ECE3210 Microprocessor Engineering

Homework 5

1. Chapter 5.5

```
ADD  AH,AL
ADD  AH,BL
ADD  AH,CL
ADD  AH,DL
MOV  DH,AH
```

2. Chapter 5.11
The instruction does not specify the size of the data addressed by BX and can be corrected with a BYTE PTR, WORD PTR, DWORD PTR, or QWORD PTR.

3. Chapter 5. 13. Show flag register bits Z, S, C, A, P, O

DL = 0F3H, BH = 72H

SUB DL, BH

DL = 81H, S = 1, Z = 0, C = 0, A = 0, P = 1, O = 0

4. Chapter 5. 39 (c) (e) (f)
(c) AND DI,BP
(e) AND [BP],CX
(f) AND DX,[SI–8]

5. Chapter 5.55

```
MOV  DI,OFFSET LIST
MOV  CX,300H
CLD
MOV  AL,66H
REPNE SCASB
```

6. Assume the following registers contain these hex contents:
AX = F000, BX = 3456, DX = E390. Perform the following operations. Indicate the result and the register where it is stored. Give also ZF and CF in each case.
(a) AND DX, AX

DX = E000, CF =0, ZF = 0

(b) OR DH, BL

DH = F7, CF = 0 , ZF = 0

(c) XOR AX, AX

AX = 0, CF= 0, ZF=1

(d) MOV CL, 3
SHR DL, CL

DL = 12, CF = 0, ZF = 0

(e) SUB CX, CX
DEC CX

CX = FFFF, CF= 0, ZF = 0

7. Indicate status of ZF and CF after CMP is executed in each of the following cases.

(a) MOV BX, 2500
CMP BX, 1400

CF = 0 and ZF = 0, because 2500 > 1400

(b) MOV al, 0AAH
AND AL, 55H
CMP AL, 00

CF = 0 and ZF = 1,

8. Write, run and analyze a program that calculate the total sum paid to a salesperson for eight month. The following are the monthly paychecks for those months: $2300, $4300, $1200, $3700, $1298, $4323, $5673, $986. Part of the program is provided below. Fill in the TO DO part. The results of the addition is 5CE4H, and should be placed in variable SUM as HEX number. You don't need to convert SUM to ASCII. Use assembler to test your program.

```
STSEG SEGMENT
        DB 64 DUP (?)
STSEG ENDS
;---------------------------
DTSEG SEGMENT
        COUNT DB 08
        DATA DW 2300,4300,1200,3700,1298,4323,5673,986
```

```
        ORG 0010H
        SUM DW 2 DUP(?)
DTSEG ENDS
;-----------------------------
CDSEG SEGMENT
MAIN PROC FAR
ASSUME CS:CDSEG,DS:DTSEG,SS:STSEG
        MOV AX,DTSEG
        MOV DS,AX

        ; TO DO – place your code here

        MOV AH,4CH
        INT 21H ;go back to DOS
MAIN ENDP
CDSEG ENDS
END MAIN
```

Solution:

```
STSEG SEGMENT
        DB 64 DUP (?)
STSEG ENDS
;-----------------------------
DTSEG SEGMENT
        COUNT DB 08
        DATA DW 2300,4300,1200,3700,1298,4323,5673,986
        ORG 0010H
        SUM DW 2 DUP(?)
DTSEG ENDS
;-----------------------------
CDSEG SEGMENT
MAIN PROC FAR
ASSUME CS:CDSEG,DS:DTSEG,SS:STSEG

        MOV AX,DTSEG
        MOV DS,AX

        MOV CX, 0
        MOV CL,COUNT ;CX is the loop counter
        MOV SI,OFFSET DATA    ;SI is the data pointer
        MOV AX,00             ;AX will hold the sum
        MOV BX,AX             ;BX will hold the carries
BACK:   ADD AX,[SI]           ;add the next word to AX
        ADC BX,0              ;add carry to BX
        INC SI                ;increment data pointer twice
        INC SI                ; to point to next word
        DEC CX                ;decrement loop counter
        JNZ BACK              ;if not finished, continue adding
        MOV SUM,AX            ;store the sum
        MOV SUM+2,BX          ;store the carries

        MOV AH,4CH
        INT 21H               ;go back to DOS
MAIN ENDP
CDSEG ENDS
END MAIN
```

9. Using code provided below, write a procedure FCTN1 that evaluate (3*X + 7* Y). Use stack to pass parameters, and DX-AX to return evaluation result from the procedure. Test your procedure with the assembler. Include a screenshot that reflects variable RESULT content after your code execution.

```
; procedure: int fctn1(int x, int y)
; variable x and y, call procedure to evaluate 3*x+7*y

.MODEL MEDIUM

.STACK 100H

.DATA
X DW    5
Y DW    10
RESULT  DW      2 DUP (?)

.CODE
        .STARTUP
MAIN      PROC FAR

        MOV     AX,@DATA                ;initialize DS to address
        MOV     DS,AX
        MOV     ES,AX

        PUSH X
        PUSH Y
        CALL FCTN1
        MOV RESULT, AX
        MOV RESULT[2], DX


        .EXIT

  MAIN ENDP


; procedure: int fctn1(int x, int y)
; returns 3*x+7*y in DX-AX
FCTN1 PROC NEAR

; TO DO – place your code here

FCTN1 ENDP


END
```

Solution:

```
; procedure: int fctn1(int x, int y)
```

```
; returns 3*x+7*y in DX-AX
FCTN1 PROC NEAR
        PUSH BP
        MOV BP, SP

        MOV AX, [BP+6] ;
        MOV BX, 3
        IMUL BX

        PUSH DX
        PUSH AX

        MOV AX, [BP+4]
        MOV BX, 7
        IMUL BX

        POP BX  ; PREVIOUS MULT RESULT - AX PART
        ADD AX, BX

        POP BX
        ADC DX, BX; ; PREVIOUS MULT RESULT - AX PART

        POP BP

        RET 4
FCTN1 ENDP
```

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