Microprocessors (0630371)
Fall 2010/2011 – Lecture Notes # 4

Outline of the Lecture

- **Arithmetic and logic Unit (ALU).**
- **Assembly Language programming-Introduction.**
- **Description of Instruction Set**

Arithmetic and logic Unit (ALU)

The basic operations are implemented in hardware level. ALU is having collection of two types of operations:

- **Arithmetic** operations
- **Logical** operations

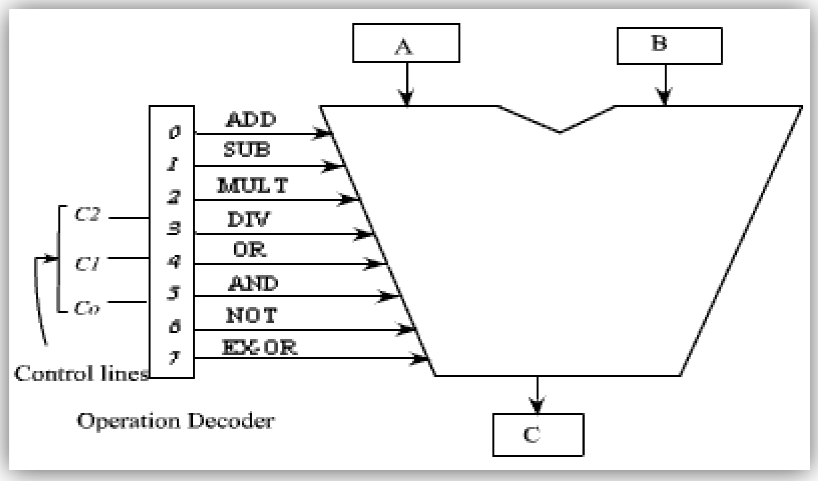
Consider **hypothetical** ALU having 4 arithmetic operations and 4 logical operations.

- Four arithmetic operations. Addition, subtraction, multiplication and division.
- Four logical operations: OR, AND, NOT & XOR.

To identify any one of these four logical operations or four arithmetic operations, two control lines are needed. Also to identify the any one of these two groups- arithmetic or logical, another control line is needed. So, we need three control lines.
We need three control lines to identify any one of these operations. The input combination of these control lines are shown below:
Control line C_2 is used to identify the group: logical or arithmetic, i.e.
$C_2 = 0$: arithmetic operation $C_2 = 1$: logical operation.
Control lines C_0 and C_1 are used to identify any one of the four operations in a group. One possible combination is given here.

C_1	C_0	Arithmetic $C_2 = 0$	Logical $C_2 = 1$
0	0	Addition	OR
0	1	Subtraction	AND
1	0	Multiplication	NOT
1	1	Division	**EX-OR**

A 3X8 decode is used to decode the instruction. The block diagram of the ALU is shown in figure



Block Diagram of the ALU

The ALU has got two input registers named as A and B and one output storage register, named as C. It performs the operation as:

C=A op B

The input data are stored in A and B, and according to the operation specified in the control lines, the ALU perform the operation and put the result in register C.

As for example, if the contents of controls lines are, 000, then the decoder enables the addition operation and it activates the adder circuit and the addition operation is performed on the data that are available in storage register A and B . After the completion of the operation, the result is stored in register C.

We should have some hardware implementations for basic operations. These basic operations can be used to implement some complicated operations which are not feasible to implement directly in hardware.

Assembly Language programming-Introduction

Assemble, Link and Run a Program

The following summarizes the process required to assemble, link and run an assembly language program.

Step 1: Edit the program using **Editor** (save file as **`*.asm`**)

Step 2: Assemble the program (**`*.asm`**) using **MASM or TASM**, you will get (**`*.obj`**).

Step 3: Link the program (**`*.obj`**) using **LINK or TLINK** you will get (**`*.exe`**)

- **`".asm"`** file is the source file created with an editor or a word processor.
- **`".obj"`** assembler (e.g.TASM) converts .asm file's Assembly language instructions into machine language.
- **`".exe"`** TLINK is the program to produce the executable file.

Assembly Language program consists of series of lines of Assembly language **instructions**.

Instruction format

- An assembly language instruction consists of four fields,

`[label:]  mnemonic  [operands]  [;comments]`

Label (optional)
Instruction mnemonic (required)
Operand(s) (usually required)
Comment (optional)

Brackets indicate that the field is optional. Brackets are not typed.

- The **label** field allows the program to refer to a line of code by name.
- In a line of assembly language program there can be **mnemonic** (instruction) and **operand(s).**

```
Ex:     ADD  AL, BL
        MOV  AX, 6764H
```

- The comment field begins with a **";".**
- Alternatively, instead of these two fields there can be directives. Directives are used by the assembler to organize the program as well as other output files. **SEGMENT, DB, ENDS, ASSUME, END,** and **ENDP** are examples of directives. **Directives** are statements that give directions to the assembler about how it should translate the assembly language instructions into machine code.

Description of Instruction Set

8086 has 117 instructions, 8086 instruction set consists of the following instructions:

- **Data transfer** instructions---- **MOV**, **PUSH**
- **Arithmetic** - add, subtract, increment and decrement -----**ADD, SUB, MUL, DIV, INC**.
- **Logic** instructions – **NOT, AND, OR, XOR**.
- **Control transfer** –
  - Conditional, unconditional program branch---- **LOOP, LOOPE, LOOPZ.**
  - Conditional, unconditional call subroutines--- **CALL, JMP, JA, JNBE.**
- **Enabling/disabling interrupts**---- **INT, INTO, IRET.**
- Stack operations----**PUSH, POP.**
- Simple **Input and Output port transfer** instructions----**IN, OUT**
- Special **Address transfer** instructions ------**LEA, LDS**
- **Setting/Clearing flag** bits--- **STC, CLC, STD, CLD**
- **String** Instructions ------**MOVS, MOVSB, MOVSW**.
- External **Hardware Synchronization** instructions---- **HLT, WAIT, NOP**