Computer Organization and Assembly Language

Introduction

Credits: Slides prepared by several faculty members who previously taught this course

About Me

Dr. Ammar Haider

Assistant Professor

Department of Computer Science

ammar.haider@nu.edu.pk

Office: Library Block, 1st floor

What is the course about?

- This course is about looking at lower level details of how computers works.
- In order to do so, we will see computer organization and architecture.
- You will also learn assembly language for Intel 8088 processor.
- At this point you have taken a course on High level language (C++), so its good time to get started looking at
  - How processor (CPU) is organized?
  - How program is executed on processor?
  - How processor interacts with memory and I/O devices?

Books

- Assembly Language Programming Lecture Notes by Belal Hashmi (BH).
- Assembly Language for x86 Processors Seventh Edition Kip R. Irvine (KI)
- Computer Organization and Architecture Designing for Performance Tenth Edition by William Stallings (WS)

Tentative Evaluation and Grading Policy

- You will be evaluated on quizzes, 2 mid term exams, 1 final exam and multiple assignments.

- Tentative Percentage Grade Distribution:

  - QUIZZES 10
  - MIDTERMS 30
  - FINAL 45
  - ASSIGNMENTS 15

Academic Integrity

- Plagiarism and Cheating against academic integrity. Both parties involved in such cases will face strict penalty (negative marking, F grade, DC)
- CODE/ ASSIGNMENT SHARING is strictly prohibited.
- Keep in mind that by sharing your code/assignment you are not helping anyone rather hindering the learning process or the other person.
- No excuse will be entertained if your work is stolen or lost. To avoid such incidents
  - Keep back up of your code on safe online storage, such as Google Drive, Drop box or One drive.
  - Do not leave your work on university lab computer, transfer your work to online storage and delete from the university lab computer (empty recycle bin as well)

Question

- What specifications do you look at when you see a computer (lets say a laptop)?

Some Definitions

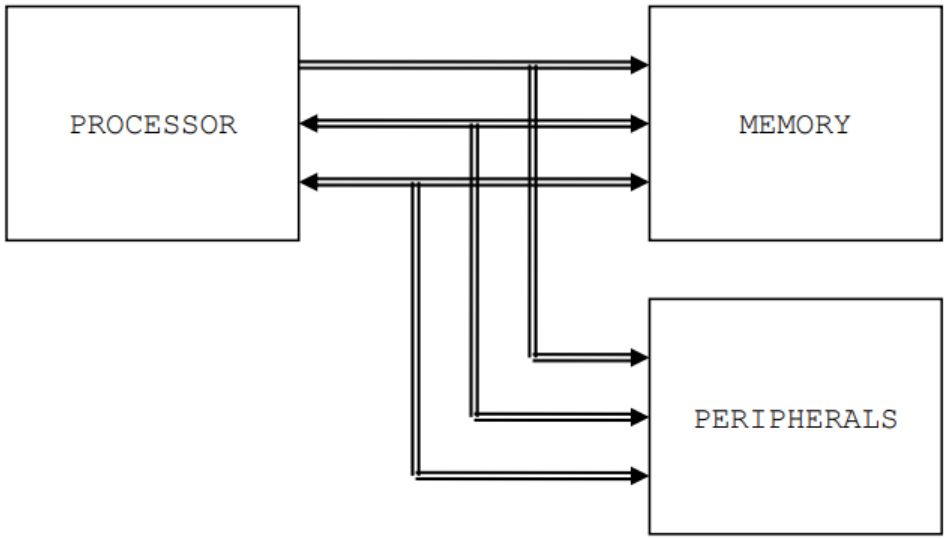
- Computer Architecture
  - The **architecture** of a computer is a logical description of its components and its basic operations.

