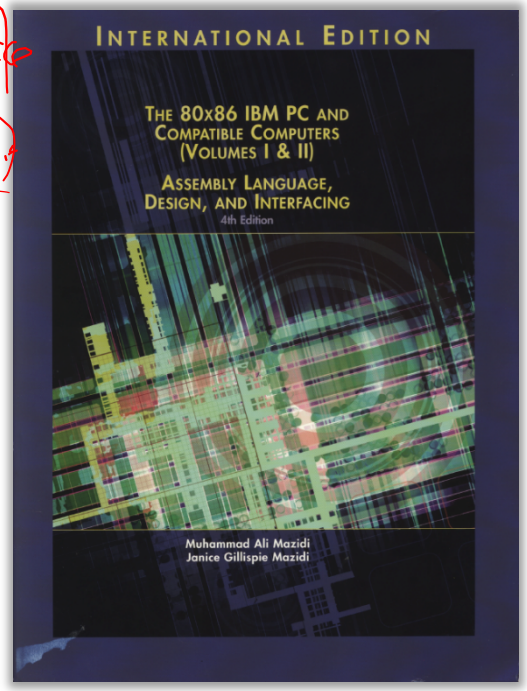
Lecture 07: 8255 PPI Chip

8255

Reference Book:

The 80x86 IBM PC and Compatible Computers

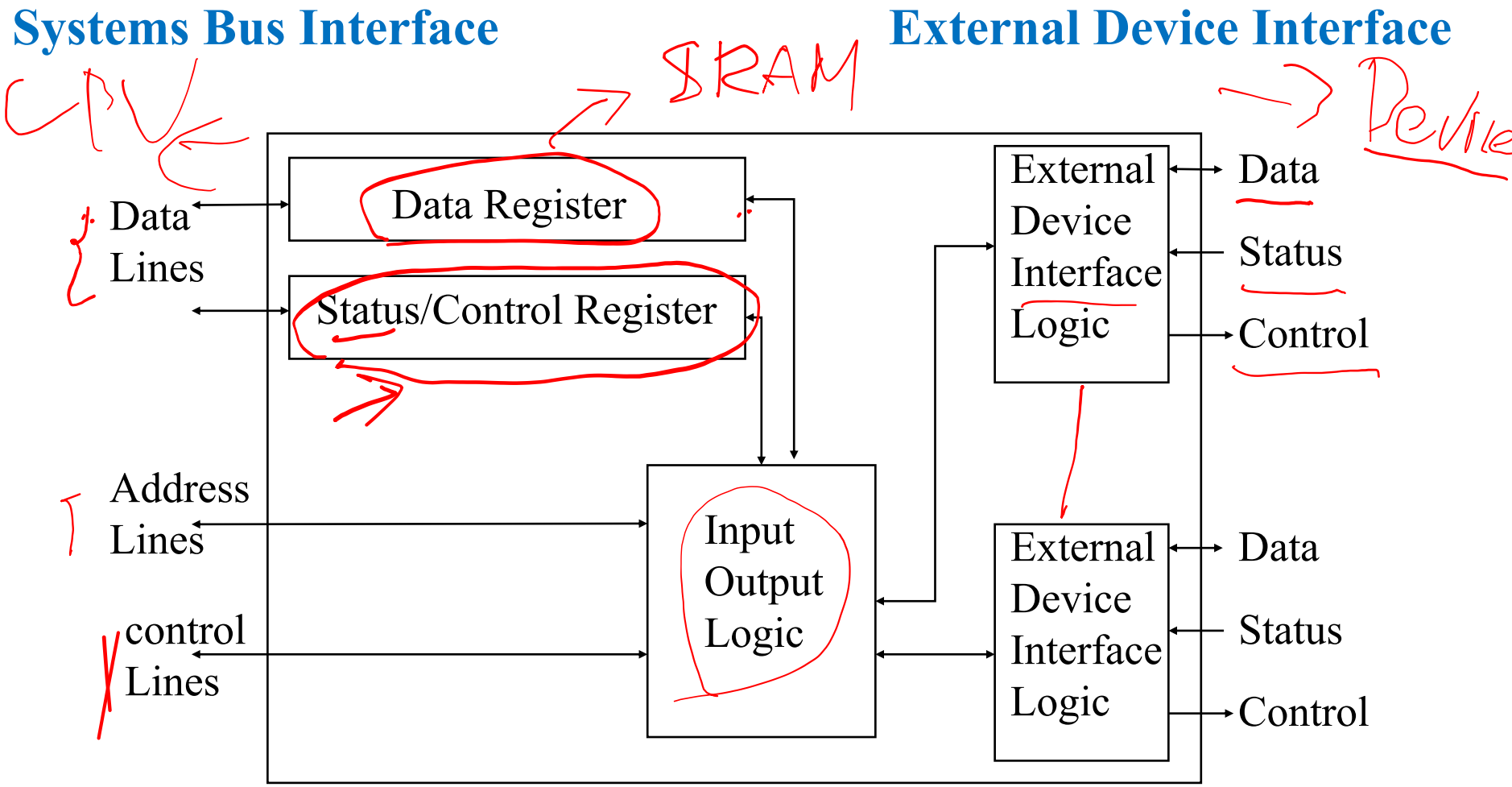


Chapter 11.4

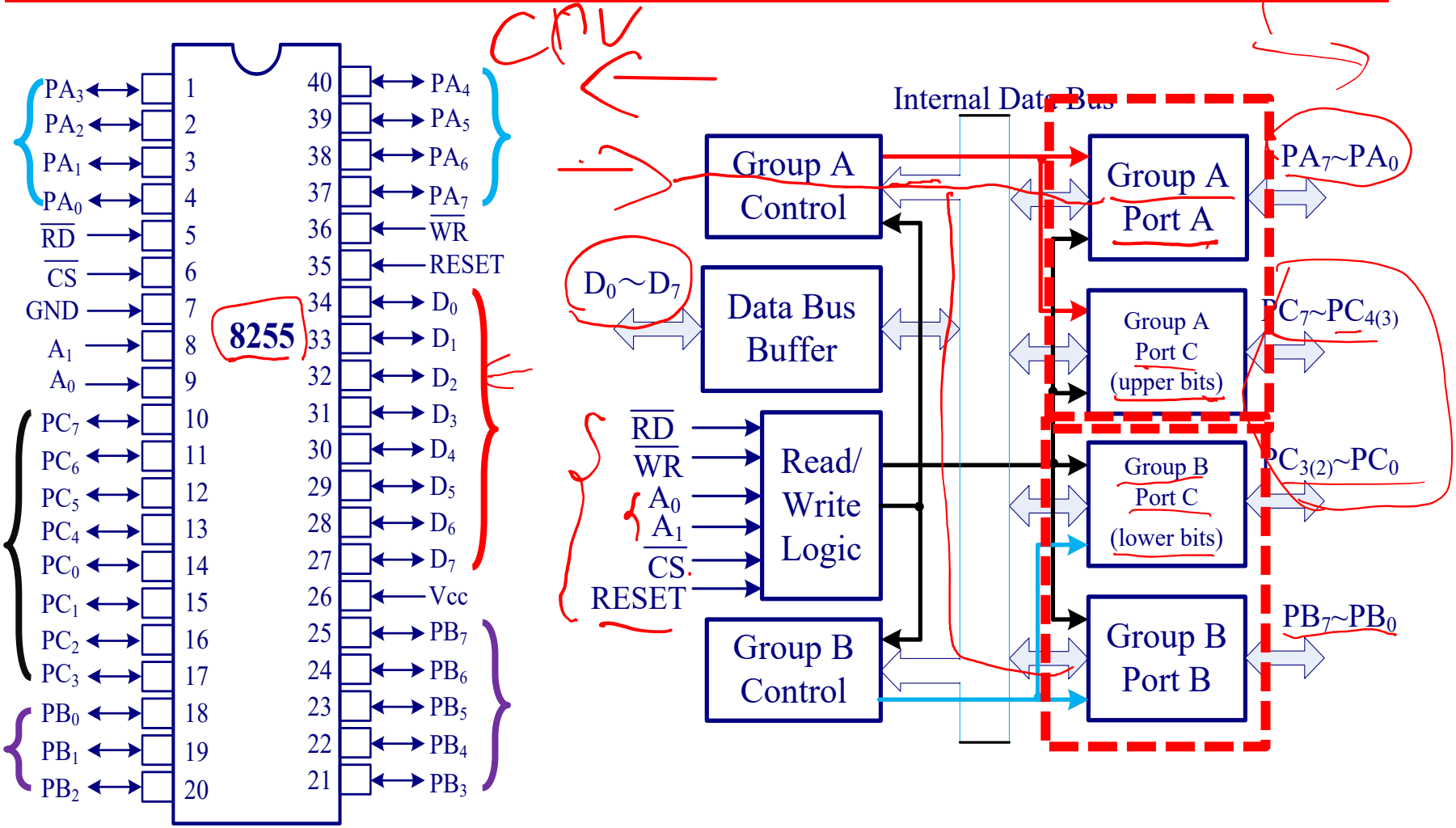
8255 PPI Chip

PPI: Programmable Parallel Interface (so it is an I/O module)

Recall in Lecture 02: I/O Module Diagram



Package & Internal Structure



Internal Structure and Pins

- Three data ports: A, B, and C
  - Port A (PA_0~PA_7): can be programmed **all** as input/output
  - Port B (PB_0~PB_7): can be programmed **all** as input/output
  - Port C (PC_0~PC_7): can be split into two separate parts *PCU* and *PCL*; any bit can be programmed **individually**
- Control register (CR)
  - Internal register: used to setup the chip
- Group A, Group B and control logic
  - Group A (PA & PCU)
  - Group B (PB & PCL)

Internal Structure and Pins

- Data bus buffer
  - An interface between CPU and 8255
  - Bidirectional, tri-state, 8-bit
- Read/Write control logic
  - Internal and external control signals
  - **RESET**: high-active, clear the control register, all ports are set as input port
  - **~CS**, **~RD**, **~WR**
