40 instructions

How to Increase Performance?

- **Faster logic and memory**
  - Design faster implementations of arithmetic and logic circuits
  - Use caches to reduce average latency of memory operations

- **Increase parallelism**
  - Perform multiple instructions in parallel (*instruction-level parallelism*)
    - Fetch next instruction while ALU executes previous instruction (*pipelining*)
    - Use multiple ALUs to execute multiple instrs. concurrently (*superscalar*)
    - Result: CPI is no longer equal to the number of clock cycles required to complete one instruction (CPI can become < 1)
  - Execute multiple programs at the same time (*thread-level parallelism*)
    - Multiple cores on a single die (*multi*/*many*-core processors)

How to Increase Performance? (cont.)

- **Increase processor's clock speed**
  - Advances in manufacturing technology
    - Faster transistors => Faster circuits => Less time needed per clock cycle
    - All operations become faster except memory access latency
  - Reducing the amount of processing done in each step (*deeper pipeline*)
    - This may increase CPI, unless there is instruction-level-parallelism
- **Better compiler technology**
  - Converts a high-level program into least expensive set of machine instructions with the goal of reducing *N*CPI*
  - Optimized according to a specific processor architecture

How to Increase Performance? (cont.)

- **Choice of Instruction Set Architecture (ISA)**
  - Tradeoff between *N*, *CPI* and/or *R*
  - ***C***omplex ***I***nstruction ***S***et ***C***omputers (CISC)
    - Fewer, more complex instructions per program (lower *N*) but instructions take longer to execute (higher *CPI* and/or lower clock rate)
  - ***R***educed ***I***nstruction ***S***et ***C***omputers (RISC)
    - Simpler instructions, more needed to perform the same task (higher *N*) but instructions quicker to execute (lower *CPI* and/or higher clock rate)
  - We'll revisit this topic later in the course

Performance Measurement and Comparison

- **Speedup** of a computer C2 over C1 is equal to **T1/T2** where T1 and T2 are execution times of a program for computers C1 and C2

- Computer performance typically measured on benchmark programs
- **S**ystem **P**erformance **E**valuation **C**orporation (SPEC) selects and publishes representative benchmarks for different application domains and testing results for commercially available computers.

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

Performance Example

- A program is executed on two processors P1 and P2. P1 does not have a cache while P2 has an on-chip cache with 10 cycle latency. The program has 500 instructions, including a 100-instruction loop that is executed 25 times. Both the processors have the same clock rate of 1 GHz and use the same instruction set. Each access to the off-chip memory requires 100 cycles. Calculate the speedup of P2 over P1. You can make the following assumptions: (i) Execution time is dominated by the time needed to fetch instructions from memory. Time required for data access and computations is ignored. (ii) P2's cache is initially empty and can store the entire loop after first loop iteration.
- **Solution:**

  For P1,

  - Number of instructions (N) = 400 + (100*25) = 2900
  - Cycles per instruction (CPI) = 100
  - Clock Rate (R) = 1 GHz = 10^9 cycles per second
  - Execution Time T1 = (2900*100)/ 10^9 = 290 microseconds

Performance Example (cont.)

- A program is executed on two processors P1 and P2. P1 does not have a cache while P2 has an on-chip cache with 10 cycle latency. The program has 500 instructions, including a 100-instruction loop that is executed 25 times. Both the processors have the same clock rate of 1 GHz and use the same instruction set. Each access to the off-chip memory requires 100 cycles. Calculate the speedup of P2 over P1. You can make the following assumptions: (i) Execution time is dominated by the time needed to fetch instructions from memory. Time required for data access and computations is ignored. (ii) P2's cache is initially empty and can store the entire loop after first loop iteration.

  Solution (cont.):

  For P2,

  - Number of instructions (N) = 400 + (100*25) = 2900
  - Cycles per instruction (CPI) = ((500*100)+(2400*10))/2900 = 25.52
  - Clock Rate (R) = 1 GHz = 10^9 cycles per second
  - Execution Time T2 = (2900*25.52)/ 10^9 = 74 microseconds

  Speedup of P2 over P1 = T1/T2 = 290/74 = 3.92

Practice Problem

- Two computers P3 and P4 are being used to solve a fluid dynamics problem. P3 uses a RISC processor with a clock speed of 3 GHz while P4 uses a CISC processor with a clock speed of 2 GHz. Solving the fluid dynamics problem involves two computation steps: (i) The first step requires 10000 instructions on each computer, having the same latency of 1 cycle. (ii) The second step requires 15000 1-cycle instructions on P3 and 1000 10-cycle instructions on P4. Which computer has the higher performance and by how much?