- Computer Organization
  - Computer organization refers to the operational units and their interconnections that realize the architectural specifications

- For example, it is an architectural design issue whether a computer will have a multiply instruction. It is an organizational issue whether that instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system

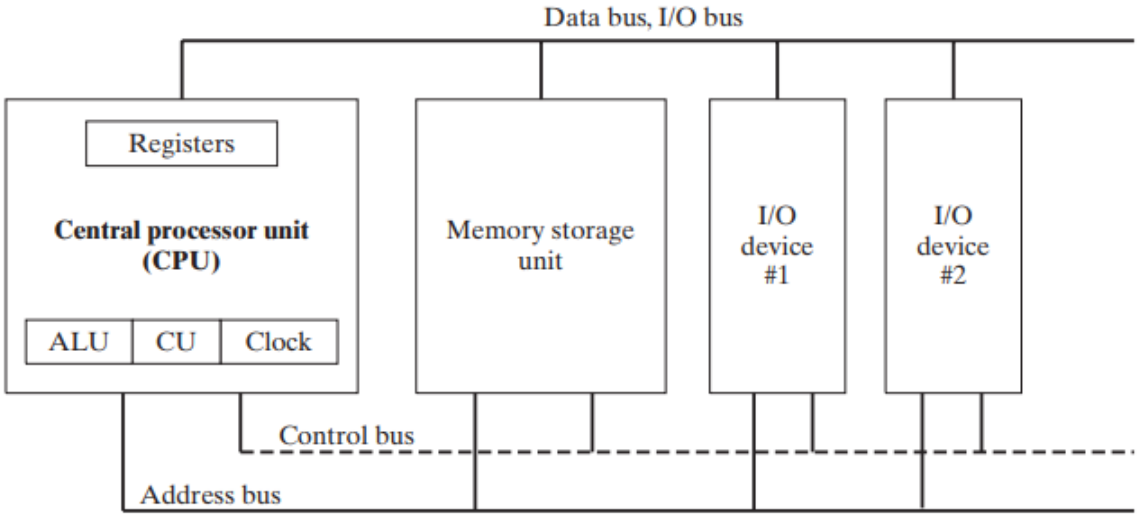
Basic Components of Computer

- Section 1.1 BH



Another diagram

FIGURE 2–1 Block diagram of a microcomputer.

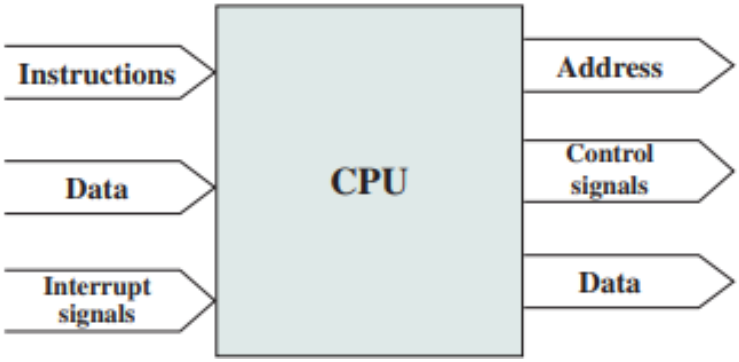


Source KI

ALU: Arithmetic and Logic Unit
CU: Control Unit
Registers: Data storage cells

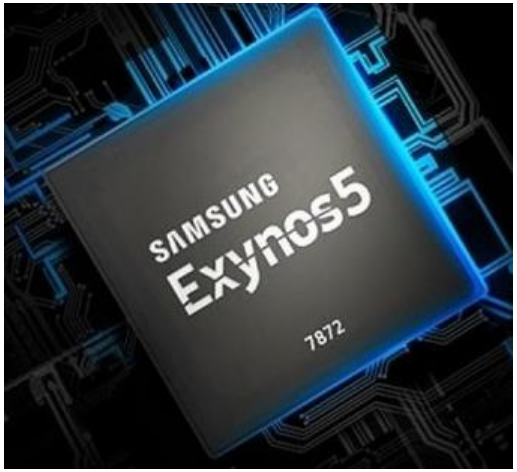
Processor

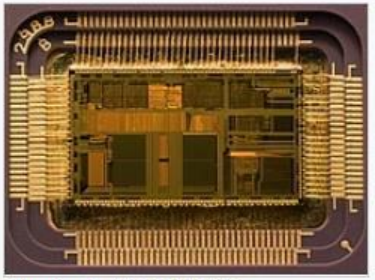
- Processor: The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system. It also receives interrupt signals