  - **A_1, A_0:** port selection signals

~CS	A_1	A_0	~RD	~WR	Function
0	0	0	0	1	PA->Data bus
0	0	1	0	1	PB->Data bus
0	1	0	0	1	PC->Data bus
0	0	0	1	0	Data bus->PA
0	0	1	1	0	Data bus->PB
0	1	0	1	0	Data bus->PC
0	1	1	1	0	Data bus->CR
1	×	×	1	1	D_0~D_7 in float

Operation Modes

- Input/output modes
  - Mode 0, simple I/O mode:
    - **PA**, **PB**, **PC**: PCU{PC_4~PC_7}, PCL{PC_0~PC_3}
    - No Handshaking: *negotiation between two entities before communication*
    - Each port can be programmed as input/output port
  - Mode 1:
    - **PA**, **PB** can be used as input/output ports with *handshaking*
    - PCU{PC_3~PC_7}, PCL{PC_0~PC_2} are used as handshake lines for PA and PB, respectively
  - Mode 2:
    - Only **PA** can be used for *bidirectional handshake* data transfer
    - PCU{PC_3~PC_7} are used as handshake lines for PA
- Bit set/reset (BSR) mode
  - Only **PC** can be used as output port
  - Each line of PC can be set/reset individually

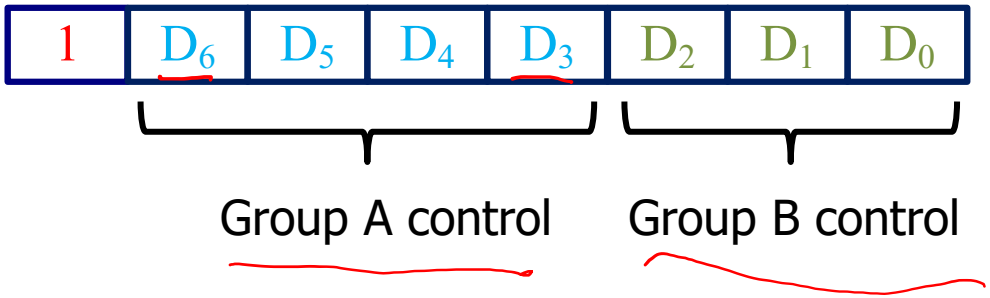
Control Register & Op. Modes

- Control Register
  - A 8-bit internal register in 8255
  - Selected when A_1=1, A_0=1
  - Mode selection word
    - Input/output modes

