



The Islamic University of Gaza
Faculty of Engineering
Department of Computer Engineering



LCOM 3010 - Computer Architecture

Lab 1

Introduction to Computer Architecture

Prof. Wesam M. Ashour

Eng. Hashem M. Hejazy

Summer 2022/2023

Computer component:

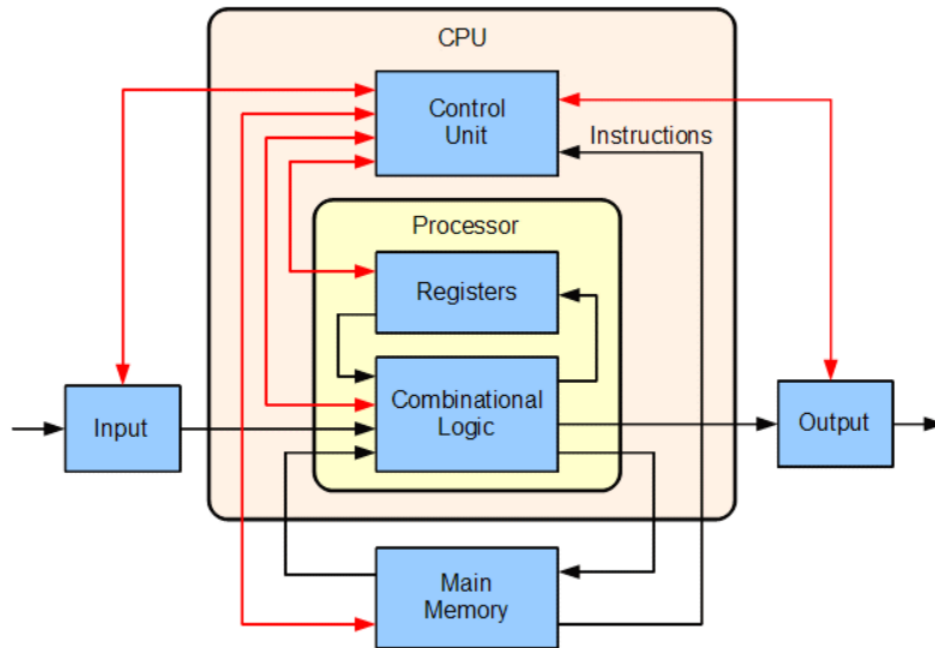
There are several components that make up a computer, including:

1. **Central Processing Unit (CPU):** This is the brain of the computer and is responsible for processing instructions and performing calculations.
2. **Random Access Memory (RAM):** This is the temporary memory that the computer uses to store data and instructions while it is running.
3. **Hard Disk Drive (HDD) or Solid State Drive (SSD):** This is the permanent storage device that stores all of the computer's data and programs.
4. **Motherboard:** This is the main circuit board that connects all of the computer's components together and provides power and communication between them.
5. **Power Supply Unit (PSU):** This provides power to the computer's components and ensures that they receive the correct amount of electricity.
6. **Graphics Processing Unit (GPU):** This is responsible for rendering images and video on the computer's display.
7. **Sound Card:** This is responsible for processing audio signals and outputting them to speakers or headphones.
8. **Input Devices:** These are devices that allow the user to input data into the computer, such as a keyboard, mouse, or touchscreen.
9. **Output Devices:** These are devices that allow the computer to output data, such as a monitor, printer, or speakers.
10. **Cooling System:** This is responsible for keeping the computer's components cool and preventing them from overheating.

Central Processing Unit (CPU):

The CPU, or Central Processing Unit, is the primary component of a computer that is responsible for executing instructions and performing calculations. It is often referred to as the "brain" of the computer, since it controls and directs the operations of the computer's hardware and software.

Structure of a CPU:



- **Control Unit:** This unit communicates with both arithmetic logic unit, Memory unit as well as the input and output devices. It does not execute any program instructions, but it starts other unit of a CPU by providing required electric signals to execute instruction set of a program.
- **Registers Unit:** It hold instructions during the execution of a program. It stores and supply operands to the ALU during the execution of an operation.
- **Combinational logic unit:** This unit is a combination of all the logical unit like Arithmetic Logic Unit(ALU), Floating-point Unit (FPU) etc. ALU performs the Integer arithmetic and bitwise logic operations and FPU performs all the mathematical operations on floating point variables.
- **Memory management Unit:** This unit converts logically memory addresses to physical RAM addresses. It also acts as virtual memory when required for any CPU operation.

Operations of CPU:

The fundamental operations a processor do are same among all processors irrespective of their version, cost or the different technologies they are using. It executes programs and a program is nothing but a set of instructions or commands. Processors process the instructions in three steps. Those are fetch, decode and execute.