- ALU, CU & Registers are 3 distinct CPU elements

Some Processors



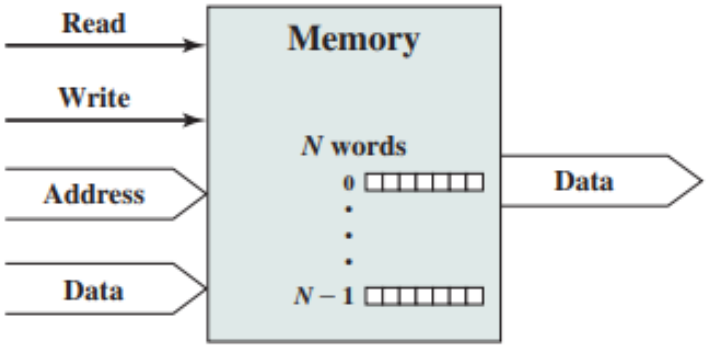


Die of an Intel 80486DX2 microprocessor (actual size: 12 × 6.75 mm) in its packaging

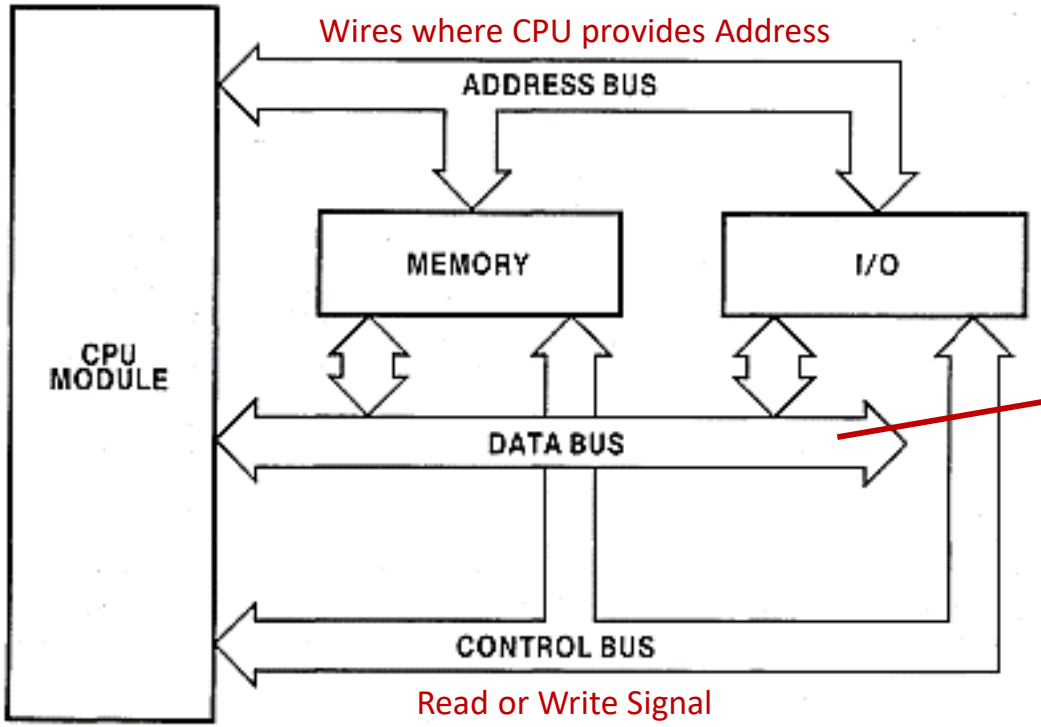
Intel Core i5 CPU on a Vaio E series laptop motherboard (on the right, beneath the heat pipe)

Memory

- Memory: Typically, a memory module will consist of N words of equal length. Each word is assigned a unique numerical address (0, 1, 2, … , N-1). A word of data can be read from or written into the memory. The nature of the operation is indicated by read and write control signals. The location for the operation is specified by an address.



