EEE 410 – Microprocessors I

Fall 04/05 – Lecture Notes # 11

Outline of the Lecture

- **BCD Addition and correction**
- **ASCII addition and subtraction**
- **Rotate Instructions**

BCD ADDITION AND CORRECTION

```
Ex1:    MOV  AL,17H          Ex2:    MOV  AL,52H
        ADD  AL,28H                  ADD  AL,87H

        result=3FH                   result=D9H
        (not a BCD)                  (not a BCD)
```

- To solve these problems add 6 to the lower nibble of 3FH and upper nibble of D9H.

  3F + 6 = 45H D9 + 60 = 139H

- Now the results are BCD.
- There is a special instruction to do this correction.

DAA ;Decimal Adjust for Addition

DAA will add 6 to the lower or upper nibble if needed.

```
Ex:     DATA1   DB   47H
        DATA2   DB   25H
        DATA3   DB   ?

                MOV  AL,DATA1     ;AL holds the first BCD operand
                MOV  BL,DATA2     ;BL holds the second BCD  "
                ADD  AL,BL        ;BCD  addition
                DAA               ;adjust for BCD addition
                MOV  DATA3,AL     ;store result in correct BCD form
```

- DAA works only after the ADD and ADC instruction. (E.g. it doesn't work with INC instruction)
- In addition the destination operand must be AL in order for DAA to work.
- Note that in BCD addition the operands can never have any digit greater than 9.

Summary of DAA action

1. 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.
2. If the upper nibble is greater than 9, **or** CF =1, add 0110 to the upper nibble.

AF (Auxiliary carry Flag) is only used for BCD addition and correction.
Ex:

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

- **BCD Subtraction and correction**

DAS ; Decimal Adjust for Subtraction

- The problem associated with the addition of packed BCD numbers also shows up in subtraction. DAS is used to correct this problem.
- DAS must come after SUB or SBB instructions.
- AL must be used as the destination register in subtraction for the DAS to work

Summary of DAS action

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

ASCII ADDITION AND SUBTRACTION

AAA: ASCII Adjust after Addition
AAS: ASCII Adjust after Subtraction

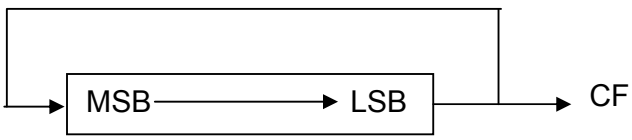
```
Ex:   MOV  AL,'5'        ;AL=35
      ADD  AL,'2'        ;add to AL 32 the ASCII for 2
      AAA                ;change 67H to 07H
      OR   AL,30         ;OR AL with 30H to get ASCII

Ex:   SUB  AH,AH         ;AH=00
      MOV  AL,'7'        ;AL=37H
      MOV  BL,'5'        ;BL=35H
      ADD  AL,BL         ;37+35=6CH  therefore AL=6C
      AAA                ;changes 6CH to 02 in AL and AH=CF=1
      OR   AX,3030H      ;AX=3132 which is the ASCII for 12H
```

Note: AAA and AAS will only work on the AL register. The data can be unpacked BCD rather than ASCII.

```
Ex:   MOV  AX,0105H      ;AX=0105 unpacked BCD for 15
      MOV  CL,06         ;CL=06H
      SUB  AL,CL         ;5-6=-1 (FFH)
      AAS                ;FFH  in AL is adjusted to 09 and
                         ;AH is decremented leaving AX=0009
```

ROTATE INSTRUCTIONS

- **ROR rotate right**



- LSB is moved to MSB and is copied to CF.
- CL holds the number of rotations. If to be rotated once 1 is used.

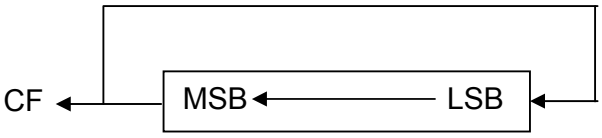
```
Ex:   MOV  AL,36H        ;AL=0011 0110
      ROR  AL,1          ;AL=0001 1011      CF=0
      ROR  AL,1          ;AL=1000 1101      CF=1
      ROR  AL,1          ;AL=1100 0110      CF=1
```

or:

```
      MOV  AL,36H        ;AL=0011 0110
```

```
        MOV  CL,3              ;CL=3 number of times to rotate
        ROR  AL,CL             ;AL=1100 0110        CF=1

Ex:     MOV  BX,C7E5H          ;BX=1100 0111 1110 0101
        MOV  CL,6              ;CL=6 number of times to rotate
        ROR  BX,CL             ;BX=1001 0111 0001 1111 CF=1
```

