LECTURE # 09

BCD AND ASCII CONVERSION

BCD Number system

- BCD stands for Binary Coded Decimal.
- BCD is used because we use the digits 0 to 9 for numbers in everyday life.
- Binary representation of 0 to 9 is called BCD.
- In computer literature, two terms used for BCD numbers:
  - Unpacked BCD
  - Packed BCD

Table 1: BCD Codes

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD Number system

Unpacked BCD:

- In unpacked BCD, the lower 4 bits of the number represent the *BCD number* and the rest of the bits are *zero.*
- For example: ‘0000 1001’ and ‘0000 0101’ are unpacked BCD for 9 and 5 respectively.
- In the case of unpacked BCD it takes 1 byte of memory location or a register of 8 bits to contain the number.

BCD Number system

Packed BCD:

- In the case of packed BCD, a single byte has two BCD numbers in it, one in the lower four bits and one in the upper 4 bits.
- For example: ‘0101 1001’ is packed BCD for 59.
- It takes only one byte of memory to store the packed BCD operands.
- This is one reason to use packed BCD since it is twice as efficient in storing data.

ASCII Numbers:

- In ASCII keyboards, when key ‘0’ is activated.
- For example: ‘011 0000’ (30H) is provided to the computer.
- In the same way, 31H (011 0001) is provided for key ‘1’, and so on, as shown in table:

Table 2: ASCII Numbers

Key	ASCII (Hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

ASCII to BCD Conversion

- To process the data in BCD, first the ASCII data provided by the keyboard must be converted to BCD.
- Whether it should be converted to packed or unpacked BCD depends on the instructions to be used.
- There are instructions that require that data be in unpacked BCD and there are others that must have packed BCD data to work properly.

ASCII to Unpacked BCD Conversion

- To convert ASCII data to BCD, the programmer must get rid of ‘011’ in the higher 4 bits of the ASCII.
- To do this, each ASCII number is ANDed with ‘0000 1111’ (0FH).

Example:

```
        ; from data segment
ASC     DB      ‘9562481273’
        ORG     0010H
UNPACK  DB      10 DUP (?)
        ; from code segment
        MOV CX, 5
        MOV BX, OFFSET ASC
        MOV DI, OFFSET UNPACK
AGAIN:  MOV AX, [BX]
        AND AX, 0F0FH
        MOV [DI], AX
        ADD BX, 2
        ADD DI, 2
        LOOP AGAIN
```

ASCII to Packed BCD Conversion

- To convert ASCII to packed BCD, it is first converted into unpacked BCD (to get rid of 3) and then combined to make packed BCD.
- For example, for 9 and 5 the keyboard gives 39 and 35 respectively. The goal is to produce 95H or ‘1001 0101’, which is called packed BCD.

ASCII to Packed BCD Conversion

Example:

```
        ; from data segment
                ORG 0010H
VAL_ASC         DB      ‘47’
VAL_BCD         DB      ?
; reminder: DB will put 34 in 0010H location and 37 in 0011H location
        ; from code segment
        MOV AX, WORD PTR VAL_ASC  ; AH = 37 and AL = 34H
        AND AX, 0F0FH; mask 3 to get unpacked BCD
        XCHG AH, AL               ; swap AH and AL
        MOV CL, 4                 ; to shift 4 times
        SHL AH, CL                ; shift left AH to get AH = 40H
        OR AL, AH                 ; OR them to get packed BCD
        MOV VAL_BCD, AL           ; save the result
```

Packed BCD to ASCII Conversion

- To convert packed BCD to ASCII, it must first be converted to unpacked and then the unpacked BCD is masked with ‘011 0000’ (30H).

Example:

```
VAL1_BCD    DB      29H
VAL2_ASC    DW      ?
            . . . .
            MOV AL, VAL1_BCD        ; AL = 29H
            MOV AH, AL              ; copy AL to AH
            AND AX, F00FH           ; mask 9 from AH and 2 from AL
            MOV CL, 4               ; to shift 4 times
            SHR AH, CL              ; shift right AH to get unpacked BCD
            OR AX, 3030H            ; combine with 30 to get ASCII
            XCHG AH, AL             ; swap for ASCII storage convention
            MOV VAL3_ASC, AX        ; store the result
```