Another block diagram of computer



Wires where data travels for read and write

Buses

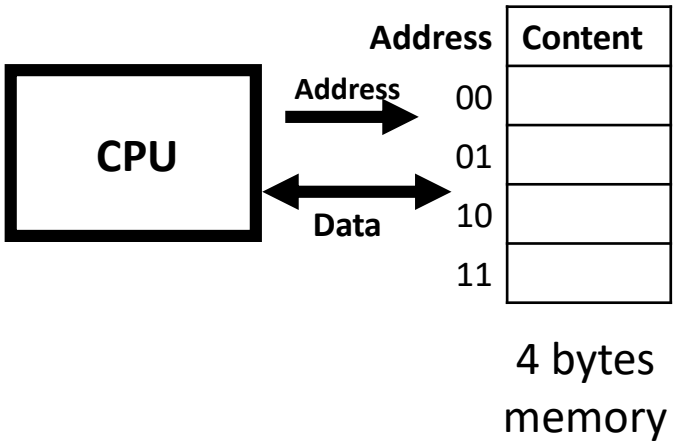
- A bus is a communication pathway connecting two or more devices. For example Processor to/from Memory, I/O to/from processor, I/O to/from Memory
- The group of bits that the processor uses to inform the memory about which element to read or write is collectively known as the **address bus**.
  - **Unidirectional** – Address always travels from processor to memory

Buses

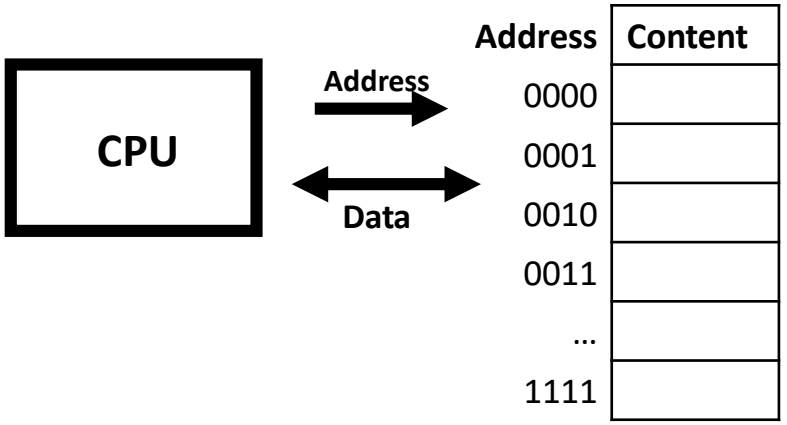
- Another important bus called the **data bus** is used to move the data from the memory or I/O device to the processor in a read operation and from the processor to the memory or I/O device in a write operation.
  - **Bidirectional**: the data can either be instructions for the CPU, or information the CPU is passing to or from I/O ports
- The third group consists of miscellaneous independent lines used for control purposes. For example, one line of the bus is used to inform the memory about whether to do the read operation or the write operation. These lines are collectively known as the **control bus**

Address Bus

- How Processor will read/write one cell out of 4 bytes memory?
- How Processor will read/write one cell out of 16 bytes memory?
- How Processor will read/write one cell out of 1MB memory?



4 bytes memory



16 bytes memory

What is the size of each Address Bus?

Address Buses

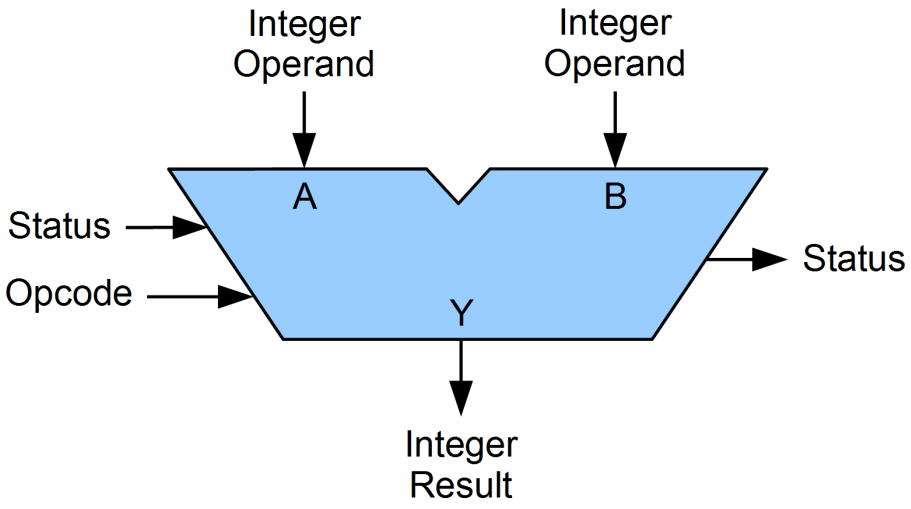
- The number of address bus lines varies from one microcomputer to another.
- The **width of the address bus** determines the (maximum) amount of memory a system can address.
- For example, a system with a 32-bit address bus can use at most 2^{32} memory locations.
  - If each memory location holds one byte, the addressable memory space is 4 GB ($G=2^{30}$).

