# **Matthew Mohler**

ECE 427: Microcomputer Architecture Lab

Experiment #2

**BCD & ASCII Addition and Subtraction** 

Performed: 9/17/2013

Submitted: 10/1/2013

# **Objective:**

- To code Assembly language instructions to perform addition and subtraction on packed BCD numbers
- To code Assembly language instructions to perform addition and subtraction on ASCII data
  - Part 1: Using DEBUG, write Assembly language instructions to add two BCD numbers and store the result in BCD.
  - *Part 2:* Using DEBUG, write Assembly language instructions to subtract two BCD numbers and store the result in BCD.
  - *Part 3:* Write a subroutine to add two 10-digit ASCII numbers and store the result in ASCII.
  - Part 4: Write a subroutine to subtract two 10-digit ASCII numbers and store the result in ASCII.
  - Part 5: Write a subroutine to add two 10-digit BCD numbers and store the result in BCD.
  - Part 6: Write a subroutine to subtract two 10-digit BCD numbers and store the result in BCD.

# **Word Description:**

In this lab, gained practical experience with summing and subtracting BCD and ASCII Numbers.

## **Code & Screenshots:**

#### Part 1:

MOV AL, 45 MOV BL, 25 ADD AL, BL DAA

```
_ 🗆 ×
 Command Prompt - debug
0AF6:0102 MOU BL,25
0AF6:0104 ADD AL,BL
0AF6:0106 DAA
0AF6:0107 ^C
AX=0045 BX=0000
DS=0AF6 ES=0AF6
0AF6:0102 B325
                           CX=0000 DX=0000
SS=0AF6 CS=0AF6
                                                       SP=FFEE
IP=0102
                                                                     BP=0000 SI=0000 DI=0000
                                                                      NU UP EI PL NZ NA PO NC
                                     MOU
                                                 BL.25
AX=0045
DS=0AF6
             BX =0025
ES =0AF6
                           CX=0000 DX=0000 SP=FFEE
SS=0AF6 CS=0AF6 IP=0104
                                                                     BP=0000 SI=0000 DI=0000
NU UP EI PL NZ NA PO NC
0AF6:0104 00D8
                                     ADD
                                                 AL.BL
AX=006A BX=0025
DS=0AF6 ES=0AF6
0AF6:0106 27
                           CX=0000 DX=0000
SS=0AF6 CS=0AF6
                                                       SP=FFEE
IP=0106
                                                                     BP=0000 SI=0000 DI=0000
                           SS=ØAF6
DAA
                                                                      NU UP EI PL NZ NA PE NC
                           CX=0000 DX=0000
SS=0AF6 CS=0AF6
AX=0070 BX=0025
DS=0AF6 ES=0AF6
                                                     SP=FFEE
                                                                     BP=0000 SI=0000 DI=0000
                           SS=ØAF6
JZ
                                                       IP=0107
                                                                      NU UP EI PL NZ AC PO NC
0AF6:0107 7405
                                                 010E
```