Practice Problem (cont.)

- Two computers P3 and P4 are being used to solve a fluid dynamics problem. P3 uses a RISC processor with a clock speed of 3 GHz while P4 uses a CISC processor with a clock speed of 2 GHz. Solving the fluid dynamics problem involves two computation steps: (i) The first step requires 10000 instructions on each computer, having the same latency of 1 cycle. (ii) The second step requires 15000 1-cycle instructions on P3 and 1000 10-cycle instructions on P4. Which computer has the higher performance and by how much?
- **Solution:**

  For P3,

  - Number of instructions (N) = 10000 + 15000 = 25000
  - Cycles per instruction (CPI) = ((10000*1)+(15000*1))/25000 = 1
  - Clock Rate (R) = 3 GHz = $3*10^9$ cycles per second
  - Execution Time T3 = (25000*1)/ ($3*10^9$) = 8.33 microseconds

Practice Problem (cont.)

- Two computers P3 and P4 are being used to solve a fluid dynamics problem. P3 uses a RISC processor with a clock speed of 3 GHz while P4 uses a CISC processor with a clock speed of 2 GHz. Solving the fluid dynamics problem involves two computation steps: (i) The first step requires 10000 instructions on each computer, having the same latency of 1 cycle. (ii) The second step requires 15000 1-cycle instructions on P3 and 1000 10-cycle instructions on P4. Which computer has the higher performance and by how much?

  For P4,

  - Number of instructions (N) = 10000 + 1000 = 11000
  - Cycles per instruction (CPI) = ((10000*1)+(1000*10))/11000 = 1.82
  - Clock Rate (R) = 2 GHz = $2*10^9$ cycles per second
  - Execution Time T4 = (11000*1.82) / ($2*10^9$) = 10 microseconds

