# CG 2007 Microprocessor Systems
# Lecture 8

## Programmable Peripheral Interface 8255

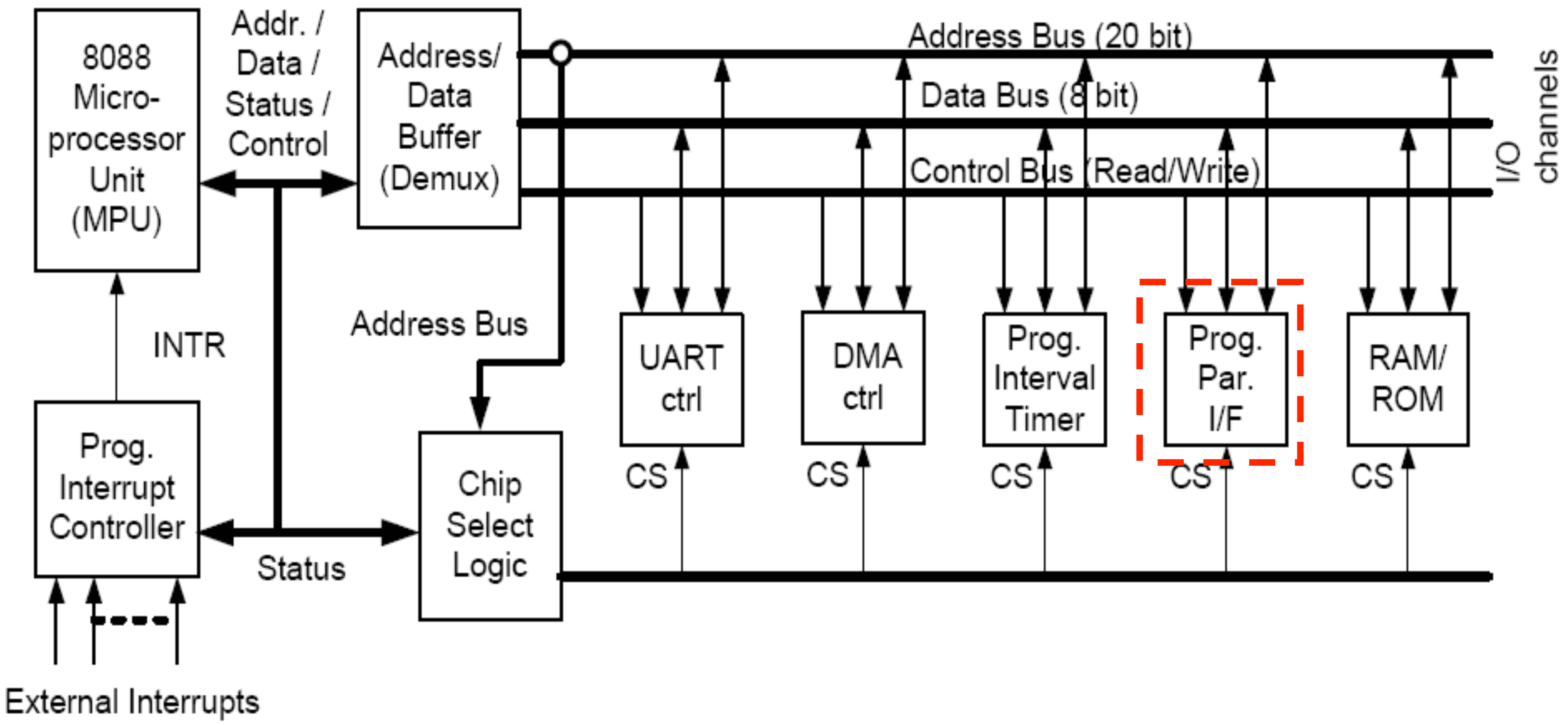
Dr. Ha Yajun
(E1-08-13, *elehy@nus.edu.sg*)

# Lecture 8: Objectives and Outline

- **Objectives**
  - Understand programmable peripheral input-output port chip 8255 and develop 8255 applications