- **ROL rotate left**



- MSB is moved to LSB and is copied to CF.
- CL holds the number of rotations. If to be rotated once 1 is used.

```
Ex:     MOV  AL,47H            ;AL=0100 0111
        ROL  AL,1              ;AL=1000 1110        CF=0
        ROL  AL,1              ;AL=0001 1101        CF=1
        ROL  AL,1              ;AL=0011 1010        CF=0
        ROL  AL,1              ;AL=0111 0100        CF=0
```

or:

```
        MOV  AL,47H            ;AL=0100 0111
        MOV  CL,4              ;CL=4 number of times to rotate
        ROR  AL,CL             ;BH=0111 0100        CF=0
```

Ex: 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             ;clear BL to keep number of 1s
        MOV  DL,8              ;rotate total of 8 times
        MOV  AL,DATA1
AGAIN:  ROL  AL,1              ;rotate it once
        JNC  NEXT              ;check for 1
        INC  BL                ;if CF =1 than increment count
NEXT:   DEC  DL                ;go through this 8 times
        JNZ  AGAIN             ;if not finished go back
        MOV  COUNT,BL          save the number of ones
```

Ex: Write a program that finds the number of 1s in a word. Provide the count in BCD.

From the data segment:

```
DATAW1  DW   97F4H
COUNT2  DB   ?
```

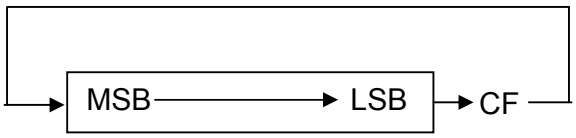
From the code segment:

```
        SUB  AL,AL             ;clear BL to keep number of 1s
        MOV  DL,16             ;rotate total of 16 times
        MOV  BX,DATAW1
AGAIN:  ROL  BX,1              ;rotate it once
        JNC  NEXT              ;check for 1
        ADD  AL,1              ;if CF =1 than add 1 to count
        DAA                    ;adjust the count for BCD
NEXT:   DEC  DL                ;go through this 8 times
```

```
        JNZ   AGAIN          ;if not finished go back
        MOV   COUNT2,AL      save the number of ones
```

Note: AL had to be used to make the BCD counter because DAA instruction works only on AL.

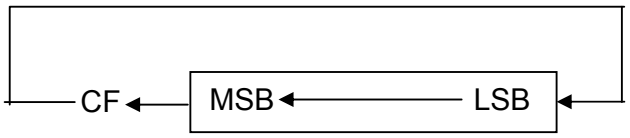
- **RCR rotate right through carry**



- LSB is moved to CF and CF is moved to MSB.
- CL holds the number of rotations. If to be rotated once 1 is used.

```
Ex:   CLC                ;clear carry, make CF=0
      MOV  AL,26H        ;AL=0010 0110
      RCR  AL,1          ;AL=0001 0011      CF=0
      RCR  AL,1          ;AL=0000 1001      CF=1
      RCR  AL,1          ;AL=1000 0100      CF=1
or:
      CLC                ;clear carry, make CF=0
      MOV  AL,26H        ;AL=0010 0110
      MOV  CL,3          ;CL=3 number of times to rotate
      RCR  AL,CL         ;AL=1000 0100      CF=1

Ex:   STC                ;set carry, make CF=1
      MOV  BX,37F1H      ;BX=0011 0111 1111 0001   CF=1
      MOV  CL=5          ;CL=5 number of times to rotate
      RCR  BX,CL         ;BX=0001 1001 1011 1111   CF=1
```

- **RCL rotate left through carry**



- MSB is moved to CF and CF is moved to LSB.
- CL holds the number of rotations. If to be rotated once 1 is used.

```
Ex:   STC                ;set carry, make CF=1
      MOV  BL,15H        ;BL=0001 0101      CF=1
      RCL  BL,1          ;BL=0010 1011      CF=0
      RCL  BL,1          ;BL=0101 0110      CF=0
or:
      STC                ;set carry, make CF=1
      MOV  BL,15H        ;BL=0001 0101      CF=1
      MOV  CL,2          ;CL=2 number of times to rotate
      RCL  BL,CL         ;BL=0010 1011      CF=0

Ex:   CLC                ;clear carry, make CF=0
      MOV  AX,191CH      ;BX=0001 1001 0001 1100   CF=0
      MOV  CL=5          ;CL=5 number of times to rotate
      RCL  AX,CL         ;AX=0010 0011 1000 0001   CF=1
```