    - BSR mode

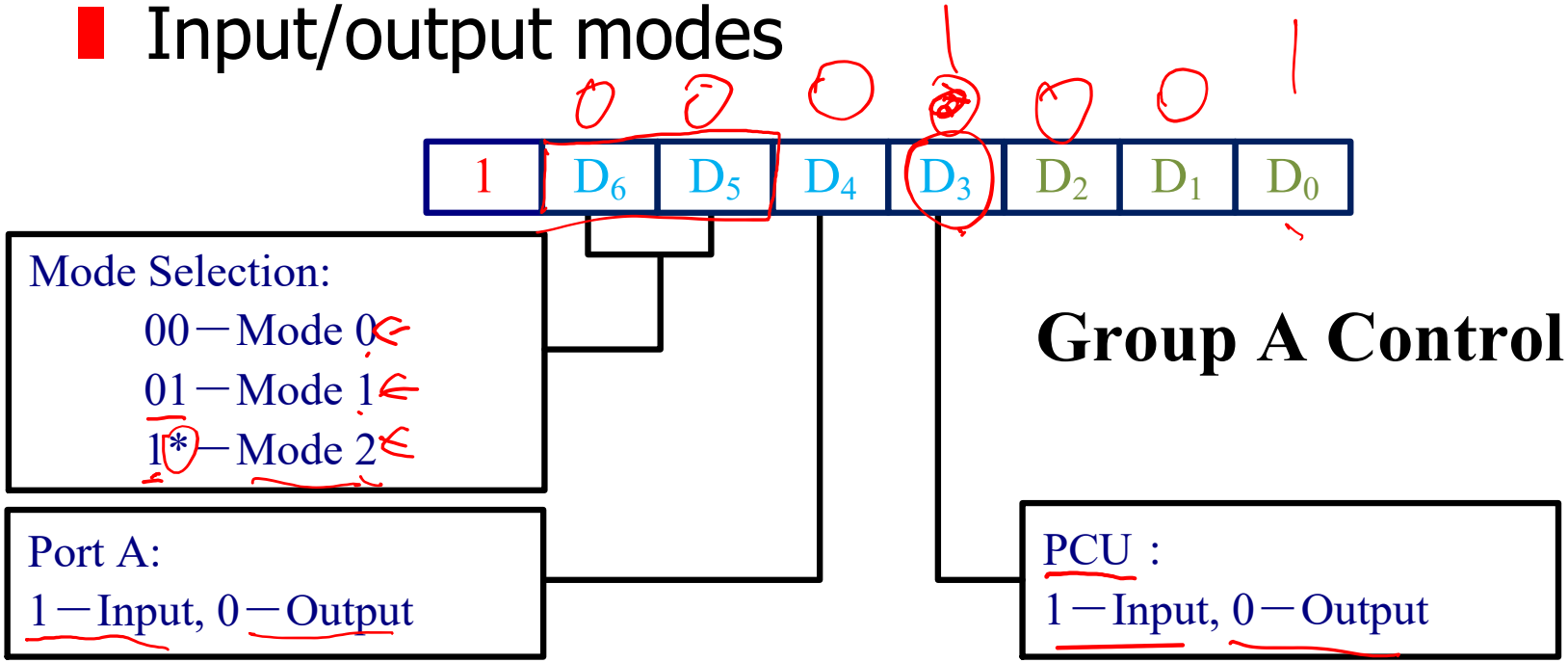
Select Input/output Modes

- Input/output modes



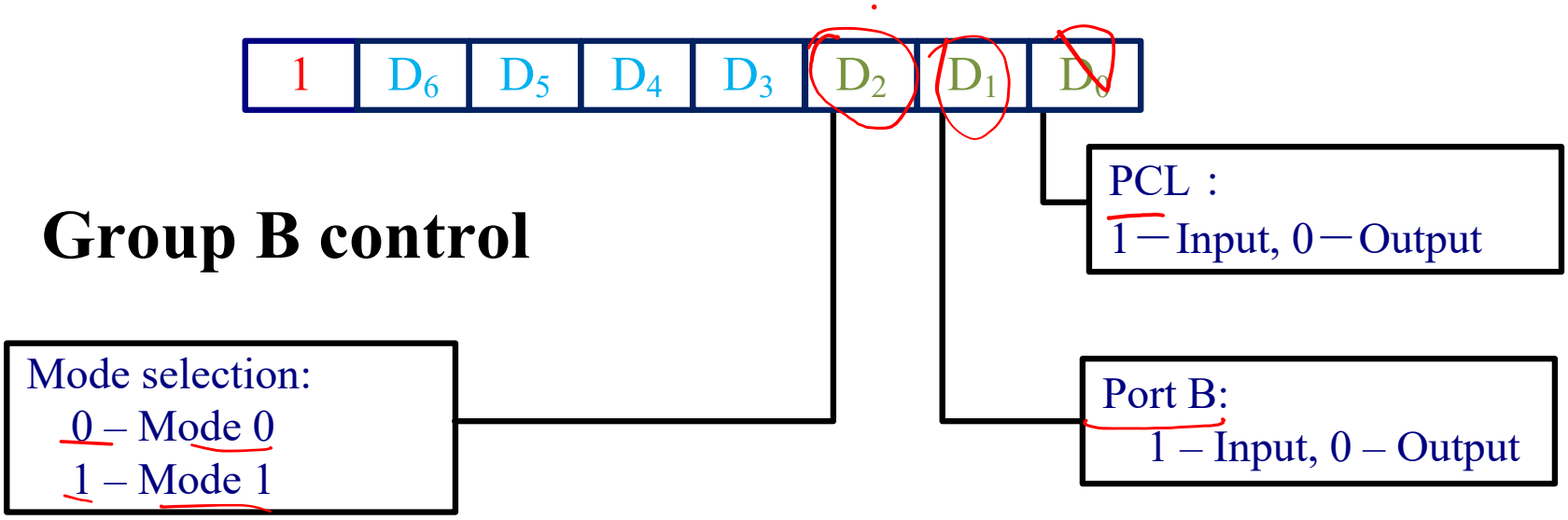
Select Input/output Modes

- Input/output modes



Select Input/output Modes

- Input/output modes



Select Input/output Mode Examples

- 1. Write ASM instructions for setting the 8255 in simple I/O mode with PA and PB being output port and PC being input port.

```
MOV  AL, 10001001B
MOV  DX, ControlPort
OUT  DX, AL
```

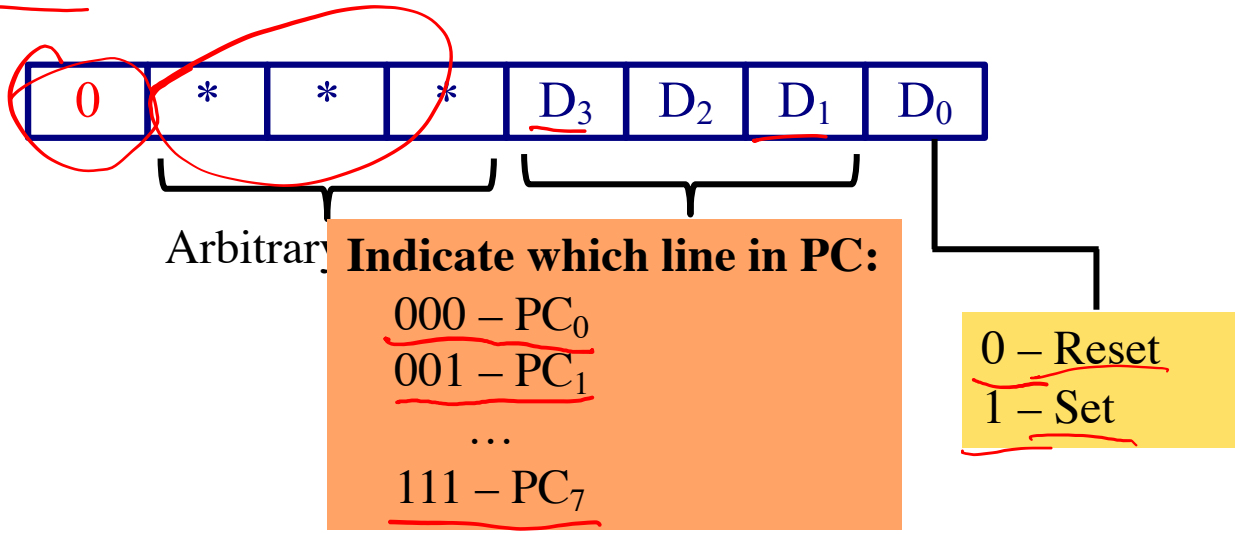
- 2. Assume that the address of the control register of the 8255 is 63H, give out the instructions that set up the 8255 in mode 0 where PA, PB and PCU are used as input ports and PCL is used as output port.

```
MOV  AL, 10011010B
OUT  63H, AL
```

10011010

Select BSR Mode

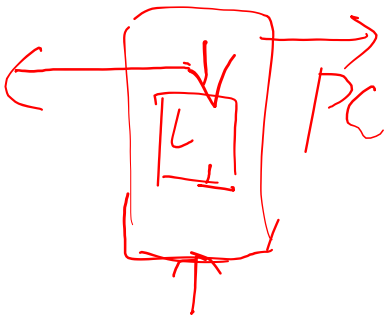
- BSR Mode



Select BSR Mode Examples

- Assume PC is used as an output port which connects to 8 LED segments, now turn off the second LED segment with the rest unchanged (e.g., for LED segments in this case, 1-on, 0-off).

```
MOV AL, 10000000
OUT  ControlPort, AL
...
IN  AL, CPORT
AND  AL, 11111101B
OUT   CPORT, AL
```



- Using BSR mode instead:

```
MOV  AL, 00000010B
OUT   ControlPort, AL
```

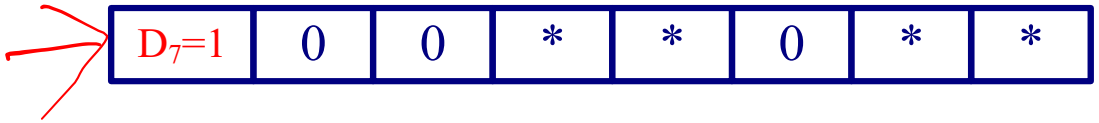
Select BSR Mode Examples

- Assume the address range for a 8255 is 60H~63H, PC_5 is outputting a low level, write code to generate a positive pulse.

```
MOV  AL, 00001011B     ; set PC5 high level
OUT  63H, AL
MOV  AL, 00001010B     ; set PC5 low level
OUT  63H, AL
```

Mode 0 (Simple I/O)

- For simple input/output scenario
  - No handshaking needed
  - Any port of PA, PB and PC (PCU, PCL) can be programmed as input or output port independently
  - PCU=PC_4~PC_7, PCL=PC_0~PC_3
  - CPU directly read from or write to a port using IN and OUT instructions
  - Input data are **not** latched, Output data are latched
  - E.g., setting up the control register for Mode 0



Mode 0 Example

The 8255 shown in Figure 11-13 is configured as follows: port A as input, B as output, and all the bits of port C as output.
(a) Find the port addresses assigned to A, B, C, and the control register.
(b) Find the control byte (word) for this configuration.
(c) Program the ports to input data from port A and send it to both ports B and C.

