CSE 331/EEE 332 (Microprocessor Interfacing & Embedded System Lab)

Lab 03 : Conditional jumps/Unconditional jumps;
Memory Management: PUSH, POP, PTR (POINTER);
Instructions: MUL, DIV, CMP, SUB, AND, JZ, JMP

Lab Instructor : Rokeya Siddiqua

Topics to be covered in class today:

- Conditional Jumps/Unconditional Jumps
- Instructions: CMP, AND, SUB, MUL, DIV, JZ, JMP
- Memory Management: PUSH, POP, PTR (POINTER)

Instruction	Operands	Description

MUL	REG, REG
	REG, memory
		Multiplication.
		
		8 bit multiplication
		If we multiply two 8 bit unsigned positive numbers, we will get an unsigned 16 bit result. For this operation, we have to put one operand in accumulator register. The output of the multiplication will be stored in ax.
		
		16 bit multiplication
		If we multiply two 16 bit unsigned positive numbers, we will get an unsigned 32 bit result. For this operation, we have to put one operand in accumulator register. The 32 bit result becomes available in the dx register and ax register. The lower 16 bit will be stored in ax register and the higher 16 bit will be stored in dx register.
		
		Algorithm:
		
		operand1 = operand1 * operand2
		
		Example:
		
		MOV AL, 5
		MOV DL, 6
		MUL DL

DIV	REG, REG
	REG, memory
		Division.
		
		8 bit division
		If we divide a 16 bit unsigned positive number by an 8 bit unsigned positive number, the quotient of the division will be stored in al register and the remainder will be stored in ah register.
		AL = Quotient
		AH = Remainder
		
		16 bit division
		If we divide a 32 bit unsigned positive number by another 16 bit unsigned positive number, the quotient of the division will be stored in ax register and the remainder will be stored in dx register.
		AX = Quotient
		DX = Remainder
		
		Algorithm:
		
		operand1 = operand1 / operand2
		Example:
		MOV AX, 1234H
		MOV BL, 23H
		DIV BL

CMP	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Compare.
		
		Algorithm:
		
		operand1 - operand2
		
		Result is not stored anywhere, flags are set (OF, SF, ZF, AF, PF, CF) according to result.
		
		Example:
		
		MOV AL, 5
		MOV BL, 5
		CMP AL, BL ; AL = 5, ZF = 1 (so equal!)

SUB	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Subtract.
		
		Algorithm:
		
		operand1 = operand1 - operand2
		
		Example:
		
		MOV AL, 5
		SUB AL, 1 ; AL = 4

AND	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Logical AND between all bits of two operands. Result is stored in operand1. These rules apply: 1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0 Example: MOV AL, 'a' ; AL = 01100001b AND AL, 11011111b ; AL = 01000001b ('A')

Jump Instruction

Jump Instructions are used for changing the flow of execution of instructions in the processor. If we want jump to any instruction in between the code, then this can be achieved by these instructions. There are two types of Jump instructions:

- Unconditional Jump Instructions
- Conditional Jump Instructions

Instruction	Operands	Description

JZ	Label	Short Jump if Zero (equal). Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions.
		
		Algorithm:
		
		if ZF = 1 then jump
		
		Example:
		
		.MODEL SMALL
		.STACK 100H
		.DATA
		.CODE
		MAIN PROC
		
		MOV AL, 5
		CMP AL, 5
		JZ label1
		MOV DL, 1
		JMP exit
		
		label1:
		MOV DL, 0
		
		exit:
		ENDP MAIN
		END MAIN

JMP	Label	Unconditional Jump. Transfers control to another part of the program. 4-byte address may be entered in this form: 1234h:5678h, first value is a segment second value is an offset.
		
		Algorithm:
		
		always jump
		
		Example:
		
		.MODEL SMALL
		.STACK 100H
		
		.DATA
		
		.CODE
		
		MAIN PROC
		
		MOV AL, 5
		JMP exit ; jump over 2 lines!
		MOV AL, 0
		
		exit:
		ENDP MAIN
		
		END MAIN

PUSH		Store 16 bit data into two locations of SSM (stack) pointed by SS:SP The data source may be: • 16 bit register (except IP, CS) • Two consecutive memory locations ; assume ax = 4567H PUSH AX PUSH DS PUSH WORD PTR DS:[BX]
POP		Retrieve 16 bit from two locations of stack pointed by SS:SP The data destination may be: • 16 bit register • Two consecutive memory locations POP AX POP DS POP WORD PTR DS:[BX]

Difference between CMP and SUB

CMP: Comparison of two numbers, is carried out in the form of a subtraction to determine which of the operands has a greater value. After a CMP instruction, PSW or flag resister get updated. For example, if the operands have equal values, then ZF will be set to 1.

The CMP instruction **does not modify** the **destination field**

SUB: SUB instruction subtracts the source value from the destination. The logic of the SUB instruction is:

destination = destination - source

The SUB instruction **modifies** the **destination field**

Labels

- Labels mark places in a program which other instructions and directives reference
- Labels in the code segment always end with a colon
- Labels in the data segment never end with a colon
- Labels can be from 1 to 31 characters long and may consist of letters, digits, and the special characters ? . @ _ $ %
- If a period is used, it must be the first character
- Labels must not begin with a digit
- The assembler is case insensitive

Legal and Illegal Labels

Examples of legal names

- COUNTER1
- @character • SUM_OF_DIGITS
- $1000 o DONE?
- .TEST

Examples of illegal names

- TWO WORDS contains a blank
- 2abc begins with a digit
- A45.28 . not first character
- YOU&ME contains an illegal character

Example :

```
Start:
mov    ax,@data
mov    ds, ax
jmp    Exit
mov    cx, 10
```

Memory Management: PTR (POINTER)

Example: Read two data (1 byte) from the last and first memory location of MPU 8086 and store their sum in the last even memory location of MPU 8086.

Solution:

```
MOV AX, 0F000 H;      % Loading the last address
MOV DS, AX
MOV BX, 0FFFF H
MOV AL, BYTE PTR DS:[BX];      % Loading data in AL
MOV CX, 0000 H;     %Loading the  first address
MOV DS, CX
MOV BX, 0000 H
MOV AH, BYTE PTR DS:[BX];     % Loading data in AH

ADD AL, AH;     % Adding
MOV CX, 0F000 H;      % Loading the last even address
MOV DS, AX
MOV BX, 0FFFE H
MOV BYTE PTR DS:[BX], AL ;      % Storing in the last address
HLT  ;      % End the program
```

Task 1

Write a program that will count the number of characters in a string.

Task 2

Write a program that will concatenate (join) two strings. Make sure the input strings are not destroyed and the final answer must be inside a third array. Input from user not required. Create two strings in your program.

Example:
String 1: “Hello World, “
String 2: “this is Assembly Language Programming”

Task 3

Read two data (1 byte) from the 3rd and 2nd last memory location of MPU 8086 and store their multiplication and division in the last 2 memory locations of MPU 8086.