Packed BCD

Data input from the Keyboard
(ASCII)

Function 10 - MS/DOS Read String

• FUNCTION 10 INT 21H Routine
• Entry Parameters:
  – Register DX: has the address of the input buffer size
  – Register AH: 0AH
  – Uses INT 21H
• Returned Value:
  – There is Screen Echo of Characters
  – The number of characters is in storage
  – The input storage buffer has a string
FUNCTION 10 INT 21H

- To use this, the function requires storage setup as follows:
  - The base address contains the number of bytes allocated for the storage buffer. In the example below, the base address is maxSize
  - The next byte (maxSize + 1) contains the number of keys pressed on return from the BIOS routine
  - The next byte (maxSize + 2) contains the string entered by the user on return from the BIOS routine.

- The data segment storage looks like the following:

```assembly
maxSize db   NumVals
keysPressed db   ?
inputString db   15 dup (?)
NumVals equ $-keysPressed
```

- To use this function once we have created storage, we would enter the following in the code segment:

```assembly
; Keyboard BIOS routine function 10 INT 21H
mov ah,0AH ;DOS function: input string
lea dx,maxSize ; the base address of storage
int 21h
```
ASCII in Decimal vs BCD

(All values are in hexadecimal)

- In the example above, unsigned BCD data are entered from the keyboard (3402) and placed in a storage buffer.
- To strip the ASCII from the data, we need to remove the upper nibble (30H).
- This can be done in a number of ways depending on whether or not we need to check for valid data.

If we know the data is valid BCD, we can:

```assembly
mov cx,4           ;load loop counter
lea si,ascii_in   ;init address pointer of data
again:
    and [si],0fH   ;mask upper nibble
    inc si        ;increment address pointer
    loop again    ;dec cx and JNZ again
; data is now in unpacked BCD
```
Packing BCD

; The following will take 4 unpacked BCD values
; return two Packed BCD values in a buffer named Packed_BCD
        mov  cl, 4 ; move number of times to rotate
        lea  si, ascii_in ; init address pointer of data
        mov  dl, 2 ; load loop counter
        lea  di, Packed_BCD ; init address pointer of result
Again:
        mov  al, [si] ; load data
        rol  al, cl ; rotate al 4 times
        add  al, [si+1] ; pack upper two values
        mov  [di], al ; store the packed number
        add  si, 2 ; increment address pointer
        inc  di
        dec  dl ; dec loop counter dl
        jnz again ; JNZ again

Adding Packed BCD

; The following will add 2 packed BCD values and
; return the sum in a buffer named Packed_Sum
        lea  si, Packed_BCD ; init address pointer of data
        mov  al, [si] ; get the first packed BCD value
        add  al, [si+1] ; add the second value
        daa ; convert SUM to back to BCD
        mov  [SI], al ; store the sum in a buffer
UnPacking BCD and converting to ASCII

```assembly
mov al, [Packed_Sum] ; al <= the Packed BCD Sum
mov ah, al ; copy to ah
and 0F00FH ; AH has tens digit, AL has the units
mov cl, 4 ; number of times to rotate 10s digit
rol ah, cl ; move tens digit to low nibble
add 3030h ; make ah and al ASCII values
mov [rslt+1], ah ; store ASCII 10 digit in rslt+1
mov [rslt+2], al ; store ASCII units digit in rslt+2
```

; Note this routine does not allow for overflow from the byte sum.
; This would have to be handled immediately after the DAA
; If there was a carry, place ASCII for 1 in rslt
; If there was no carry, place ASCII for SPACE in rslt