Solution:
(a) The port addresses are as follows:

$\overline{CS}$	A1	A0	Address	Port
11 0001 00	0	0	310H	Port A
11 0001 00	0	1	311H	Port B
11 0001 00	1	0	312H	Port C
11 0001 00	1	1	313H	Control register

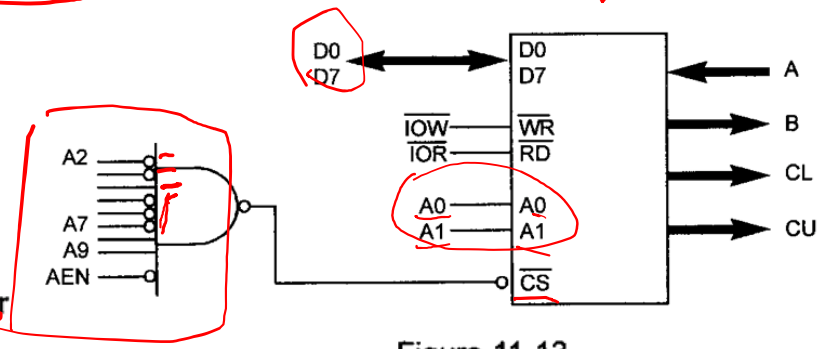


Figure 11-13

(b) The control word is 90H or 1001 0000.

(c) One version of the program is as follows:

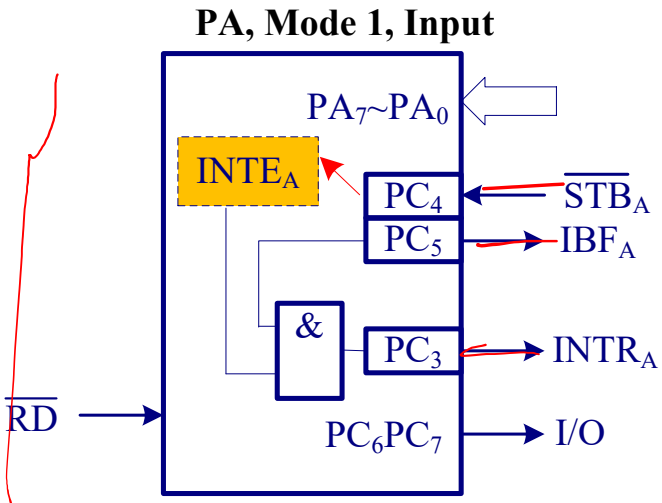
```
MOV   AL,90H      ;control byte PA=in, PB=out, PC=out
MOV   DX,313H     ;load control reg address
OUT   DX,AL       ;send it to control register
MOV   DX,310H     ;load PA address
IN    AL,DX       ;get the data from PA
MOV   DX,311H     ;load PB address
OUT   DX,AL       ;send it to PB
MOV   DX,312H     ;load PC address
OUT   DX,AL       ;and to PC
```

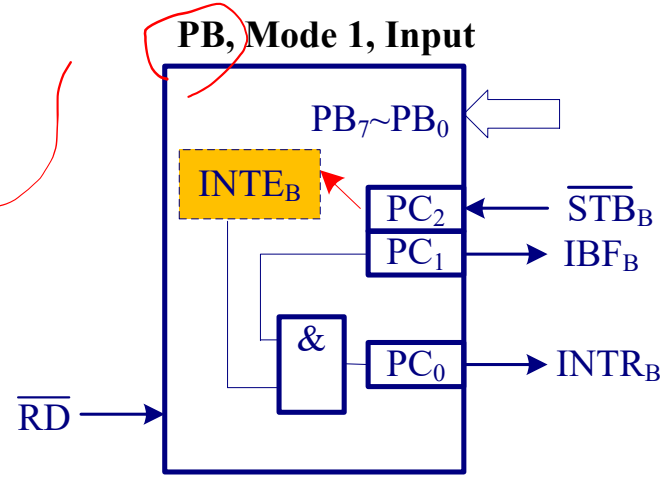
Mode 1 (Strobe I/O)

- For handshake input/output scenario
  - PA and PB can be used as input or output ports
  - PC_3~PC_5 used as handshake lines for PA
  - PC_0~PC_2 used as handshake lines for PB
  - Both input and output data are latched

Mode 1: As Input Ports

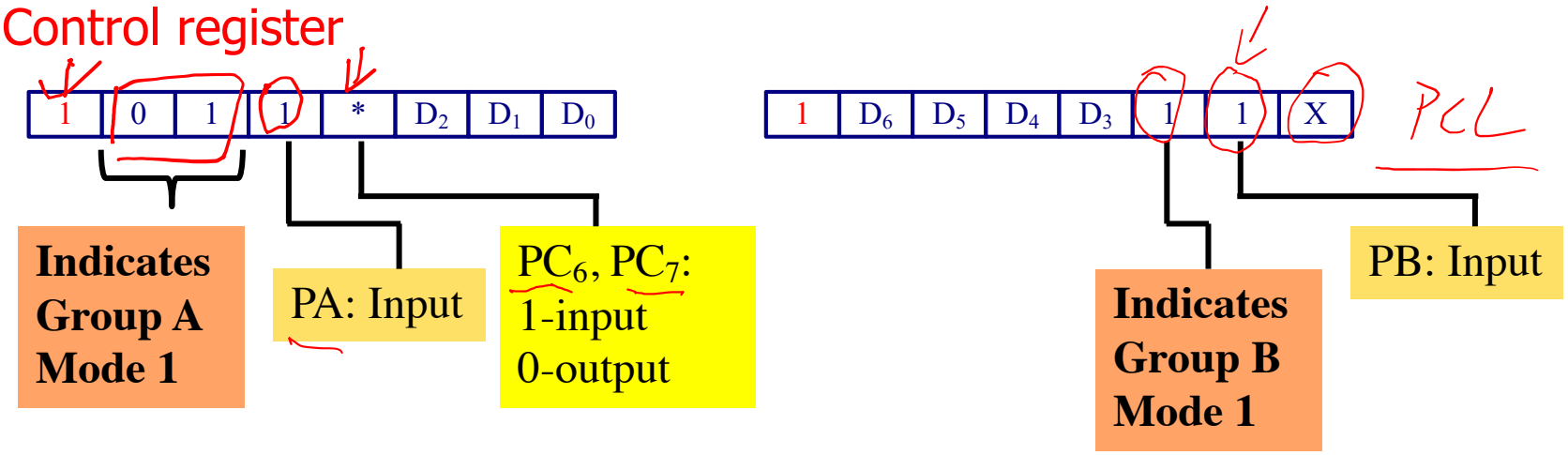
- PC_3~PC_5 and PC_0~PC_2 are used as handshake lines for PA and PB, respectively
  - **~STB:** the strobe *input* signal from input device loads data into the port latch
  - **IBF:** Input Buffer Full *output* signal to the device indicates that the input latch contains information (can also be used for programmed I/O)
  - **INTR:** Interrupt request is an output to CPU that requests an interrupts (for interrupted I/O)
- **PC_6** and **PC_7** can be used as separate I/O lines for any purpose
- **INTE:** the interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC_4 (port A) or PC_2 (port B); 1-allowed, 0-forbidden