- **Outline**
  - 8255 Chip
  - 8255 I/O Mode 0 Operation
  - 8255 I/O Modes 1 & 2 Operation (not required)

# Simplified System Architecture of an IBM Compatible PC

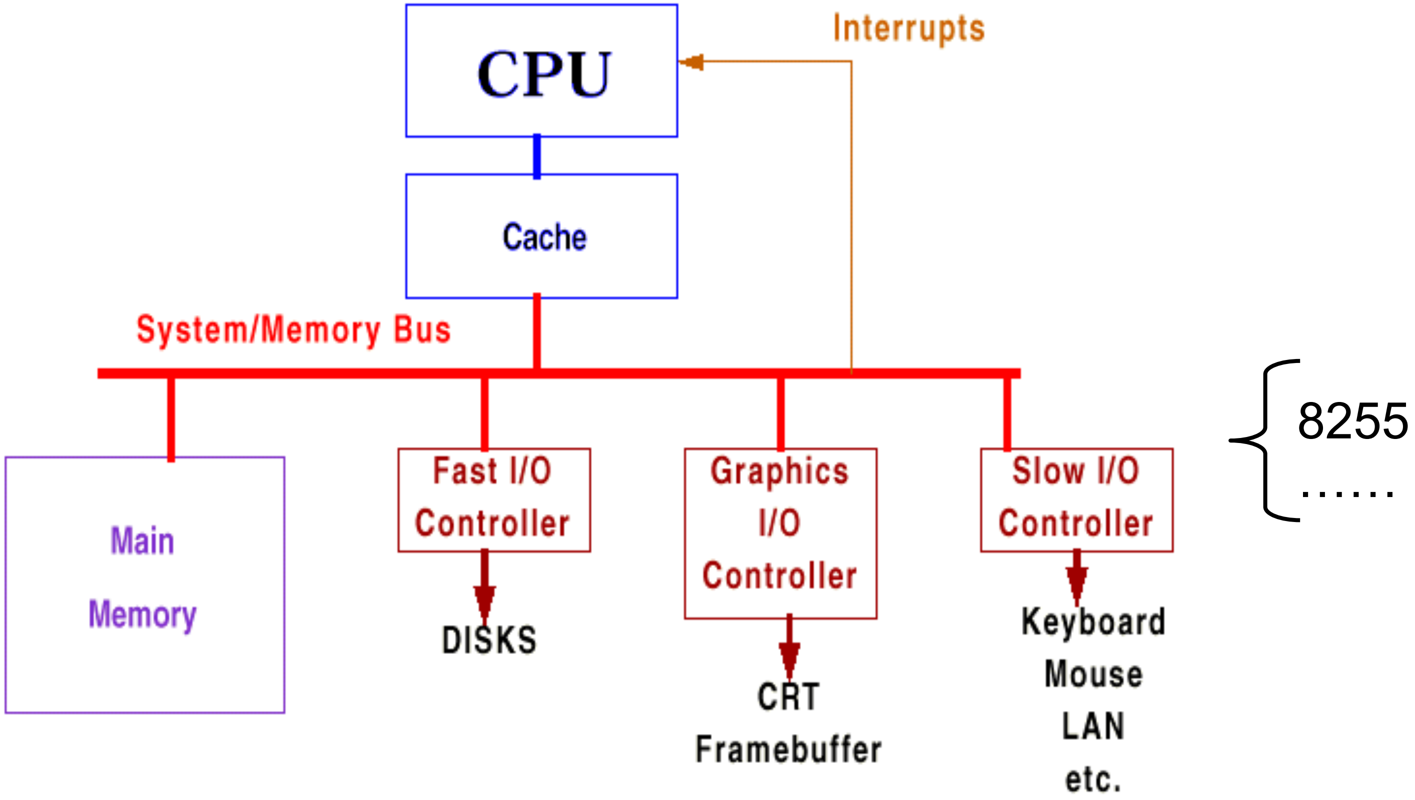


# Outline: Parallel Peripheral Input-Output Port

– 8255 Chip

– 8255 I/O Mode 0 Operation

# I/O Controllers



I/O devices are not directly connected to the system bus. Instead, I/O controllers interface I/O devices to the CPU!

