Computer Organization and Application

Lecture 3

Von Neuman architecture and its components

Dr Mohammed Fadhil
Email: mohammed.fadhil1@uomus.edu.iq

Learning Objectives

- Understand the origin of the Von Neumann Architecture
- Understand Von Neumann Architecture
- Understand its components and functionalities
- To be aware of other architectures

Basic Components of a Computer

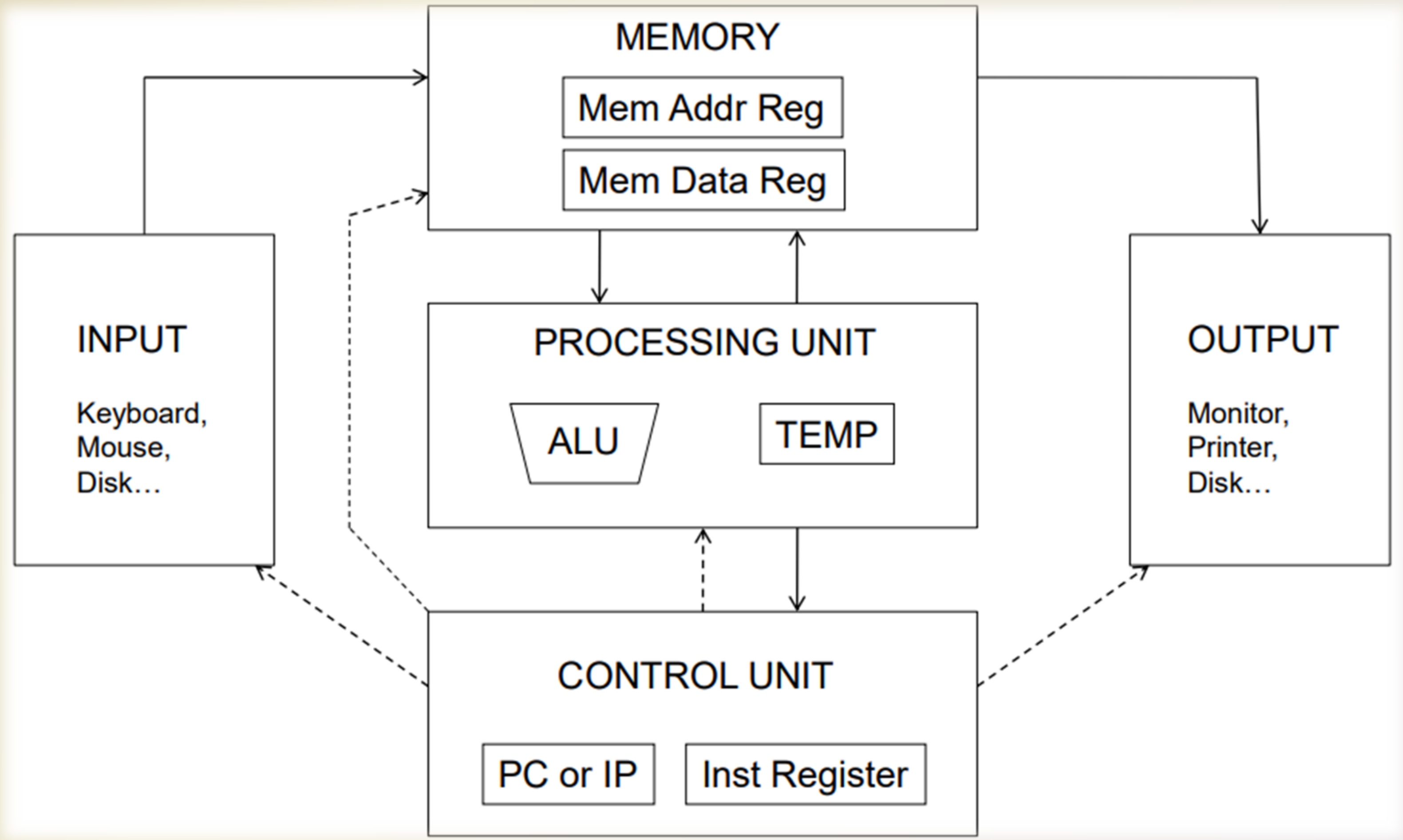
- To get a task done by a (general-purpose) computer, we need:
  - **A computer program**
    - That specifies what the computer must do
  - **The computer itself**
    - To carry out the specified task
  - **Program**: A set of instructions
    - Each instruction specifies a well-defined piece of work for the computer to carry out
    - Instruction: the smallest piece of specified work in a program
  - **Instruction set**: All possible instructions that a computer is designed to be able to carry out

The von Neumann Model

- In order to build a computer, **we need an execution model for processing computer programs**
- John von Neumann proposed **a fundamental model** in 1946
- The von Neumann Model consists of 5 components
  - **Memory** (stores the program and data)
  - **Processing unit**
  - **Input**
  - **Output**
  - **Control unit** (controls the order in which instructions are carried out)
- Throughout this lecture, we will examine two examples of the von Neumann model
  - ❑ **LC-3**
  - ❑ **MIPS**

All general-purpose computers today use the von Neumann model

The von Neumann Model



Memory

- **Memory stores**
  - Programs
  - Data
- **Memory contains bits**
  - Bits are logically grouped into bytes (8 bits) and words (e.g., 8, 16, 32 bits)
- **Address space:** Total number of uniquely identifiable locations in memory
  - In LC-3, the address space is 2^16
    - **16-bit addresses**
  - In MIPS, the address space is 2^32
    - **32-bit addresses**
  - In x86-64, the address space is (up to) 2^48
    - **48-bit addresses**
- **Addressability**: How many bits are stored in each location (address)
  - E.g., 8-bit addressable (or byte-addressable)
  - E.g., word-addressable
  - A given instruction can operate on a byte or a word

A Simple Example

- **A representation of memory with 8 locations**
- **Each location contains 8 bits (one byte)**
  - Byte addressable memory; address space of 8
  - Value 6 is stored in address 4 & value 4 is stored in address 6

Address	Data Value
000	
001	
010	
011	
100	00000110
101	
110	00000100
111	

Question: How can we make same-size memory bit addressable?