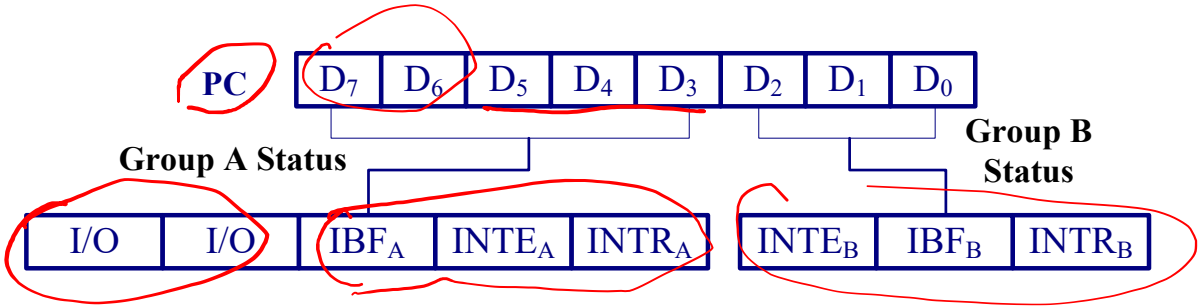


Mode 1: As Input Ports

- Control register

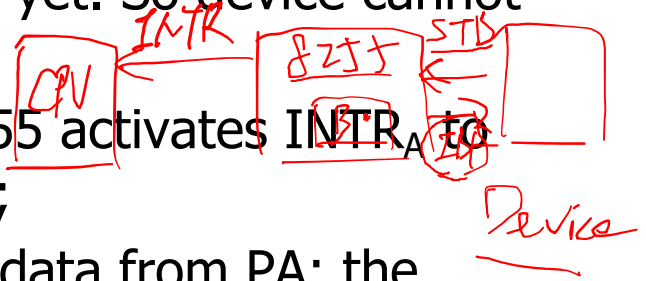


- PC stores all status information

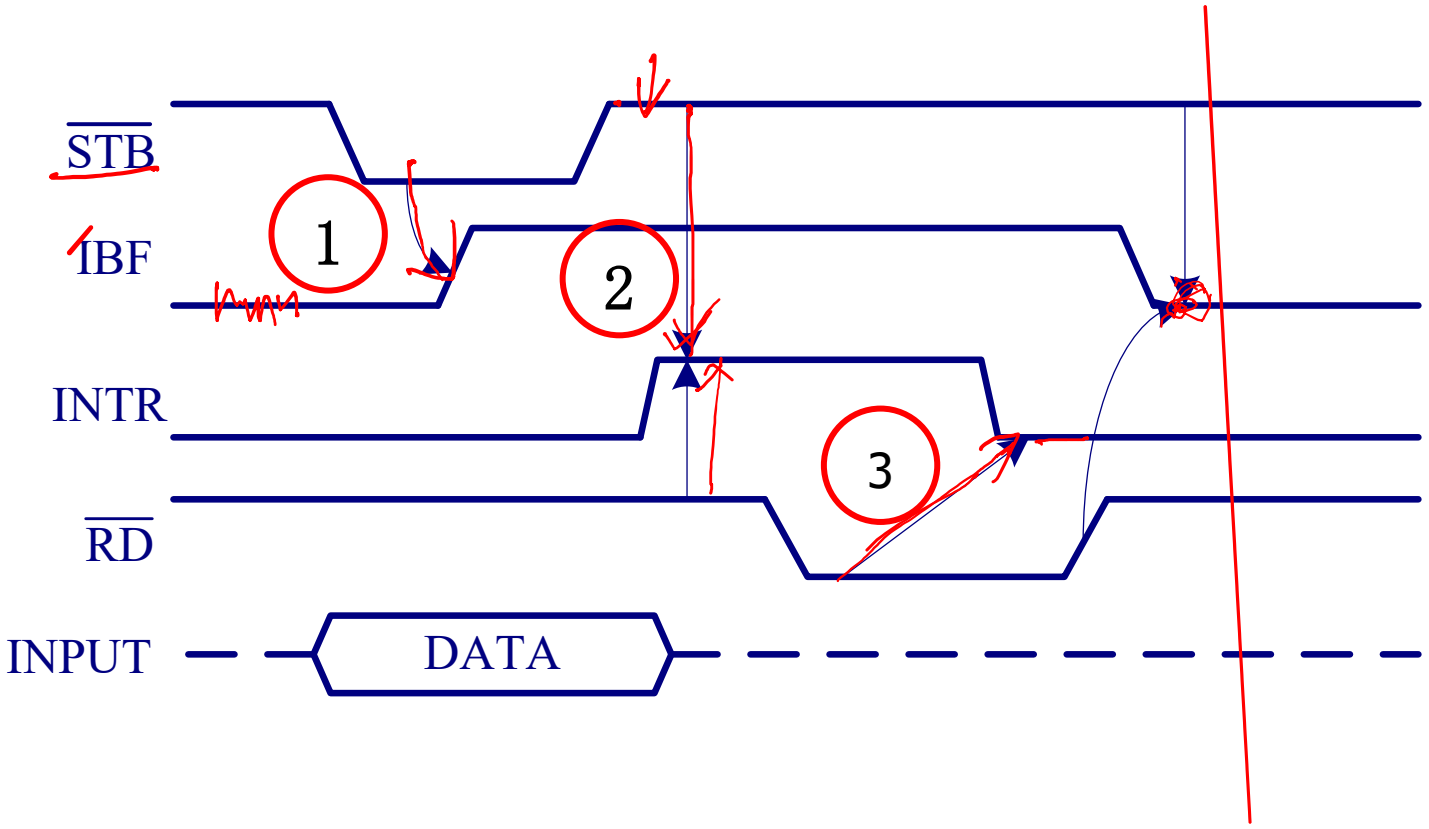


Timing in Mode 1 Input

- Input device first puts data on PA_0~PA_7, then activates ~STB_A, data is latched in Port A;
- 8255 activates IBF_A which indicates the device that the input latch contains information but CPU has not taken it yet. So device cannot send new data until IBF_A is cleared;
- When IBF_A, ~STB_A and $INTE_A$ are all high, 8255 activates $INTR_A$ to inform CPU to take data in PA by interruption;
- CPU responds to the interruption and read in data from PA; the ~RD signal will clear $INTR_A$ signal;
- After CPU finishes reading data from PA (i.e., ~RD signal goes high), the IBF_A signal is cleared.