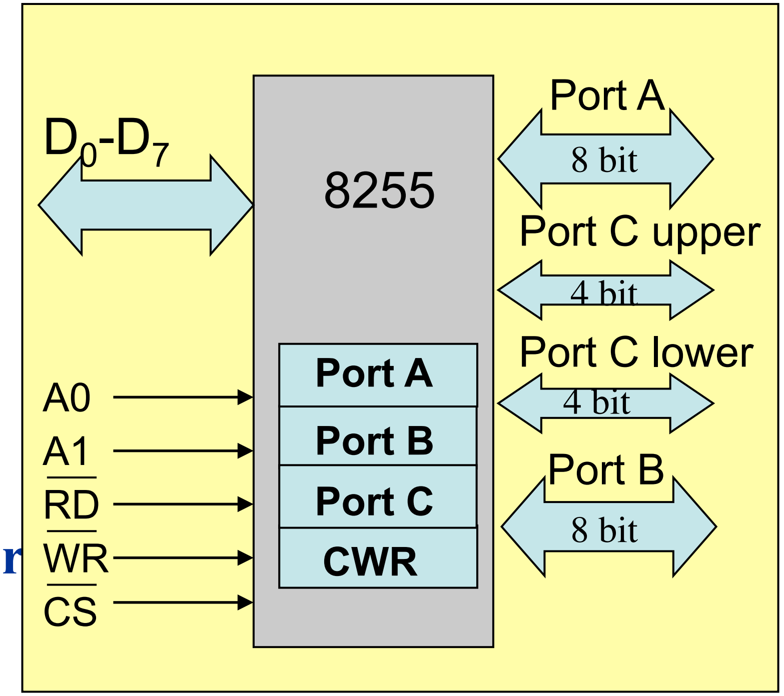
# PPI 8255

- The parallel input-output port chip 8255 is also called as programmable peripheral input-output port.