  P3 is faster because T3 is less than T4

  Speedup of P3 over P4 = T4/T3 = 10/8.33 = 1.2

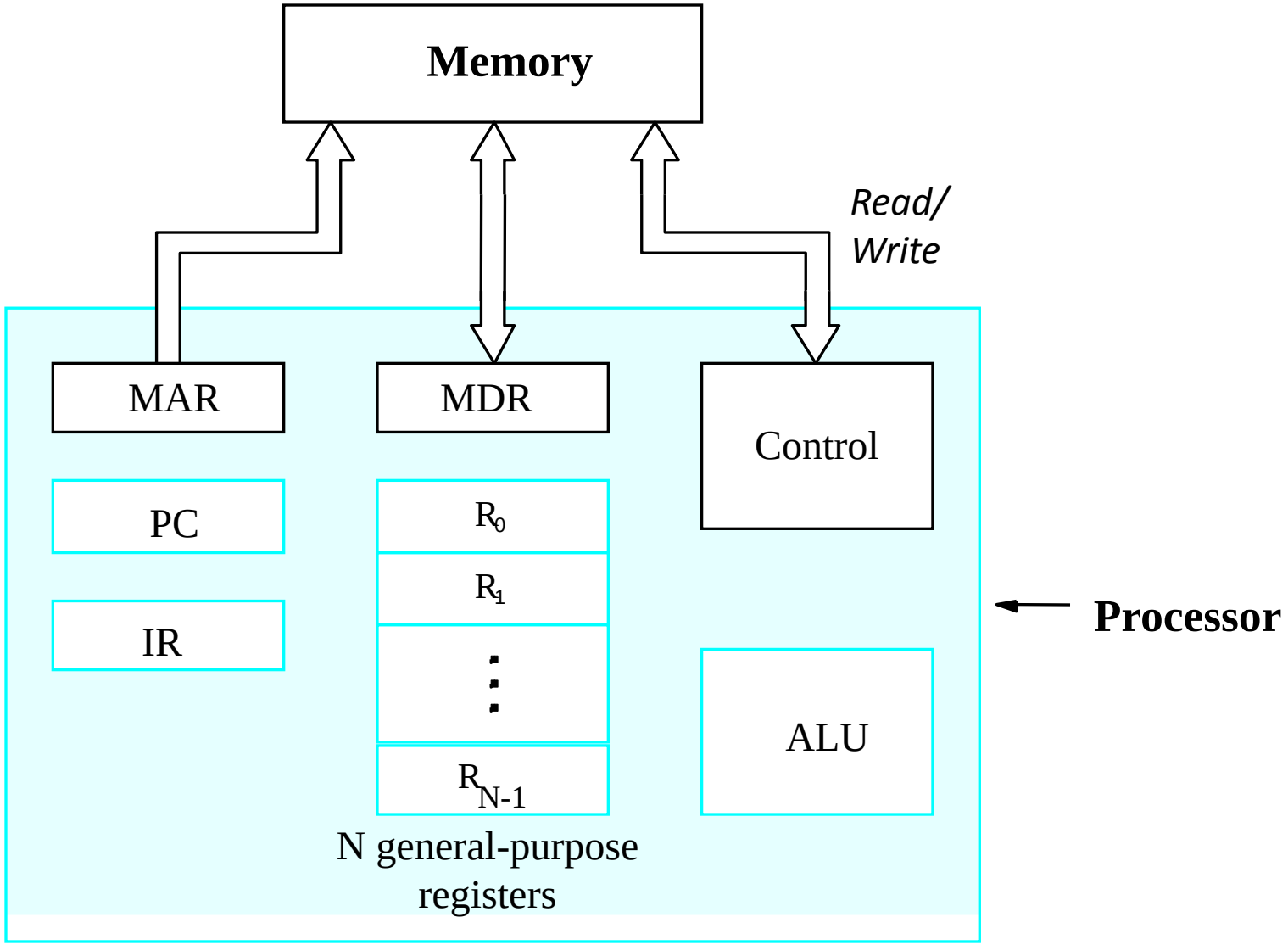
BASIC PROCESSING UNIT

Overview

- A typical computing task consists of a a sequence of *instructions* that constitute a *program*
- Processing Unit (*processor*) carries out these instructions
- An instruction is *processed* by performing a sequence of more rudimentary operations, such as:
  - *Fetching* the instruction from memory
  - *Decoding* the instruction to understand its semantics
  - *Executing* the instruction
- Different processor architectures differ in the way they perform the above operations

We'll focus on architectural principles and hardware components that are common to (almost) all the existing processors

Processor Organization



Processor Registers

- Instruction register (IR)
  - Holds the instruction that is currently being executed
- Program counter (PC)
  - Contains address of the next instruction to be fetched/executed
- General-purpose register ($R_0 – R_{n-1}$)
  - Hold operands that are currently being (or soon to be) processed
- Memory address register (MAR)
  - Contains address of the word being read/written from memory
- Memory data register (MDR)
  - Contains contents of the data word being read/written from memory

Fundamental Concepts

- Processor fetches Instructions from successive memory addresses until a branch/jump instruction is encountered
- Processor keeps track of the address of memory location containing the next instruction in a register called **program counter (*PC*)**
- After every instruction fetch, the **instruction address generator** updates the contents of *PC* to point to the next instruction
- While being processed, the instruction is kept in **instruction register (*IR*)**. *IR* holds the instruction until its execution is completed
- **Control circuitry** decodes (interprets) the instruction in *IR* and orchestrates the necessary control signals and inputs needed to execute the instruction

Fundamental Concepts (cont.)

- **Register file** is a fast memory whose storage locations are organized to form the processor's general purpose registers
- The contents of the registers named in an instruction are sent to the **Arithmetic and Logic Unit (ALU)**, which executes the instruction
- The results computed by the ALU are stored in a register in the register file
- The processor communicates with the memory through the **processor memory interface**

ISA and Register Transfer Notation

Instruction set Architecture (ISA) specifies the operations that a computer is able to perform. ISA must include four types of operations:

1. Data transfers between memory and processor registers (load, store)
2. Arithmetic and logic operations on data (add, mult, and, or etc.)
3. Program sequencing and control (branch, jump)
4. I/O transfers

Register Transfer Notation (RTN) describes how each operation transfers data

- The contents of a location are denoted by placing square brackets around its name
- Right hand side of a RTN expression always denotes a value
- Left hand side of a RTN is the name of a location where the value is to be placed
- Examples:
  - R2 ← [LOC] means that the contents of memory location LOC are transferred into register R2
  - R4 ← [R2] + [R3] means that the contents of registers R2 and R3 are added and the result is transferred into the register R4