8086 Microprocessor Laboratory Experiments

Experiment 5: Arithmetic Instructions

Murtadha Hssayeni, Ph.D.

m.hssayeni@uobabylon.edu.iq

Outlines

- Flag register in 8086
- Arithmetic Instructions
- ADD, SUB
- MUL, IMUL, DIV, and IDIV
- INC, DEC
- Procedure and Discussion

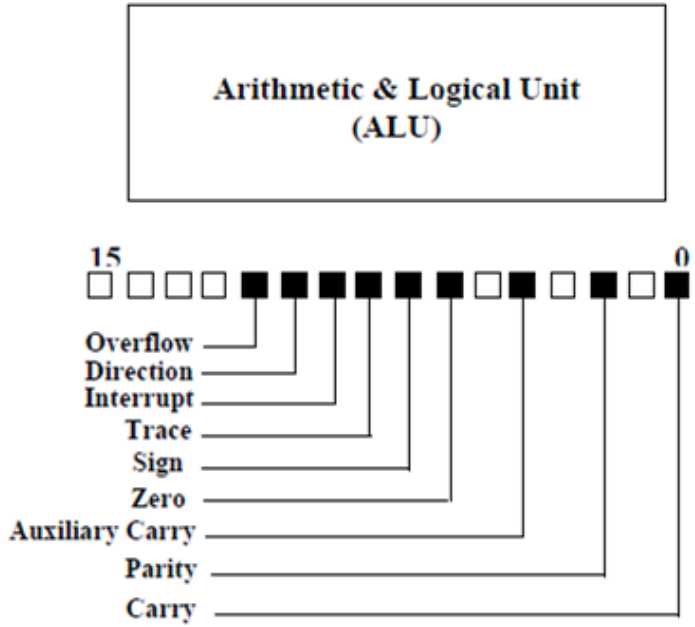
Arithmetic Instructions

- The purpose of this lab is to become familiar with the use of arithmetic instructions.
- Most Arithmetic and Logic Instructions affect the processor status register (or Flags)

- **There are 3 groups of instructions:**
  - ADD, SUB
  - MUL, IMUL, DIV, and IDIV
  - INC, DEC

Flag register in 8086

- The flag register is a 16 bit register that holds information of microprocessor status after executing an operation.
- Most Arithmetic and Logic Instructions affect the processor status register (or Flags).
  - Sign: Contains the sign bit of the result of the last arithmetic operation.
  - Zero: Set when the result is 0.
  - Carry: Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit.
  - Overflow: Used to indicate arithmetic overflow.
  - Interrupt Enable/Disable: Used to enable or disable interrupts.
  - The Parity flag (PF): set if the least-significant byte in
  - the result contains an even number of 1 bits. Otherwise, PF is clear.



Arithmetic Instructions: ADD

Two-address Instruction

- These types of operands are supported:
  - *ADD REG, memory*
  - *ADD memory, REG*
  - *ADD REG, REG*
  - *ADD memory, immediate*
  - *ADD REG, immediate*
  - REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.
  - memory: [BX], [BX+SI+7], variable, etc.
  - Immediate: 5, -24, 3Fh, 10001101b, etc.
- ADD: add second operand to first without adding carry flag
  - ADD operand1, operand2
  - Algorithm: operand1 = operand1 + operand2
- **ADD result is always stored in first operand**.
- This instruction affect these flags only:
  - CF, ZF, SF, OF, PF, AF.

Arithmetic Instructions: ADD

- **Example**:

```
MOV AL, 5H       ; AL = 5
ADD AL, -3H      ; AL = 2
MOV CL, AL
MOV AX, 00AAH
MOV BX, 0065H
ADD AX, BX       ; AX=010FH
HLT
```

- **ADC** - add second operand to first with carry.
  - ADC operand1, operand2
  - Algorithm: operand1 = operand1 + operand2 + CF

- **Example**:

```
STC ; set CF = 1
MOV AL, 5        ; AL = 5
ADC AL, 1        ; AL = 7
HLT
```

Arithmetic Instructions: SUB

- **SUB -** Subtract second operand to first without borrow.
- Algorithm: operand1 = operand1 - operand2
- **Example**:

```
MOV AL, 5
SUB AL, 1        ; AL = 4
MOV BL, 66H
SUB BL, AL       ; BL=62
```

- This instruction affect these flags only:
  - CF, ZF, SF, OF, PF, AF.