## Part 2:

MOV AL, 45 MOV BL, 25 SUB AL, BL DAS

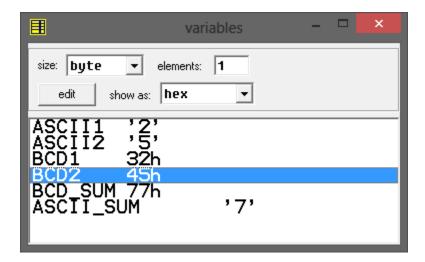
```
_ 🗆 ×
 Command Prompt - debug
0AF6:0102 MOU BL,25
0AF6:0104 SUB AL,BL
0AF6:0106 DAS
0AF6:0107 ^C
AX=0045 BX=0000
DS=0AF6 ES=0AF6
                                 CX=0000 DX=0000 SP=FFEE
SS=0AF6 CS=0AF6 IP=0102
                                                                                     BP=0000 SI=0000 DI=0000
                                                                                       NU UP EI PL NZ NA PO NC
0AF6:0102 B325
                                             MOU
AX=0045 BX=0025
DS=0AF6 ES=0AF6
0AF6:0104 28D8
                                  CX=0000 DX=0000 SP=FFEE
SS=0AF6 CS=0AF6 IP=0104
SUB AL,BL
                                                                                       3P=0000 SI=0000 DI=0000
NU UP EI PL NZ NA PO NC
                                                                                      BP=0000
AX=0020 BX=0025
DS=0AF6 ES=0AF6
0AF6:0106 2F
                                  CX=0000 DX=0000
SS=0AF6 CS=0AF6
DAS
                                                                    SP=FFEE
IP=0106
                                                                                     BP=0000 SI=0000 DI=0000
NU UP EI PL NZ NA PO NC
                                 CX=0000 DX=0000 S
SS=0AF6 CS=0AF6 D
JZ 010E
                                                                    SP=FFEE
IP=0107
                                                                                     BP=0000 SI=0000 DI=0000
NU UP EI PL NZ NA PO NC
AX=0020 BX=0025
DS=0AF6 ES=0AF6
0AF6:0107 7405
```

# Part 3 & 5:

```
;ASCII_SUM Holds the value of the sum in ASCII, while BCD_SUM holds the value of the sum in BCD
org 100h
.DATA
ASCII1 DB '2345623456'
ASCII2 DB '5432154321'
BCD1 DB 5 DUP(?)
BCD2 DB 5 DUP(?)
BCD SUM DB 5 DUP(?)
ASCII_SUM DB 10 DUP(?)
.code
MAIN PROC FAR
  MOV AX, @DATA
  MOV DS, AX
  MOV BX, OFFSET ASCII1
  MOV DI, OFFSET BCD1
  MOV CX, 10
  CALL ASCII TO BCD
  MOV BX, OFFSET ASCII2
  MOV DI, OFFSET BCD2
  MOV CX, 10
  CALL ASCII_TO_BCD
  CALL\ ADD\_BCD
  MOV SI, OFFSET BCD SUM
  MOV DI, OFFSET ASCII_SUM
  MOV CX, 05
  CALL BCD TO ASCII
  MOV AH,4CH
  INT 21H
MAIN ENDP
ASCII_TO_BCD PROC
  ;Convert ASCII to BCD
  AGAIN: MOV AX,[BX]
  AND AX,0F0FH
  PUSH CX
  MOV CL,4
  SHL AH,CL
  OR AL, AH
  MOV [DI],AL
  ADD BX,2
  INC DI
  POP CX
  LOOP AGAIN
  RET
ASCII_TO_BCD ENDP
ADD_BCD PROC
```

```
MOV BX, OFFSET BCD1
 MOV DI, OFFSET BCD2
 MOV SI, OFFSET BCD_SUM
 MOV CX, 05
  CLC
  BACK: MOV AL,[BX] + 4
 ADCAL, [DI] + 4
 DAA
 MOV[SI] + 4, AL
 DEC BX
 DEC DI
 DEC SI
 LOOP BACK
 RET
ADD_BCD ENDP
BCD_TO_ASCII PROC
 AGAIN2: MOV AL, [SI]
 MOV AH, AL
 AND AX, OFOOFH
 PUSH CX
 MOV CL, 04
 SHR AH, CL
 OR AX, 3030H
 XCHG AH, AL
 MOV [DI], AX
 INC SI
 ADD DI,2
 POP CX
 LOOP AGAIN2
 RET
BCD_TO_ASCII ENDP
```

**END MAIN** 



# Part 4 & 6:

org 100h

.DATA
ASCII1 DB '6543265432'
ASCII2 DB '5432154321'
BCD1 DB 5 DUP(?)
BCD2 DB 5 DUP(?)
BCD\_SUM DB 5 DUP(?)
ASCII\_SUM DB 10 DUP(?)

