# ECET330 Lab 1 Part 2 Procedures

# Title: Introduction to Assembly and Machine Language

# I. OBJECTIVES

1. To become familiar with the CPU function
2. Understanding machine language
3. Understanding assembly language

## II. PARTS LIST

###### Equipment: IBM PC or compatible with Windows 2000 or higher

# INTRODUCTION

The Central Processing Unit (CPU) executes each instruction of a program one at a time. Each instruction has an operation field which describes the operation such as add or subtract that should be performed on the data. The CPU’s hardware is designed to realize binary codes.

**Machine language**: The binary language in which instructions are binary codes and the machine (CPU) understands them.

In the simple CPU design of Lecture 1, a set of operations is defined and a binary code was assigned to each operation. The following table shows the table defined in Lecture 1.

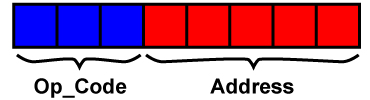
|  |  |  |
| --- | --- | --- |
| Operation | OP\_Code | Symbol |
| Add | 000 | ADD |
| Subtract | 001 | SUB |
| NAND | 010 | NND |
| NOR | 011 | NOR |
| LOAD | 100 | LOD |
| STORE | 101 | STR |
| JUMP | 110 | JMP |
| HALT | 111 | HLT |

Table 1 Operations, Op\_Code, and Symbols for the Simple CPU

In the above table, a three-letter symbol is also defined for each operation. Writing a program using these symbols would be easier since remembering the symbols is easier than remembering the codes.

**Assembly language**: A language in which each instruction of the program uses symbols rather than binary code.

Also, the instruction format for the simple CPU was designed as follows.

     
The format shows that each instruction consists of three bits of Op\_Code and five bits of address. For example, if the first three bits of an instruction is 000, the simple CPU hardware is designed to recognize it as the add operation (refer to the table above). Since only one format of an instruction is defined, there will be eight instructions for this CPU. One instruction would be associated with one operation.

**Instruction Set**: The set of all instructions that a CPU realizes is called the instruction set of that CPU.

## PROCEDURE

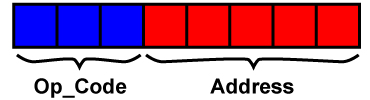
1. Watch the animation at the end of Lecture 1. The machine language of a program that adds two numbers is given in that animation. Explain the cycles that the CPU goes through in that animation.
2. Given the diagram shown in the animation, assume that there are five numbers stored in Memory Locations 20-24. Write the code to add these numbers and store the result in Location 25.
3. Write the machine language program (binary code).
4. Write the assembly language program.
5. The following problems use a different instruction format than the one used in the animation of Lecture 1. Use the following instruction format for this problem.

There are three numbers stored in Memory Locations 10-12. Write a machine language program to load each number in the accumulator one at a time. Use the immediate addressing mode to add four to each value that is loaded to accumulator. Store the results in Locations 13-15. Let’s assume that in Locations 10-12 we have the values 5, 6, and 7. After executing your code, the values in Locations 13-15 should be 9, 10, and 11.

1. Write the assembly language program.
2. Write the machine language.

Use the following instruction format only for Problem 3.

When the value of the green bit is zero, the four bit to the right (red bits) is the address of the operand.



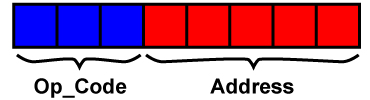
**0**

**Address**

**Op\_Code**

**When this bit = 0, the red bits are the address. This is called direct addressing mode.**

When the green bit is one, the four bits to the right (the red bits) are the actual data.



**1**

**Data**

**Op\_Code**

**When this bit = 1, the red bits are the actual data.**

**This is called immediate addressing mode.**

Example of this instruction format:

The binary code 00001100 means add Accumulator A with the value in address 1100. Let’s assume that we have the value 5 in Location 1100. The CPU will get the content in address 1100 (which is value 5) and add 5 to Accumulator A. Let’s say that the original value of A is 8. After the execution of this operation, it becomes 13.

What does the binary code 00011100 mean? Since the green bit is 1, it means that the last four bits is not the address, but the actual data. Therefore, the CPU will add Accumulator A with the value 1100. Now, if the original value of A is 8, after this operation, it becomes 20.

Now that we have two ways of specifying where the operand resides, we have two different ways to address where the data is located. One way is to provide the data immediately after the operation. The other is to provide the address of the data. Therefore, we have two different ways, or two different modes, of specifying where the operand resides.

In our assembly language of simple CPU, let’s use the symbol # to distinguish between the two types of addressing modes.

Now we have two types of instructions for the ADD operation. These instructions are:

1. ADD 12 means the accumulator will be added to the content of location 12. Its machine code is 00001100; and
2. ADD #12 means the accumulator will be added to 12. Its machine code is 00011100.

Let’s call the first instruction direct addressing mode. Let’s call the second addressing mode immediate addressing mode.