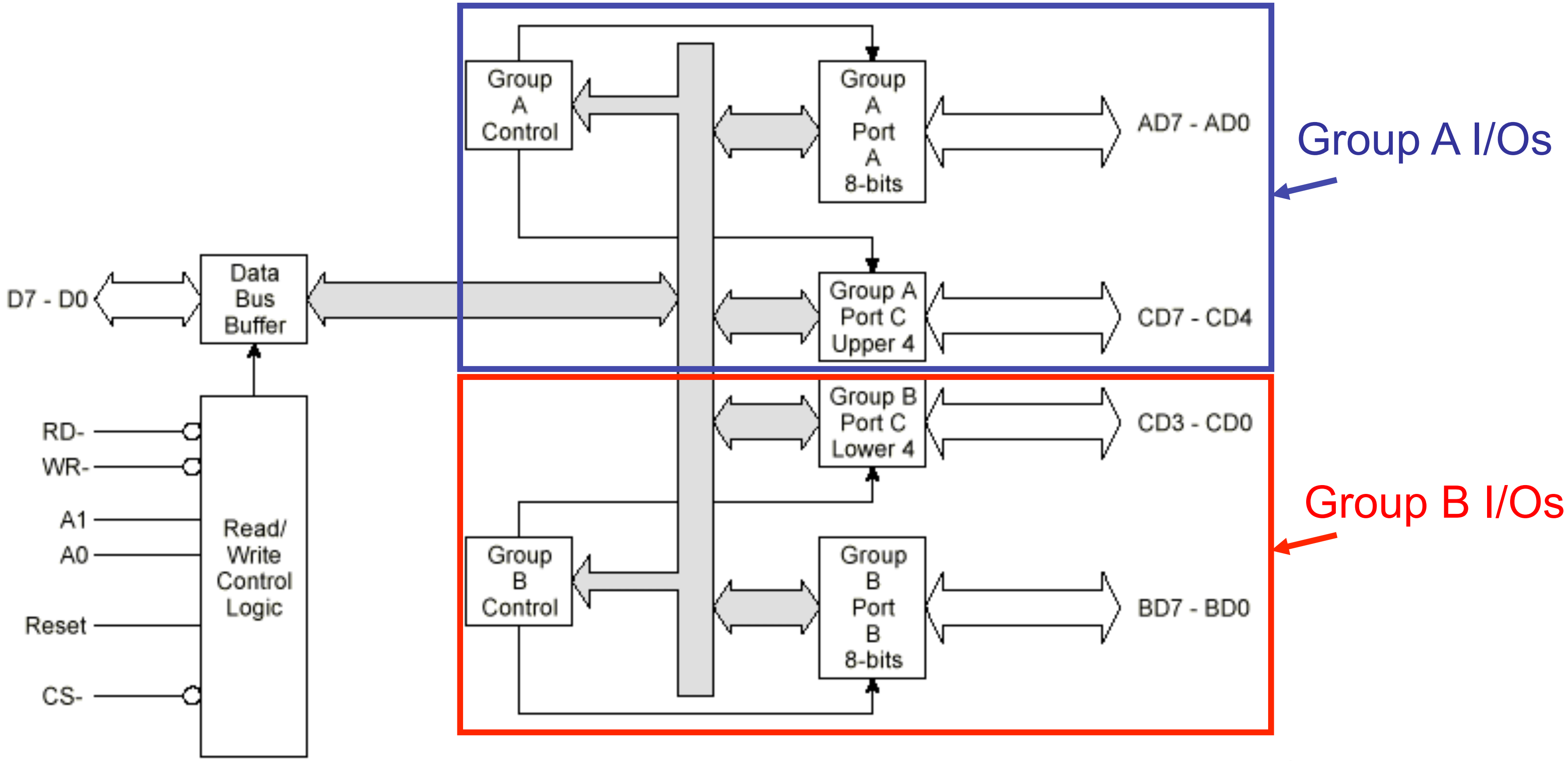
- It is designed for use with Intel's 8-bit, 16-bit and higher capability microprocessors.

- It has 24 input/output lines which can be individually programmed in two groups of twelve lines each or three groups of eight lines each.

- All of these ports can function independently either as input or as output ports by programming the control word register (CWR) of 8255.

