LECTURE # 10

BCD ADDITION AND SUBTRACTION

BCD Addition and Correction

- There is a problem with adding BCD numbers, which must be corrected.
- The problem is that after adding packed BCD numbers, the result is no longer BCD.

BCD Addition and Correction

- For example, MOV AL, 17H
  ADD AL, 28H
- Adding them gives 3FH, which is not BCD.
- A BCD number can only have digits from 0000 to 1001.
- In other words, adding two BCD numbers must give a BCD result.
- The result above should have been 17+28=45 (0100 0101)
- To *correct* this problem, the programmer must add 6 (0110) to the low digit: 3F + 06 = 45H.

BCD Addition and Correction

- The same problem could have happened in the upper digit.
- For example: MOV AL, 52H
  ADD AL, 87H
- Adding them gives D9H.
- Again to solve this problem, 6 must be added to the upper digit: D9 + 60H = 139H, to ensure that the result is BCD (52+87 = 139).

BCD Addition and Correction

DAA (Decimal Adjust for Addition)

- The DAA instruction in x86 microprocessor is provided exactly for the purpose of correcting the problem associated with BCD addition.
- DAA will add 6 to the lower nibble or higher nibble if needed, otherwise, it will leave the result alone.
- It is important to note that DAA works only after an ADD instruction, it will not work after the INC instruction.
- Note that the DAA works only on AL. In other words, while the source can be an operand of any addressing mode , the destination must be in AL in order for DAA to work.

BCD Addition and Correction

DAA (Decimal Adjust for Addition)

Program:

```
DATA1    DB    47H
DATA2    DB    25H
DATA3    DB    ?
         . . . .
         MOV AL, DATA1
         MOV BL, DATA2
         ADD AL, BL
         DAA
         MOV DATA3, AL
```

- After the program is executed, the DATA3 field will contain 72H (47 + 25 = 72).

BCD Addition and Correction

DAA (Decimal Adjust for Addition)

Summary of DAA action:

- If after an ADD or ADC instruction the lower nibble (4 bits) is greater than 9, or if AF = 1, add 0110 to the lower 4 bits.
- If after an ADD or ADC instruction the upper nibble (4 bits) is greater than 9, or if CF = 1, add 0110 to the upper nibble.
- For example, adding 29H and 18H will result in 41H, which is incorrect for BCD.

Hex	BCD	
29	0010 1001	
+18	0001 1000	
41	0100 0001	Because AF = 1
+ 6	+ 0110	DAA adds 6 to lower nibble
47	0100 0111	the final result is BCD

BCD Subtraction and Correction

DAS (Decimal Adjust after Subtraction)

- There is an instruction DAS specifically designed to solve the problem.
- Therefore, when subtracting packed BCD (single byte or multibyte) operands, the DAS instruction is put after the SUB or SBB instruction.
- AL must be used as the destination register to make DAS work.

BCD Subtraction and Correction

DAS (Decimal Adjust after Subtraction)

Summary of DAS instruction:

- If after a SUB or SBB instruction the lower nibble is greater than 9, or if AF = 1, subtract 0110 from the lower 4 bits.
- If after SUB or SBB instruction the upper nibble is greater than 9, or CF = 1, subtract 0110 from the upper nibble.

BCD Subtraction and Correction

DAS (Decimal Adjust after Subtraction)

Example:

```
BUDGET      DT      87965141012
EXPENSES    DT      31610640392
BALANCE     DT      ?           ; balance = budget – expenses
            MOV CX, 10
            MOV BX, 00
            CLC                 ; clear carry for first iteration
BACK:       MOV AL, BYTE PTR BUDGET [BX]
            SBB AL, BYTE PTR EXPENSES [BX]
            DAS
            MOV BYTE PTR BALANCE [BX], AL
            INC BX
            LOOP BACK
```

Two sets of ASCII data have come in from the keyboard. Write a program to:

a) Convert from ASCII to packed BCD.

b) Add multibyte packed BCD and save it.

c) Convert the packed BCD result to ASCII.

```
                .MODEL SMALL
                .STACK 64
                .DATA
DATA1_ASC       DB      '0649147816'
                ORG     10H
DATA2_ASC       DB      '0072687188'
                ORG     20H
DATA1_BCD       DB      5 DUP (?)
                ORG     30H
DATA2_BCD       DB      5 DUP (?)
                ORG     40H
DATA_ADD        DB      5 DUP (?)
                ORG     50H
ADD_ASC         DB      10 DUP (?)
                .CODE
MAIN            PROC    FAR
                MOV     AX, @DATA
                MOV     DS, AX
                MOV     BX, OFFSET DATA1_ASC
                MOV     DI, OFFSET DATA1_BCD
                MOV     CX, 10
                CALL    CONV_BCD
                MOV     BX, OFFSET DATA2_ASC
                MOV     DI, OFFSET DATA2_BCD
                MOV     CX, 10
                CALL    CONV_BCD
                CALL    BCD_ADD
                MOV     SI, OFFSET DATA_ADD
                MOV     DI, OFFSET ADD_ASC
                MOV     CX, 05
                CALL    CONV_ASC
                MOV     AH, 4CH
                INT     21H
MAIN            ENDP
```

```
; This subroutine converts ASCII to Packed BCD
CONV_BCD        PROC
AGAIN:          MOV     AX, [BX]
                XCHG    AH, AL
                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
CONV_BCD        ENDP
; This subroutine adds two multibyte Packed BCD operands
BCD_ADD         PROC
                MOV     BX, OFFSET DATA1_BCD
                MOV     DI, OFFSET DATA2_BCD
                MOV     SI, OFFSET DATA_ADD
                MOV     CX, 05
                CLC
BACK:           MOV     AL, [BX]+4
                ADC     AL, [DI]+4
                DAA
                MOV     [SI]+4, AL
                DEC     BX
                DEC     DI
                DEC     SI
                LOOP    BACK
                RET
BCD_ADD         ENDP
```

Program Contd.

```
; This subroutine converts from Packed BCD to ASCII
CONV_ASC        PROC
AGAIN2:  MOV    AL, [SI]
                MOV     AH, AL
                AND     AX, F00FH
                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
CONV_ASC        ENDP
                END     MAIN
```

Write a program that finds the number of 1s in a byte.

```
; from the data segment
DATA1 DB      97H
COUNT         DB      ?
; from the code segment
          SUB     BL, BL
          MOV     DL, 8
          MOV     AL, DATA1
AGAIN:    ROL     AL, 1
          JNC     NEXT
          INC     BL
NEXT:     DEC     DL
          JNZ     AGAIN
          MOV     COUNT, BL
```

Write a program to count the number of 1s in a word. Provide the count in BCD.

```
; from the data segment
DATA1  DW      97F4H
COUNT  DB      ?
; from the code segment
       SUB     AL, AL
       MOV     DL, 16
       MOV     BX, DATA1
AGAIN: ROL     BX, 1
       JNC     NEXT
       ADD     AL, 1
       DAA
NEXT:  DEC     DL
       JNZ     AGAIN
       MOV     COUNT, AL
```