- **SBB -** Subtract second operand to first with Borrow.
  - Algorithm: operand1 = operand1 - operand2 – CF
- **Example**:

```
STC
MOV AL, 5
SBB AL, 3        ; AL = 5 - 3 - 1 = 1
```

Arithmetic Instructions: MUL

One-address Instruction

- These types of operands are supported:
  - *MUL REG*
  - *MUL memory*
  - REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.
  - memory: [BX], [BX+SI+7], variable, etc.
- MUL and IMUL instructions affect these flags only:
  - CF, OF
  - when result exceeds the operand size these flags are set to 1, when result fits in operand size these flags are set to 0.

Arithmetic Instructions: MUL

- **MUL** - Unsigned multiply:
  - When operand is a byte:
    - AX = AL * operand.
  - When operand is a word:
    - (DX AX) = AX * operand.

- **Example**:

```
MOV AL, C8H     ; AL = (200)
MOV BL, 4H
MUL BL          ; AX = 0320H (800)
MOV SI, AX
MOV CX, 190H    ; CX= (400)
MUL CX          ; DX AX= 0004 E200
HLT
```

- **IMUL** - signed multiply:
  - When operand is a byte:
    - AX = AL * operand.
  - When operand is a word:
    - (DX AX) = AX * operand.

- **Example**:

```
MOV AL, -2
MOV BL, -4
IMUL BL         ; AX = 8
```

Arithmetic Instructions: DIV

One-address Instruction

- **DIV** - Unsigned divide:
  - When operand is a byte:
    - AL = AX / operand
    - AH = remainder (modulus).
  - When operand is a word:
    - AX = (DX AX) / operand
    - DX = remainder (modulus).
- **Example**:

```
MOV AX, 00CBH ; AX = 203
MOV BL, 04
DIV BL        ; AL = 50 (32h), AH = 3
```

- **IDIV** - signed divide:
  - When operand is a byte:
    - AL = AX / operand
    - AH = remainder (modulus).
  - When operand is a word:
    - AX = (DX AX) / operand
    - DX = remainder (modulus).
- **Example**:

```
MOV AX, -203    ; AX = 0FF35h
MOV BL, 4
IDIV BL         ; AL = -50 (0CEh), AH = -3 (0FDh)
```

Arithmetic Instructions: INC, DEC

- **These types of operands are supported:**
  - DEC or INC REG
  - DEC or INC memory
- Decrement Algorithm: operand = operand - 1
- Increment Algorithm: operand = operand + 1
- INC, DEC instructions affect these flags only:
  - ZF, SF, OF, PF, AF.

- **Example**:

```
MOV AL, 255     ; AL = 0FFh (255 or -1)
DEC AL          ; AL = 0FEh (254 or -2)
```

Procedure

1. Write a program in 8086 emulator to perform the following tasks:
   - A = X - Y + 20H where X= 60H, Y = 19H and store the results in to register DX.
   - Run the program and write the results with flags.
2. Write the program again and store the results into memory location 3DD00H with segment register DS= 3000H using an addressing mode of your choice. Run the program and write the result of each register and memory being used.
3. Write a program to solve the following expression Y = B/A where A= 0FH and B = E5H and store the result in the logical address 3000H:DD02H..

Discussion

1. How can you modify the program in the procedure to calculate the following expression:
   - C = A+A-B.
   - C = A*2 +B.
   - C = B/A.
2. Write a program to store the reminder of the division (C = B/A) to the memory location 14141H.
3. Write a program to calculate the result of (FA89H+03CDH +79CEH), illustrating the effect of this operation on flag registers?