**-Assembly Language – Data Transfers, Addressing, and Arithmetic (1)**

4-1 Integer Arithmetic, Data Transfer Instructions

Objective： Familiar with the instructions MOV and SHIFT.

a. The following codes intend to set Rval = 19\*Val1. Please complete the codes.

| .data  Val1 SBYTE -1 ; Val1 = 0ffh  Rval SWORD ?  .code  main PROC  L1:  movsx ax, Val1 ;BaseValue  movzx bx, Val1  mov cl, Val1  mov ch, 10h     | L2: |  |  |  | AX | DX | | --- | --- | --- | --- | --- | --- | | mov dx, | 0 | ;1. dx = 0 | 1 | **0ffffh** | **0000h** | | add dx, | ax | ;2.*dx*=*dx*+*ax* | 2 | **0ffffh** | 0ffffh | | shl ax, | 1 | ;3.*ax*\*2 | 3 | 0fffeh | 0ffffh | | add dx, | ax | ;4.dx=2Val1+Val1=3Val1 | 4 | 0fffeh | 0fffdh | | shl ax, | 3 | ;5.*ax*\*8, ax = 16Val1 | 5 | 0fff0h | 0fffdh | | add dx, | ax | ;6.*dx*=19\**Val1* | 6 | 0fff0h | 0ffedh |     L3: exit  main ENDP  END main |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

1. Based the code above, when is not any command executed at L1 position, registers values will be as follows:

| EAX | 00000000h | EBP | 0012fb50h |
| --- | --- | --- | --- |
| EBX | 00000000h | ESP | 0012fb24h |
| ECX | 0012fb08h | ESI | fffffffeh |
| EDX | 772070b4h | EDI | 00000000h |

When all the commands have been executed before L3 position, what is registers values?

| EAX | 0000fff0h | EBP | 0012fb50h |
| --- | --- | --- | --- |
| EBX | 000000ffh | ESP | 0012fb24h |
| ECX | 001210ffh | ESI | fffffffeh |
| EDX | 7720ffedh | EDI | 00000000h |

**Assembly Language – Data Transfers, Addressing, and Arithmetic (2)**

| Group : | Student ID : | Name : |
| --- | --- | --- |
| Student ID : | Name : |

4-2 Data Transfer Instructions, Addition and Subtraction

Objective: Understanding the instructions (ADD, SUB, and NEG)

1. Complete the following code to implement this equation:

Rval = -(Val3 - (Val1 - Val2))

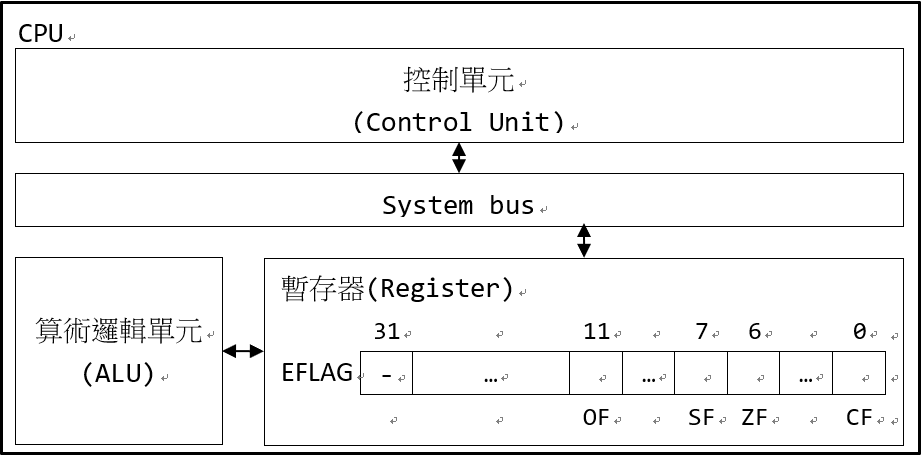
| .data  Val1 SBYTE 03h  Val2 SBYTE 02h  Val3 SBYTE 8fh  Rval SWORD ?  .code  movsx ax, Val1  movsx bx, Val2  movsx cx, Val3  ; Rval = -(Val3 - (Val1 - Val2))   | sub | ax, | bx | ; ax= (Val1 – Val2) | | --- | --- | --- | --- | | sub | cx,ax | | ; cx = Val3 – (Val1 – Val2) | | neg cx | | | ; cx = -cx | |  | | |  | | mov | Rval, cx | | ; Rval = cx | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Assembly Language – Data Transfers, Addressing, and Arithmetic (3)**

| Group : | Student ID : | Name : |
| --- | --- | --- |
| Student ID : | Name : |

4-3 Flags

Objective: Understanding changes of Flags and its function related to data operator.



1. The picture shows a simple diagram of the CPU architecture, FLAG register will record every time based on the result from ALU. Each operation makes a corresponding change. Fill the form below to show changes Carry Flag, Zero Flag, Sign Flag and Overflow Flag respectively in every following sequence.

| .data  mmyArray WORD 00ffh | OF | SF | ZF | CF | AL |
| --- | --- | --- | --- | --- | --- |
| .code | 0 | 0 | 1 | 0 | 00h |
| mov ax,[myArray] | 0 | 0 | 1,#0 | 0 | 0ffh |
| add al, 1 | 0,#1 | 0 | 1 | 1 | 00h |
| mov al, 7fh | 0 | 0 | 1 | 1 | 7fh |
| add al, 2 | 1 | 1 | 0 | 0,#1 | 81h |

**Assembly Language – Data Transfers, Addressing, and Arithmetic (4)**

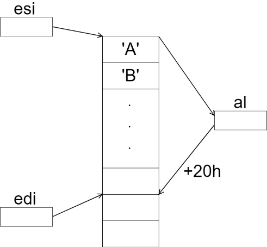
* 1. Indirect Addressing、JMP and LOOP Instruction

Objective: Understanding indirect addressing for LOOP application.

1. Using indirect addressing and loop, store five letters in ChStr1 to ChStr2 to change capital letter to its lowercase by adding 20h!

| .data  ChStr1 BYTE 'A', 'B', 'C', 'D', 'E'  ChStr2 BYTE LENGTHOF ChStr1 DUP(?)  .code   | mov ecx, | LENGTHOF ChStr1 | ; set number of LOOP execution | | --- | --- | --- | |  | | ; for ChStr1 using LENGHTOF |   mov esi, OFFSET ChStr1 ; esi is pointing to ChStr1[0]  mov edi, OFFSET ChStr2 ; edi is pointing to ChStr2[0]  L: ; LOOP starting point   | mov | al, [esi] | ; set value of al with address | | --- | --- | --- |   ; value in esi    add al,20h ; 'A' -> 'a'   | mov | [edi],al | ; set address value in edi | | --- | --- | --- | |  | | ; with value of al |   L2:  inc esi ; esi is pointing to the next  ; point of ChStr1  inc edi ; edi is pointing to the next  ; point of ChStr2 Element   | LOOP | L | | --- | --- | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

| item | [esi] | [edi] | ecx |
| --- | --- | --- | --- |
| 0 | 41h | 00h | 5 |
| 1 | **41h** | **61h** | **5** |
| 2 | **42h** | **62h** | **4** |
| 3 | **43h** | **63h** | **3** |
| 4 | **44h** | **64h** | **2** |
| 5 | **45h** | **65h** | **1** |



| Symbol | hex | Symbol | Hex |
| --- | --- | --- | --- |
| A | 41h | a | 61h |
| B | 42h | b | 62h |
| C | 43h | c | 63h |
| D | 44h | d | 64h |
| E | 45h | e | 65h |