.code MAIN PROC FAR MOV AX, @DATA MOV DS, AX MOV BX, OFFSET ASCII1 MOV DI, OFFSET BCD1 MOV CX, 10 CALL ASCII\_TO\_BCD **MOV BX, OFFSET ASCII2** MOV DI, OFFSET BCD2 MOV CX, 10 CALL ASCII TO BCD CALL SUB\_BCD MOV SI, OFFSET BCD\_SUM MOV DI, OFFSET ASCII\_SUM MOV CX, 05 CALL BCD\_TO\_ASCII MOV AH,4CH INT 21H

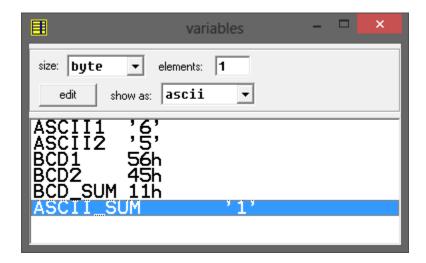
ASCII\_TO\_BCD PROC
;Convert ASCII to BCD
AGAIN: MOV AX,[BX]
AND AX,0F0FH
PUSH CX
MOV CL,4
SHL AH,CL
OR AL,AH
MOV [DI],AL
ADD BX,2
INC DI
POP CX
LOOP AGAIN
RET
ASCII\_TO\_BCD ENDP

**MAIN ENDP** 

SUB\_BCD PROC MOV BX, OFFSET BCD1

```
MOV DI, OFFSET BCD2
  MOV SI, OFFSET BCD_SUM
  MOV CX, 05
  CLC
  BACK: MOV AL,[BX] + 4
 SBB\ AL,\ [DI]+4
  DAS
  MOV[SI] + 4, AL
  DEC BX
 DEC DI
  DEC SI
 LOOP BACK
  RET
SUB_BCD ENDP
BCD_TO_ASCII PROC
 AGAIN2: MOV AL, [SI]
 MOV AH, AL
 AND AX, OFOOFH
 PUSH CX
 MOV CL, 04
 SHR AH, CL
  OR AX, 3030H
 XCHG AH, AL
  MOV [DI], AX
 INC SI
 ADD DI,2
 POP CX
 LOOP AGAIN2
  RET
BCD_TO_ASCII ENDP
```

**END MAIN** 



## **Conclusion & Comments:**

In this lab, we were able to convert ASCII numbers to Packed BCD, and then perform simple mathematical operations on the numbers (addition and subtraction). We also gained experience with the DAA and DAS operations, offsetting the values as needed to return a BCD value, rather than a hex value. We also were introduced to Processes in assembly, and how to use them effectively.

TASK 2	Name:
Worksheet	Date:
	Section:

1- Demonstrate your understanding of the different formats ASCII, binary and BCD by filling in the corresponding formats for each of the decimal numbers below in the following table.

Decimal	ASCII(Hex)	Binary	BCD
0	30	00110000	00000000
3	33	00110011	00000011
6	36	00110110	00000110
9	39	00111001	00001001
19	31 39	00010011	00011001
99	39 39	01100011	10011001

- 2- For the test data you selected for activities 1 and 2, write the result of the addition in BCD as the program stored it. Also write it in hex. Verify that the addition was done correctly.
- 3- For the test data you selected for Activities3 and 4, write the result of the subtraction in BCD. Also write it in hex. Verify that the subtraction was done correctly.
- 4- Verify the sum obtained in the addition in Activity 5. Attach a trace of the execution to show your program obtained the same result.
- 5- Verify the result obtained in the subtraction in Activity 6. Attach a trace of the execution to show your program obtained the same result.
- 6- Verify the sum obtained in the addition in Activity 7. Attach a trace of the execution to show your program obtained the same result.
- 7- Verify the result obtained in the subtraction in Activity 8. Attach a trace of the execution to show your program obtained the same result.

For parts 2-7, see lab report.