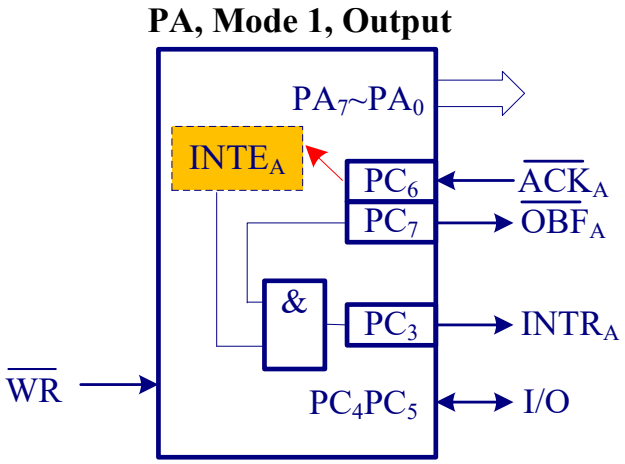


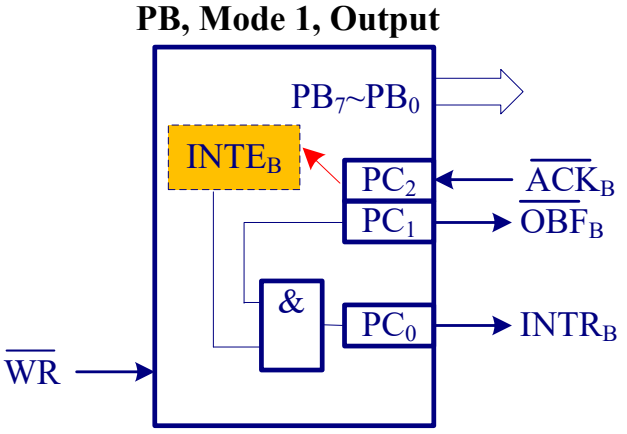
Timing in Mode 1 Input



Mode 1: As Output Ports

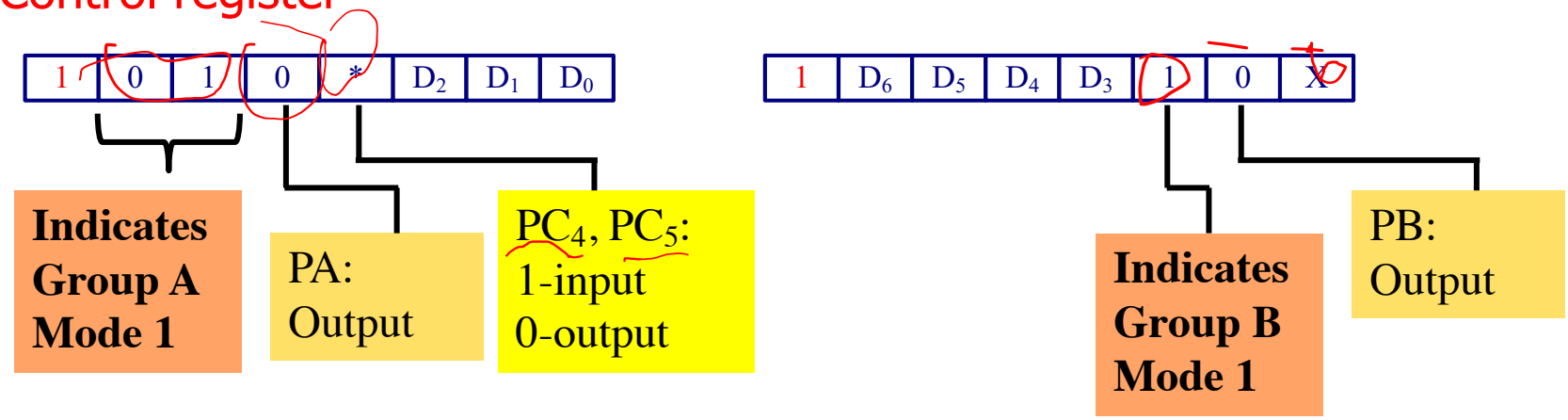
- PC_3, PC_6, PC_7 and PC_0~PC_2 are used as handshake lines for PA and PB, respectively
  - **~OBF** : Output buffer full is an *output* signal that indicates the data has been latched in the port
  - **~ACK** : The acknowledge *input* signal indicates that the external device has taken the data
  - **INTR:** Interrupt request is an output to CPU that requests an interrupts
- **PC_4** and **PC_5** can be used as separate I/O lines for any purpose
- **INTE:** the interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC_6 (port A) or PC_2 (port B); 1-allowed, 0-forbidden



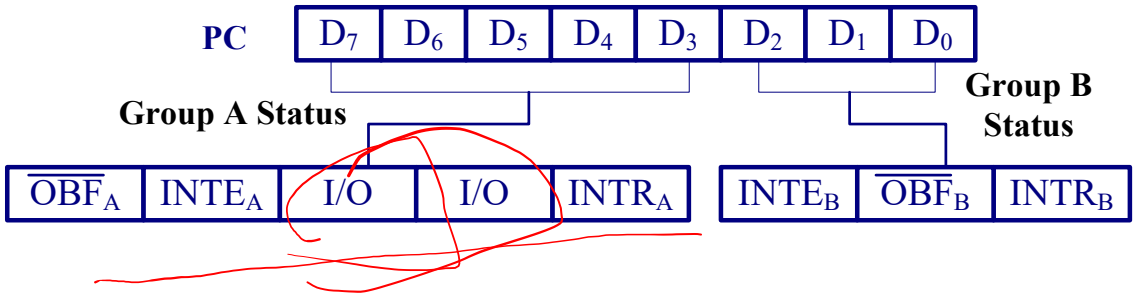


Mode 1: As Output Ports

- Control register



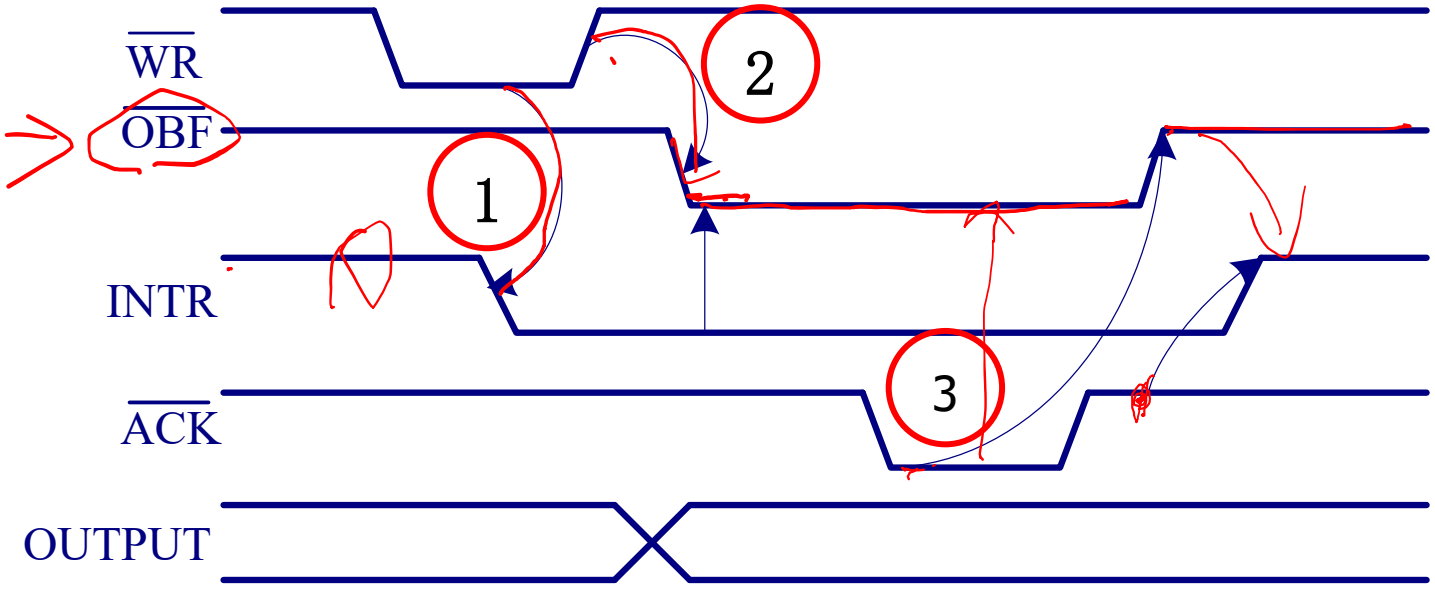
- PC stores all status information



Timing in Mode 1 Output

- If $INTR_A$ active, CPU responds to the interruption and writes data to PA and clears the $INTR_A$ signal;
- When data has been latched in PA, 8255 activates $\sim OBF_A$ which informs the output device to pick up data;
- After the output device has taken the data, it sends $\sim ACK_A$ signal to 8255 which indicates that the device has received the data, and also makes $\sim OBF_A$ go high, indicating CPU can write new data to 8255;
- When $\sim OBF_A$, $\sim ACK_A$ and $INTE_A$ are all high, 8255 sends an $INTR_A$ to inform CPU to write new data to PA by interruption.