64 locations
Each location stores 1 bit

Word-Addressable Memory

- **Each data word has a unique address**
  - In **MIPS**, a unique address for each **32-bit data word**
  - In **LC-3**, a unique address for each **16-bit data word**

MIPS memory

Word Address	Data	
⋮	⋮	⋮
00000003	D 1 6 1 7 A 1 C	Word 3
00000002	1 3 C 8 1 7 5 5	Word 2
00000001	F 2 F 1 F 0 F 7	Word 1
00000000	8 9 A B C D E F	Word 0

Byte-Addressable Memory

- **Each byte has a unique address**
  - **32-bit word = 4 byte,** so word address increments by **4**
  - **MIPS** is actually **byte-addressable**
  - **LC-3**, (updated version of LC-3) is also **byte-addressable**

MIPS memory

Byte Address of the Word	Data				
.	.				.
.	.				.
.	.				.
0000000C	D 1	6 1	7 A	1 C	Word 3
00000008	1 3	C 8	1 7	5 5	Word 2
00000004	F 2	F 1	F 0	F 7	Word 1
00000000	How are these four bytes ordered?				Word 0

Which of the four bytes is most vs. least significant?

Big Endian vs. Little Endian

- How to number bytes within a word?
- Little-endian: byte numbers start at the little (least significant) end.
- Big-endian: byte numbers start at the big (most significant) end.
- Word address is the same for big- or little-endian.

- Jonathan Swift's Gulliver's Travels
  - **Big Endians** broke their eggs on the big end of the egg.
  - **Little Endians** broke their eggs on the little end of the egg.
- It doesn't really matter which addressing type used – except when the two systems need to share data!

Accessing Memory: MAR and MDR

- There are two ways of **accessing memory**
  - **Reading** or **loading** data **from** a memory location
  - **Writing** or **storing** data **to** a memory location
- **Two registers** are usually used to access memory
  - Memory Address Register (**MAR**)
  - Memory Data Register (**MDR**)
- **To read**
  - Step 1: Load the **MAR with the address** we wish to read from
  - Step 2: **Data in the corresponding** location gets placed in MDR
- **To write**
  - Step 1: Load the **MAR with the address** and the **MDR with the data** we wish to write
  - Step 2: Activate **Write Enable** signal → value in MDR is written to address specified by MAR

Processing Unit

- Performs the actual computation(s)
- The processing unit can consist of many **functional units**
- We start with a simple **Arithmetic and Logic Unit (ALU)**, which executes computation and logic operations
  - **LC-3**: ADD, AND, NOT (XOR in LC-3b)
  - **MIPS**: add, sub, mult, and, nor, sll, slr, slt…
- The ALU processes quantities that are referred to as **words**
  - **Word length** in LC-3 is 16 bits
  - **Word length** in MIPS is 32 bits

Processing Unit: Fast Temporary Storage

- It is almost always the case that a computer provides a small amount of storage very close to ALU
  - **Purpose: to store temporary values and quickly access them later**
- E.g., to calculate ((A+B)*C)/D, the intermediate result of A+B can be stored in temporary storage
  - Why? It is too slow to store each ALU result in memory & then retrieve it again for future use
    - **A memory access is much slower than an addition, multiplication or division**
  - Ditto for the intermediate result of ((A+B)*C)
  - This temporary storage is usually a set of registers
    - Called **Register File**

Registers: Fast Temporary Storage

- **Memory** is large but slow
- **Registers in the Processing Unit**
  - Ensure fast access to values to be processed in the ALU
  - Typically one register contains **one word (same as word length)**
- **Register Set or Register File**
  - **Set of registers that can be manipulated by instructions**
  - LC-3 has **8 general purpose registers (GPRs)**
    - **R0 to R7**: 3-bit register number
    - **Register size** = Word length = 16 bits
  - MIPS has 32 general purpose registers
    - **R0 to R31**: 5-bit register number (or Register ID)
    - **Register size** = Word length = 32 bits

Input and Output

- Enable information to get into and out of a computer
- Many devices can be used for input and output
- They are called peripherals
  - **Input**
    - Keyboard ■ Mouse ■ Scanner ■ Disks ■ Etc.
  - **Output**
    - ■ Monitor ■ Printer ■ Disks ■ Etc.
- In LC-3, we consider keyboard and monitor

Control Unit

- **The control unit** is like the conductor of an orchestra
- It conducts the step-by-step process of executing (every instruction in) a program (in sequence)
- It keeps track of which instruction being processed, via
  - **Instruction Register (IR)**, which contains the instruction
- It also keeps track of which instruction to process next, via
  - **Program Counter (PC) or Instruction Pointer (IP)**, another register that contains the address of the (next) instruction to process

THANK YOU ☺