PORTS

The 80xxx family communicates with hardware devices by means of PORTS

A PORT is an ADDRESS where an I/O device is connected
Data is SENT TO the device by sending it to the port
Data is received/INPUT FROM the device by reading it from the port

A PORT ADDRESS can be any value between 0000H and FFFFH

Instructions for dealing with ports:

```
        IN      Accum, Immed8
```

Accum MUST be AL or AX
If AL then a byte is transferred
IF AX then a word is transferred

Transfer data FROM 8-bit PORT address given by Immed8 operand TO Accum. For ports between 00H and FFH

```
        IN      Accum, dx
```

Transfer data FROM 16-bit PORT address given in DX operand TO Accum. This is faster if we ignore time to load DX.

```
        OUT     Immed8, Accum
        OUT     dx, Accum
```

Transfer data TO PORT address given in Immed8/DX operand FROM Accum.

Before we start moving things to/from ports we need to set it up. We need to know about the interface chip to do this.

The Lab boards have two 8255A-5 Peripheral Interface Adapters (interface chip) from Intel.

One chip (the BOARD 8255) is connected to the LEDs, Speaker, keypad, etc.

The other (the USER 8255) is for the user to use for custom projects.

The Board 8255 is divided into 3 8-bit ports.

PORT A: LEDS
PORT B: later
PORT C: Keypad

Before you use the CHIP to flash LEDS, you need to tell it that PORT A is an output port, Port B is also output and Port C is (half) input.

See figure 1-1 in Interface Manual.

Each PORT on the Board's 8255 is either INPUT or OUTPUT with no handshaking (never mind what handshaking is :-)).

This is MODE 0 for the 8255

According to fig 1-1 we want to tell the chip that we want:

PORT A to be OUTPUT
PORT B to be OUTPUT
Lower Nibble (LSN) of PORT C to be OUTPUT
Upper Nibble (MSN) of PORT C to be INPUT

Look at INTEL 8085 data sheets
Appendix D in the Interface Manual

To tell the chip we want this configuration
We send the control byte 88h to port 303h

```
        mov     DX, 303h
        mov     AL, 88H
        OUT     DX, AL
```

We could also use TVC to output 88h to port 303h
You need to get into privelage mode first
then
>> priv
-- O 303 88
--
should cause all LEDS to light up

Why Port 303?

The board is set up that way.
In hex:

300	Port A
301	Port B
302	Port C
303	Mode Port

Sending the Mode control Byte to the mode port sets up the chip the way we want it.

How do we make the chip flash the LEDS

Sending a Byte to PORT A (300h) will cause the LEDS specified by the PORT to Light up.

Bit 0	corresponds to	Rightmost LED
.		
.		
.		
Bit 7	corresponds to	Leftmost LED

The LEDS are connected so that a logic 0 turns them on while a Logic 1 turns them off.

-- O 300 FF

will turn off all LEDS

-- O 300 FE will turn on only the first LED

```
TITLE   flash1.asm

DOSSEG
.MODEL  SMALL

.STACK  100h

.DATA

.CODE

START:
        mov     ax, @DATA
        mov     ds, ax

        mov     dx, 303h
        mov     al, 88h
        out     dx, al

        mov     al,-1
        mov     dx, 300h
        out     dx, al

        mov     al, -1
        mov     bl, 1

TOP:
        xor     al, bl
        out     dx, al
```

```
        mov     cx, 0
l1:     loop    l1
                                ; A delay Loop, loop doing nothing
                                ; for a while


        mov     cx, 0
l2:     loop    l2
                                ; Without Delays, the LEDS flash
                                ; too fast
                                ; Just kinda flicker on and off

        shl     bl,1

        ; shift the contents of bl left by 1
        ; shl   Op1, 1  shift left by 1
        ; shl   Op1, cl shift left by count in cl
        ; must use cl no other register

        jnz     TOP

        CALL    EXIT

END     START
```

```
TITLE   as4.asm
...

EXTRN  EXIT:NEAR

START:
        mov     ax, @DATA
        mov     ds, ax

        CALL    INIT

        ....

CNTR:

        CALL    DELAY

        CALL    LEDOFF

        inc     bl

        CALL    LEDCNT

        loop    CNTR

        CALL    EXIT
```

```
;------------------------------------------

PROC    LEDCNT          NEAR

        push    ax      ; Save AX
        push    dx      ; Save DX

        ...

        pop     dx
        pop     ax

LEDCNT          ENDP

        ...; other procedures...

END     START
```