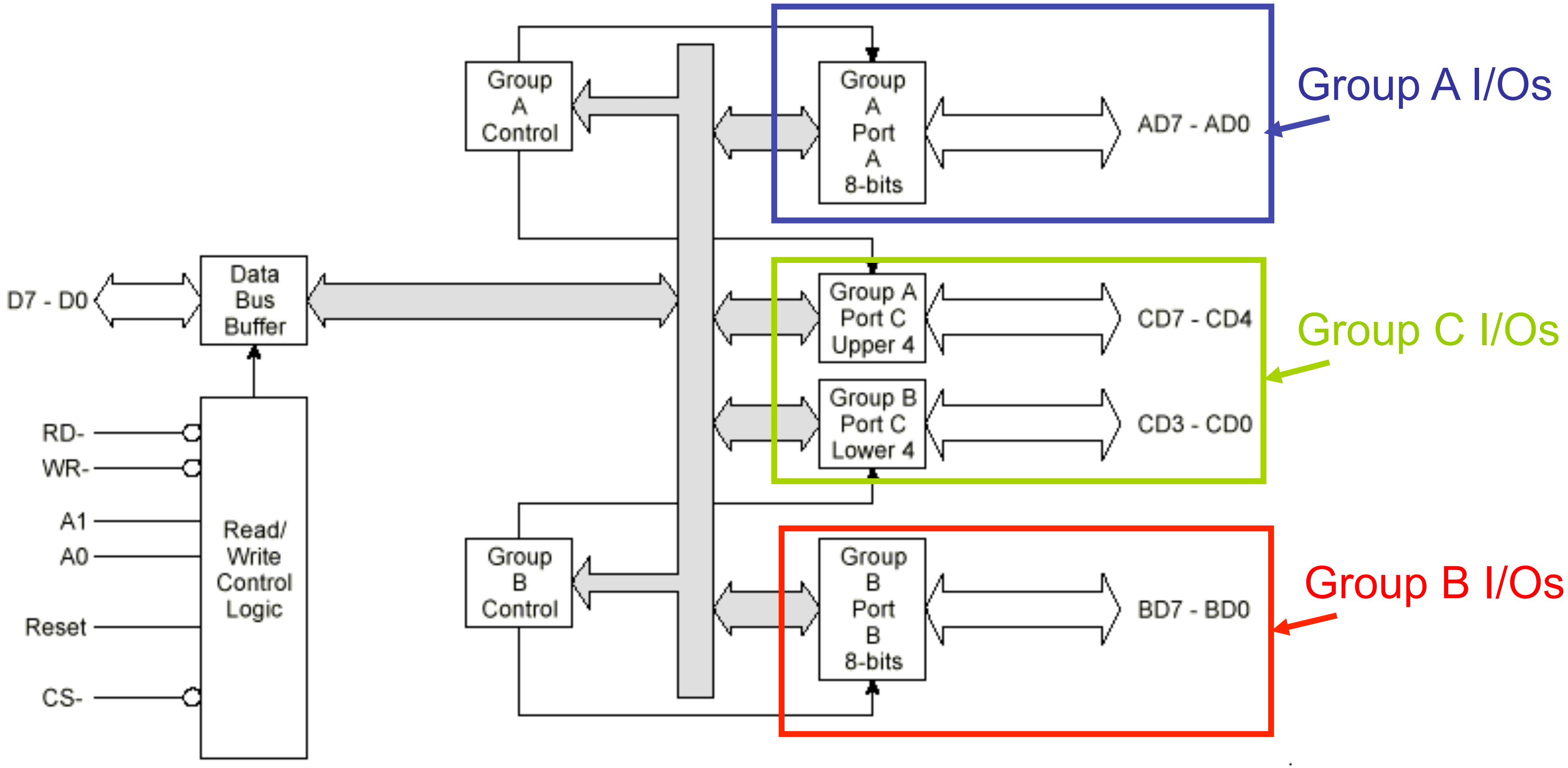


# PPI 8255 Two-Group Configuration



- In the two group configuration, group A contains an 8-bit port A ($PA_0$-$PA_7$) along with a 4-bit port, C upper ($PC_4$-$PC_7$).
- Similarly, group B contains an 8-bit port B ($PB_0$-$PB_7$) along with a 4-bit port, C lower ($PC_0$-$PC_3$).

# PPI 8255 Three-Group Configuration



In the three-group configuration, group A contains an 8-bit port A ($PA_0$-$PA_7$).

Group B contains an 8-bit port B ($PB_0$-$PB_7$).

Group C contains an 8-bit port C ($PC_0$-$PC_7$).