A word about registers

- Registers are named storage locations in the CPU that hold intermediate results of operations.
- As they are located inside CPU they are fastest to access.
- Each processor has a limited number of registers.

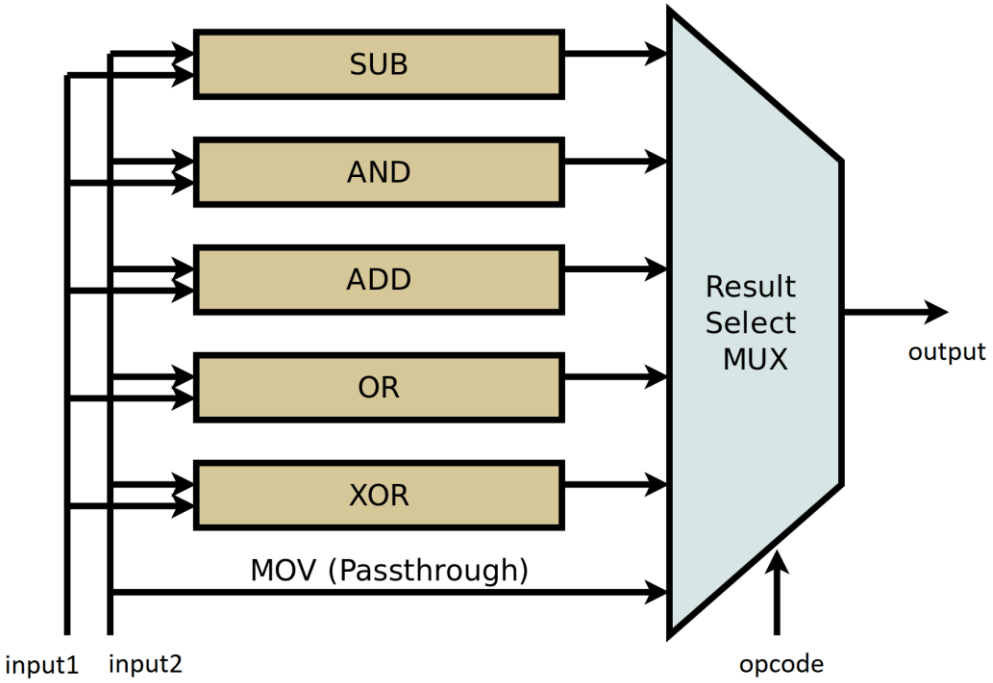
ALU

- Arithmetic Logic Unit (ALU) performs arithmetic and data-processing operations as specified by program



Inside the ALU

- An ALU calculates **all** possible operations in parallel. A multiplexer (MUX) chooses any one output based on instruction operation code (opcode)



Word size of CPU

- The ALU in each CPU is designed to process data of a fixed size – it is called the computer's **word size**.
- Typical word sizes are 8 bits, 16 bits, 32 bits and 64 bits.
- Most of the registers in CPU will also be word sized.
- Question
  - What is the word size in your laptop CPU?
  - What about the one in your smartphone?

