ECE3210 Microprocessor Engineering

Homework 6

1. Chapter 6. 27

```
        MOV  SI,OFFSET BLOCK
        MOV  UP,0
        MOV  DOWN,0
        MOV  CX,100H
        MOV  AL,42H
        CLD
L1:     SCASB
        JE   L3
        JA   L2
        INC  DOWN
        JMP  L3
L2:     INC  UP
L3:     LOOP L1
```

2. Chapter 6.47

IRET:

1. pop stack data back into IP
2. pop stack data back into CS
3. pop stack data back into the flag register

So IRET accomplishes the same task as a FAR RET + POPF

3. Conditional jump

(1) Indicate whether or not the jump happens (show work)

a.
```
MOV CL,5
SUB AL,AL
SHL AL,CL
```

b.
```
MOV BH,65H
MOV AL,48H
OR  AL,BH
```

```
JNC LABEL1                    SHL AL,1
                              JC  LABEL1
```

(Yes jump) (No jump)

(2) Under what conditions (CX, ZF) does the LOOPE instruction jump to its label?

CX!=0 and ZF =1

The LOOPE instruction jumps when an equal condition exists and CX is not a zero and it also decrements CX on each iteration of the loop.

(3) Implement the following pseudo code into assembly language using conditional JMP. Assume signed number comparison.

```
a.   if (DX <= CX)
          X =1
     else
          X =2
```

```
     CMP DX, CX
     JG L1      ; else case
     MOV X, 1
     JMP ENDC
L1:  MOV X, 2
ENDC:
```

```
 b. if (BX > CX OR BX > VAL1)
          X =1
     else
          X=2
```

```
     CMP BX, CX
     JG L1
     CMP BX, VAL1
     JLE L2
L1:   MOV X, 1
      JMP ENDC
L2:   MOV X, 2
ENDC:
```

4. Assume that array x has been loaded with 100 signed numbers. Write a program which copies the absolute value of each number in x into the corresponding location of y.

```
.data
x   db     100 dup (?)
y   db     100 dup (?)

.code
; TO DO - place your code here
```

Solution:

```
.MODEL MEDIUM
.STACK 100H

.DATA
x   db     100 dup (?)
y   db     100 dup (?)


.CODE
.STARTUP
MAIN PROC FAR
     mov cx, 100
     lea si, x
     lea di, y

begin:
     mov al, [si]
     cmp al, 0
     jg l1
     ; or
     ; test al, 80h
     ; jz l1
     neg al
l1:  mov [di], al
     inc si
     inc di
     loop begin

.EXIT
MAIN ENDP

END
```

5. Write an assembly language program that would implement the following C program, as might be produced by C compiler. All variables must be kept up to date in memory.

```
Main()
{
      int k;
      int p[500];
      for (k=0; k<500; k++){
            if (p[k]<300){
                  p[k]= p[k] + k;
            }
            else{
                  p[k] = p[k]-k;
            }
      }
}
```

Assembly:

```
.data
k   dw     ?
p   dw     500 dup (?)

.code
; TO DO - place your code here
```

Solution:

```
.MODEL MEDIUM
.STACK 100H

.DATA
k   dw     ?
p dw 500 dup (0)


.CODE
.STARTUP
MAIN PROC FAR
      mov k, 0
      mov cx, 500
      mov si, 0

start:
      mov ax, k
      cmp k, 300
      jg minus
plus:
```

```
        add p[si], ax
        jmp next
minus:
        sub p[si], ax
next:
        inc k  ; k++
        inc si
        inc si
        loop start

.EXIT
MAIN ENDP

END
```

6. Macros

(a) Fill in the blanks for the following macro to add any array of bytes. Some blanks might not need to be filled.

```
SUMMING MACRO COUNT, VALUE
LOCAL ___AGAIN_______
;; this macro adds an array of byte size elements,
;; ax holds the total sum
MOV CX, ____COUNT ______                ; size of array
MOV SI, OFFSET ____VALUES _______       ; load offset address of
array
SUB AX, AX                          ; clear ax
AGAIN: ADD AL, [SI]
ADC AH, 0                       ; take care of carries
INC  SI                         ; point to the next byte
LOOP AGAIN                          ; continue until finish
ENDM _____________
```

(b) In the data segment

```
DATA1 DB   89, 75, 98, 91, 65
SUM1 DW ?
```

Write instructions below to invoke the macro in (a) for the above data, and save the results in SUM1

_____ SUMMING 5, DATA1_____

_____ MOV SUM1, AX _____

7. Write a procedure using instruction ***loop*** that returns the minimum 16-bit **signed** integer from an array of integers. On procedure entry, register SI points to the start of the array (each element is 16 bits), and register CX has the number of integers in the array. The minimum value should be returned in the AX register. An example call to this procedure is shown below. Test your procedure with the assembler. Include a screenshot that reflects variable RESULT content after your code execution.

```
.DATA
MYARRAY DW -45, 1000, -34, 1500, 20, 60
COUNT EQU 6 ; SIX ELEMENTS IN THE ARRAY
RESULT   DW 0

.CODE
MOV AX,@DATA
MOV DS,AX
MOV CX, COUNT
LEA SI, MYARRAY
CALL FINDMIN
; TO DO – place your code here
```

Solution:

```
.MODEL MEDIUM
.STACK 100H

.DATA
MYARRAY DW -45, 1000, -34, 1500, 20, 60
COUNT EQU 6 ; SIX ELEMENTS IN THE ARRAY
RESULT DW 0

.CODE
.STARTUP
MAIN PROC FAR

 MOV AX,@DATA
 MOV DS,AX
 MOV CX, COUNT
 LEA SI, MYARRAY
 CALL FINDMIN
 MOV RESULT, ax


.EXIT
MAIN ENDP
```

```
FINDMIN PROC NEAR

        MOV AX, [SI]
        DEC CX
BEGIN:
        INC SI
        INC SI
        CMP AX, [SI]
        JL L1
        MOV AX, [SI]
L1:     LOOP BEGIN

          RET
FINDMIN ENDP


END
```

----------End----------