Timing in Mode 1 Output

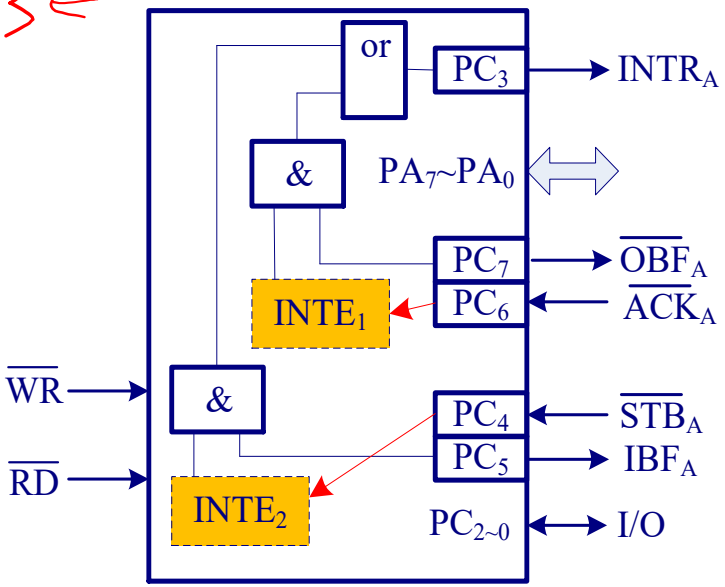


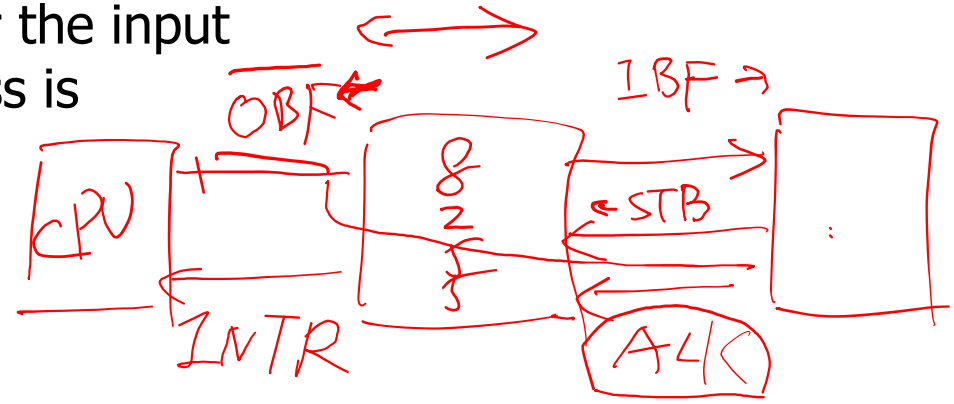
Mode 2 (Bidirectional Bus)

- For bidirectional handshake input/output scenario
  - Only PA can be used as *both input and output port*
  - PCU=PC_3~PC_7, used as handshake lines for PA
  - Both input and output data are latched

Mode 2: As Input & Output Port

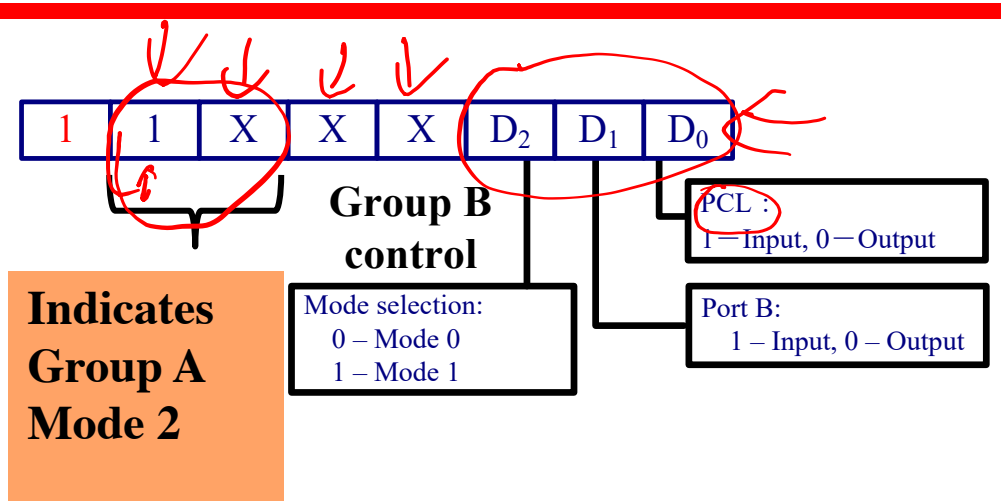
- $PC_3 \sim PC_7$ are used as handshake lines for PA
- **$\sim OBF_A$, $\sim ACK$, IBF_A, $\sim STB_A$, $INTR_A$**
- **$PC_0 \sim PC_2$** can be used as separate I/O lines for any purpose, or as handshake lines for PB
- When CPU responds to an interrupt of 8255 working in Mode 2, the interrupt handler has to check the **$\sim OBF_A$** and **IBF_A** in order to tell whether the input process or the output process is generating the interrupt.