# 8255 Operation

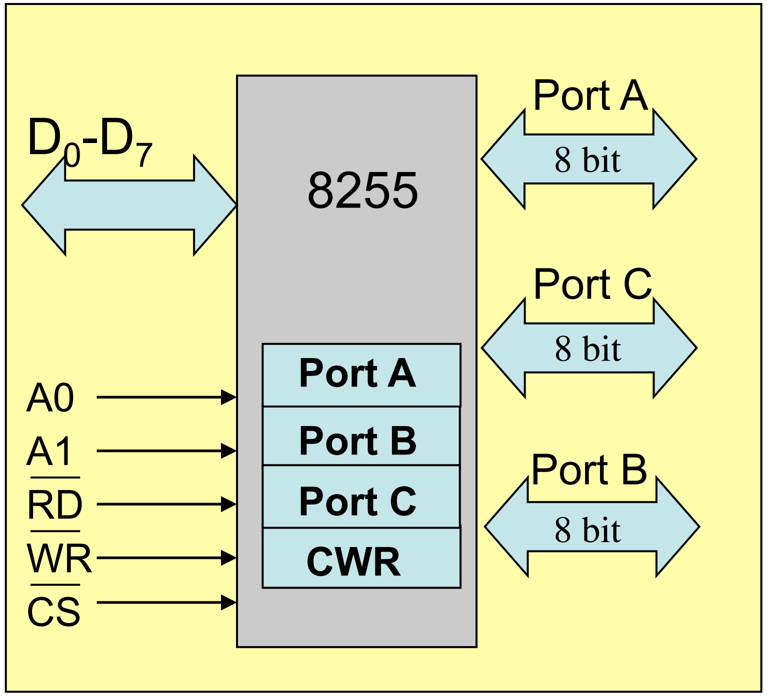
- On the microprocessor side, the 8255a has an 8-bit bidirectional data bus ($D_0$ to $D_7$) which carries data, status information, and commands.

- On the I/O side, it has three 8-bit ports called PORT A, PORT B, and PORT C which can be configured for input or output.

- It contains an internal 8-bit control register which can be modified under software control.

- There are two register select inputs $A_0$ and $A_1$ where: 00 - PORT A, 01 - PORT B, 10 - PORT C, 11 - Control Reg.

- There are two basic modes of operation of 8255: bit set/reset (BSR) mode or I/O mode.



# BSR Mode Control Register ($D_7$=0)

- In the BSR mode, (when $D_7$ of the CWR is set to 0), only PORT C ($PC_0$-$PC_7$) can be used to set or reset its individual bits.

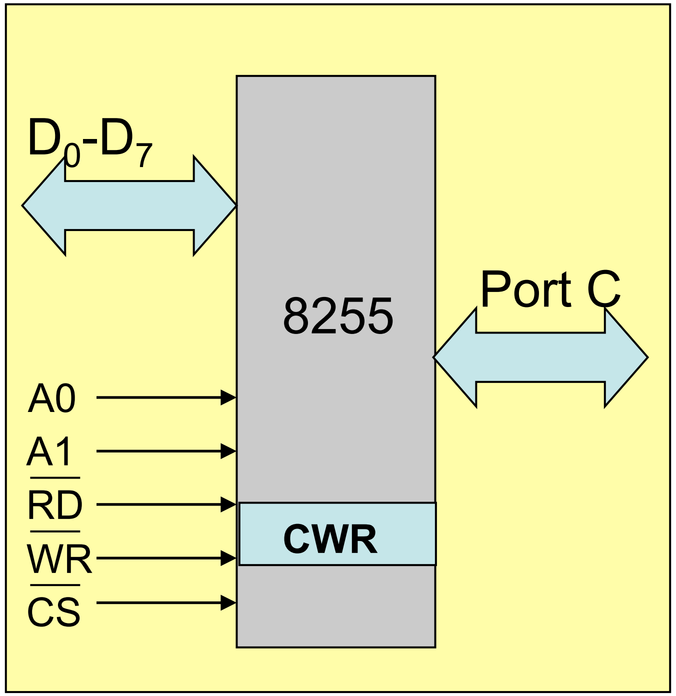
CWR setting to configure PPI in BSR mode:

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 0 | x | x | x | $b_2$ | $b_1$ | $b_0$ | |
| S | | | | | | | |

0 – for BSR mode
1 – I/O mode

| | $b_2$ | $b_1$ | $b_0$ |
|---|---|---|---|
| bit 0: | 0 | 0 | 0 |
| bit 1: | 0 | 0 | 1 |
| bit 2: | 0 | 1 | 0 |
| | . | . | . |
| bit 7: | 1 | 1 | 1 |

$D_0$: 1=set
0=clear



# I/O mode Control Register ($D_7$=1)

- In the I/O mode, (when $D_7$ of CWR is set to 1), the 8255 ports work as programmable I/O ports.