How instructions are executed

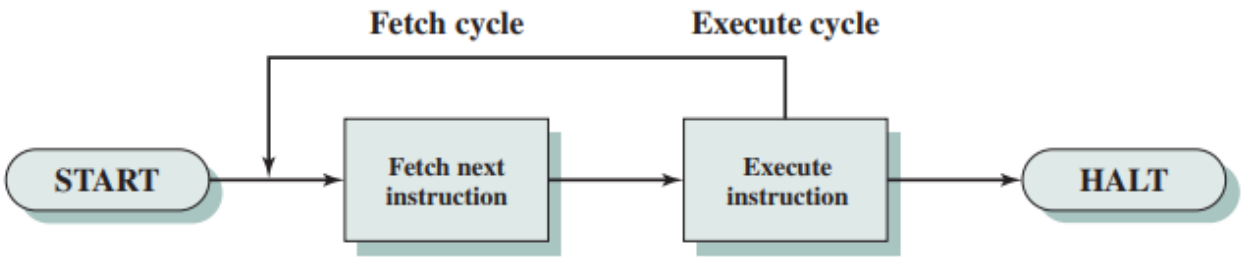


Figure 3.3 Basic Instruction Cycle

Where is Assembly Language?

- Up until now, the instruction you provided to your computer were in High level language, for example code in C++.
- But processor doesn't understand these high level languages. It can only understand binary 0s and 1s. This is called machine Language.
- Example of machine instruction to add two registers is

```
00000010 00110010 01000000 00100000
```

- Not very meaningful to us humans, hard to remember, and hard to distinguish between multiple instructions

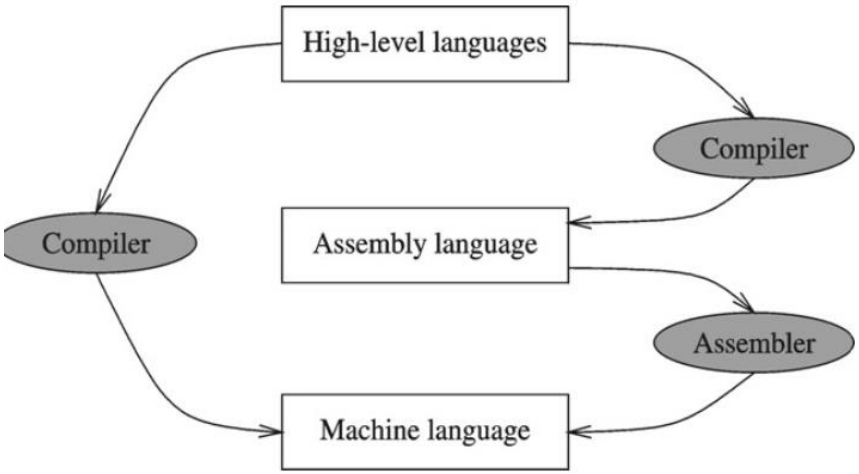
Still, where is Assembly Language?

- There are many different types of processors out there, and so many high level programming languages (HLL) are well.
- The processors don’t provide a direct way to convert each HLL to machine language.
- But each processor comes with its own assembly language, which is much like human readable machine language.
- So assembly language lies between HLL and machine language.
- Example of assembly language instruction to add two registers is

```
add R8, R17, R18
```

Compiler and Assembler

- Some compilers will first convert the code to assembly language and then to machine language (or object code)
- Some compilers will directly convert the code to machine language (or object code)
- The object code generated is for a particular type of processor.



Example:

- add.cpp file

```
gcc -S -o add.s add.cpp
```

- The above command will save the assembler source code of add.cpp file in add.s file

Why need Assembly language?

- To better understand the processor.
  - An assembly language program describes exactly what the hardware should do, step by step, in terms of the basic operations of the hardware. In a high level programming language like C or Java a programmer is mostly unaware of computer architecture.
  - This can also help in code optimization in terms of space and time.
- In some processors it might not be feasible to code in high level languages.
  - Why?

Intel 8088 Processor

- There is not just one language called "assembly language.“ Each processor family (such as Intel, MIPS, ARM, Alpha, ...) has its own architecture, organization and machine instructions and a corresponding assembly language.
- Learning about one processor and its assembly language is good enough to know how things work.
- In this course we will work with Intel 8088 16-bit processor, which is a part of Intel x86 family of processors.
- You will not work with the real processor but with emulator ***DosBox***
- Other tools you will need are assembler and debugger

Reading and References

- Reading
  - Section 1.1 Kip Irvine
  - Section 1.1 Belal Hashmi

- References
  - Chapter 1 KI
  - Chapter 3 WS
  - Chapter 1 BH
  - Chapter 1 https://chortle.ccsu.edu/assemblytutorial/index.html