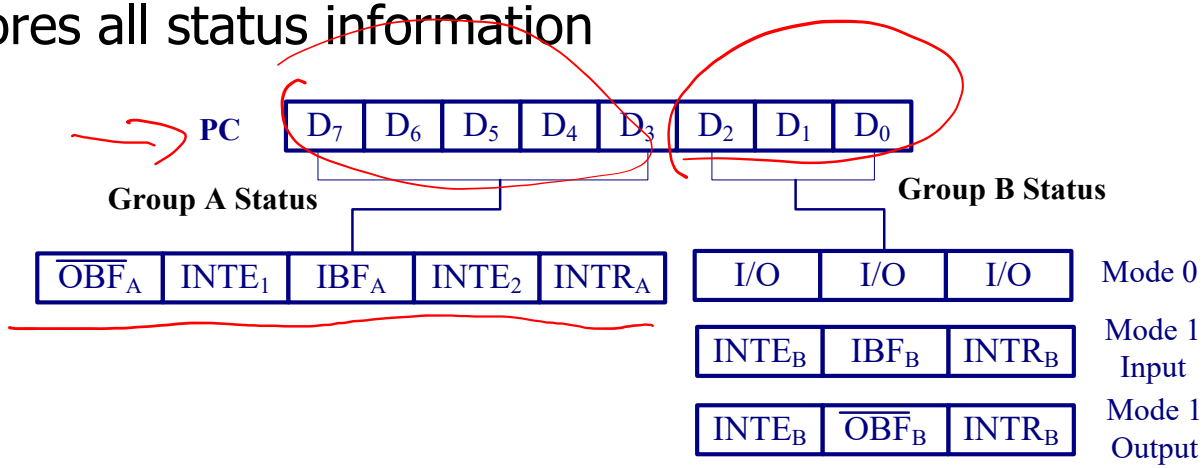


Mode 2: As Input & Output Port

- Control register

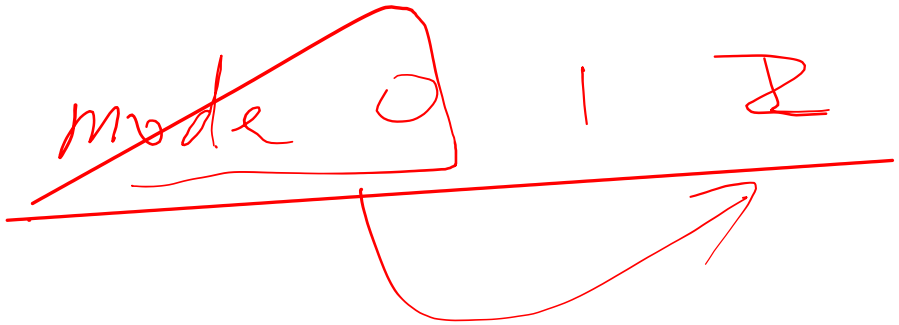


- PC stores all status information



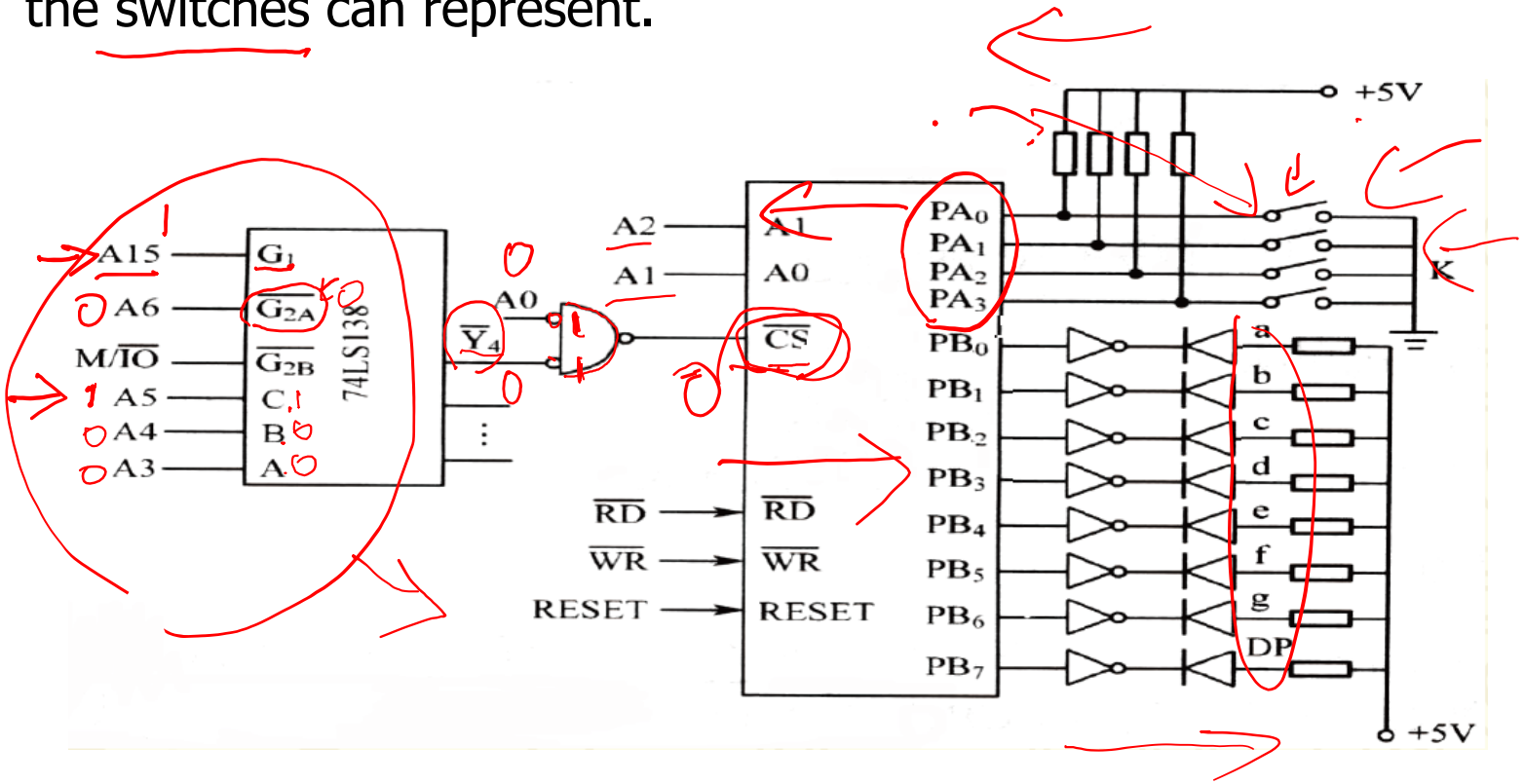
Polling vs. Interruptions

- In Mode 1 and 2, PC stores status of Group A and/or Group B
- By reading from PC using **`IN`** instruction, you can use polling method to check the state of I/O devices

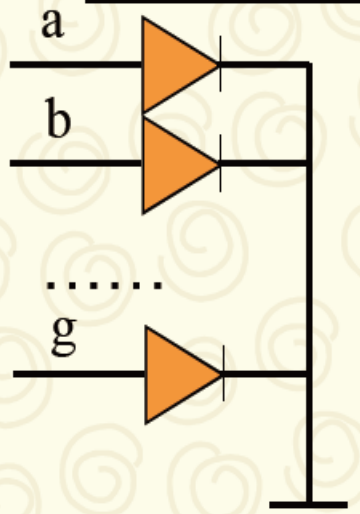


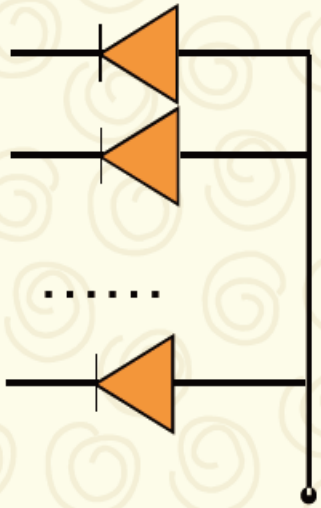
Programming with 8255

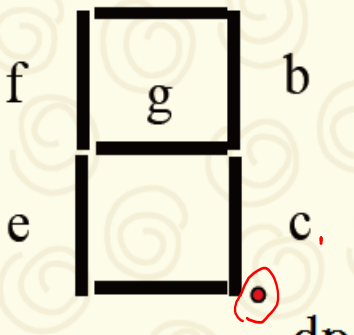
- As shown in the figure, PA and PB of the 8255 are working in mode 0. PA used as input port connects to 4 switches, and PB used as output port connects to a 7-segment LED. Write a program to display a hex digit that the switches can represent.



dp	g	f	e	d	c	b	a
*	*	*	*	*	*	*	*







+5V

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3F	06	5B	4F	66	6D	7D	07	7F	6F	77	7C	39	5E	79	31

Address Decoding

- What are the addresses of ports and the control register?

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

- PA : 8020H
- PB : ?
- PC : ?
- CR : ?

A Solution Program

```
A_PORT   EQU  8020H
B_PORT   EQU  8022H
C_PORT   EQU  8024H
CTRL_PORT  EQU  8026H

DATA  SEGMENT
      TAB1  DB  C0H, F9H, C4H, …, 0DH
DATA  ENDS

CODE  SEGMENT
      ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
       MOV  DS, AX
```

A Solution Program

```
        MOV  AL, 90H        ; Set up the mode of 8255
        MOV  DX, CTRL_PORT
        OUT  DX, AL
ADD1:   MOV DX, A_PORT
        IN   AL, DX         ;read the status of switches
        AND AL, 0FH
        MOV BX, OFFSET TAB1
        XLAT
        MOV DX, B_PORT  ;output to LED
        OUT  DX, AL

        MOV CX, 0600H   ; delay for lighting the LED
ADD2:   LOOP ADD2
        JMP  ADD1
CODE  ENDS
        END START
```

AL ← array[AL]

Experiment 2

0-9 A-F