1. **Fetch:** The instructions written in a program is in a state of numbers or a sequence of numbers present in program memory called program counter(PC). PC stores addresses/locations of all the instructions set in a sequence. In first step CPU fetch instruction by determining the address defined by PC. After fetching the first instruction, PC(The memory address) increments by the length of the fetched instruction so that it will point to the next instruction. This way CPU fetch all the instructions of a program one by one.
2. **Decode:** There is a circuitry called instruction decoder in every processor. It converts the instructions to electric signal that is send to other circuitry of CPU for further operations.
3. **Execute:** After fetch and decode, execute step is performed. This may be a single process or multiple, all parts of CPU is connected with each other to perform the required operation.

Multi-core processor:

Modern CPU is of multiple cores. We have heard terms like Dual core processor, Quad core or Octa core processor. Multi-core processor is actually a physical unit that contains more than one individual processor. So Dual core processor contains two processors and similarly Quad core contains four and octa core contains 8 processors.

Multi-Core enables hardware threading. Means if one processor is executing one program then another new program that is started by user does not have to wait for the completion of currently executing program, because another processor of multi-core CPU will be allotted to execute second program. This way the execution and operation speed of a computer is increases and a computer can do more task in less time.

Computer Languages:

- **High-level language:** A high-level language is a programming language that is designed to be easy to read and write, and is closer to human language than machine language. High-level languages are often used for rapid application development and are generally more productive than low-level languages. Examples: python, java, PHP and so more.

- **Low-level language:** A low-level language is a programming language that is closer to machine language than human language. Low-level languages are often used for system programming and other tasks that require direct manipulation of hardware. Example: Assembly Language.

Difference between Machine language and Assembly Language

Machine language (computer's native language) is a system of impartible instructions executed directly by a computer's central processing unit (CPU).

1. Instructions consist of binary code: 1's and 0's.
2. Machine language can be made directly from java code using interpreter.
3. Ex: 10011010

Assembly Language is a programming language that is very similar to machine language but uses symbols instead of binary numbers. It is converted by the assembler (e.g., Tasm and Masm) into executable machine-language programs. Ex: ADD R2, R1, R0.

Introduction to ARM Architecture

What do we mean by architecture?

When we use the term architecture, we mean a functional specification. In the case of the Arm architecture, we mean a functional specification for a processor. An architecture specifies how a processor will behave, for example what instructions it has and what the instructions do.

ARM Instruction Set

❖ Arithmetic Instructions

$\langle \text{operation} \rangle \{ \text{cond} \} \{ S \} \text{ Rd, Rn, Operand2}$

1. ADD (Add)
 $\text{Rd} := \text{Rn} + \text{Operand2}$
2. ADC (Add with Carry)
 $\text{Rd} := \text{Rn} + \text{Operand2} + \text{Carry}$
3. SUB (Subtract)
 $\text{Rd} := \text{Rn} - \text{Operand2}$

❖ Logical Instructions

$\langle \text{operation} \rangle \{ \text{cond} \} \{ S \} \text{ Rd, Rn, Operand2}$

1. AND (logical AND)

$\text{Rd} := \text{Rn AND Operand2}$

2. ORR (logical OR)

$\text{Rd} := \text{Rn OR Operand2}$

❖ Branch Instructions

$\langle \text{operation} \rangle \{ \text{cond} \} \langle \text{address} \rangle$

1. B (Branch)

$\text{PC} := \langle \text{address} \rangle$

Flags in ARM CPU

In ARM CPU, the flag register is a special-purpose register that contains status flags that reflect the outcome of the most recent arithmetic or logic operation. These flags are also sometimes referred to as condition code flags.

The flag register in ARM CPU typically includes the following flags:

- Negative (N): This flag is set if the result of an operation is negative.
- Zero (Z): This flag is set if the result of an operation is zero.
- Carry (C): This flag is set if an arithmetic operation generates a carry out of the most significant bit.
- Overflow (V): This flag is set if an arithmetic operation generates an overflow of the signed result.

These flags are used in conditional execution of instructions, which allows the program to execute different instructions based on the outcome of a previous operation. For example, a conditional branch instruction may be used to jump to a different location in the program based on the value of the Zero flag.

Example: add two numbers using ARM assembly (10 + 20)

Solution:

MOV R0, #10

MOV R1, #20

ADD R2, R1, R0

stop B stop

Task #1: Convert C code to assembly

```
int total;
int i;

total = 0;
for (i = 10; i > 0; i--) {
    total += i;
}
```

Solution: Hint: you need to set flags (using S after operation)

```
                MOV  R0, #0           ; R0 accumulates total
                MOV  R1, #10          ; R1 counts from 10 down to 1
again          ADD  R0, R0, R1
                SUBS R1, R1, #1
                BNE  again
halt          B     halt              ; infinite loop to stop computation
```