$D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$

$D_7$: **Mode Set Flag**
**0 - bit set/reset**
**1 – I/O modes**

$D_6$ $D_5$: **Group A Mode**
**00 - mode 0**
**01 - mode 1**
**1x - mode 2**

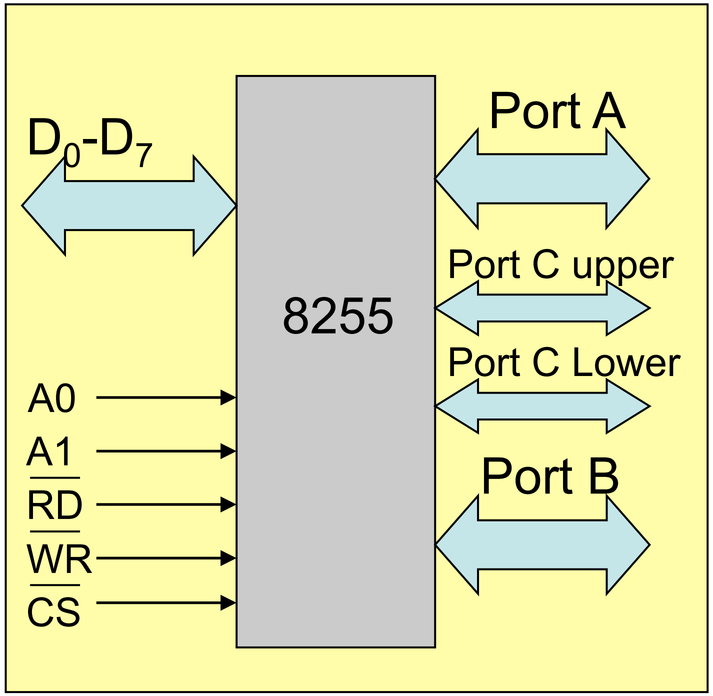
$D_4$: **PORT A**
**0 - output**
**1 - input**

$D_3$: **PORT C (upper)**
**0 - output**
**1 - input**

$D_2$: **Group B Mode**
**0 - mode 0**
**1 - mode 1**

$D_1$: **PORT B**
**0 - output**
**1 - input**

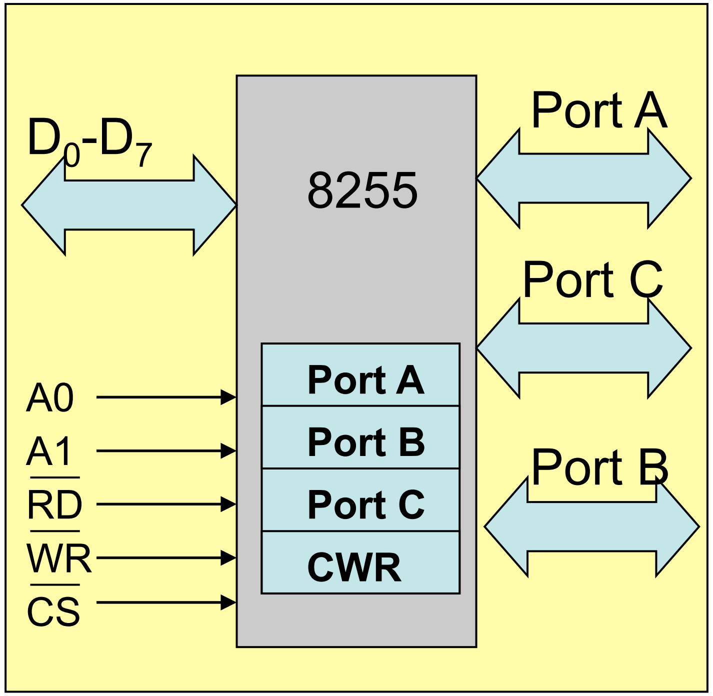
$D_0$: **PORT C (lower)**
**0 - output**
**1 - input**



# I/O Direction

- The I/O direction is determined by $A_0$, $A_1$, $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$

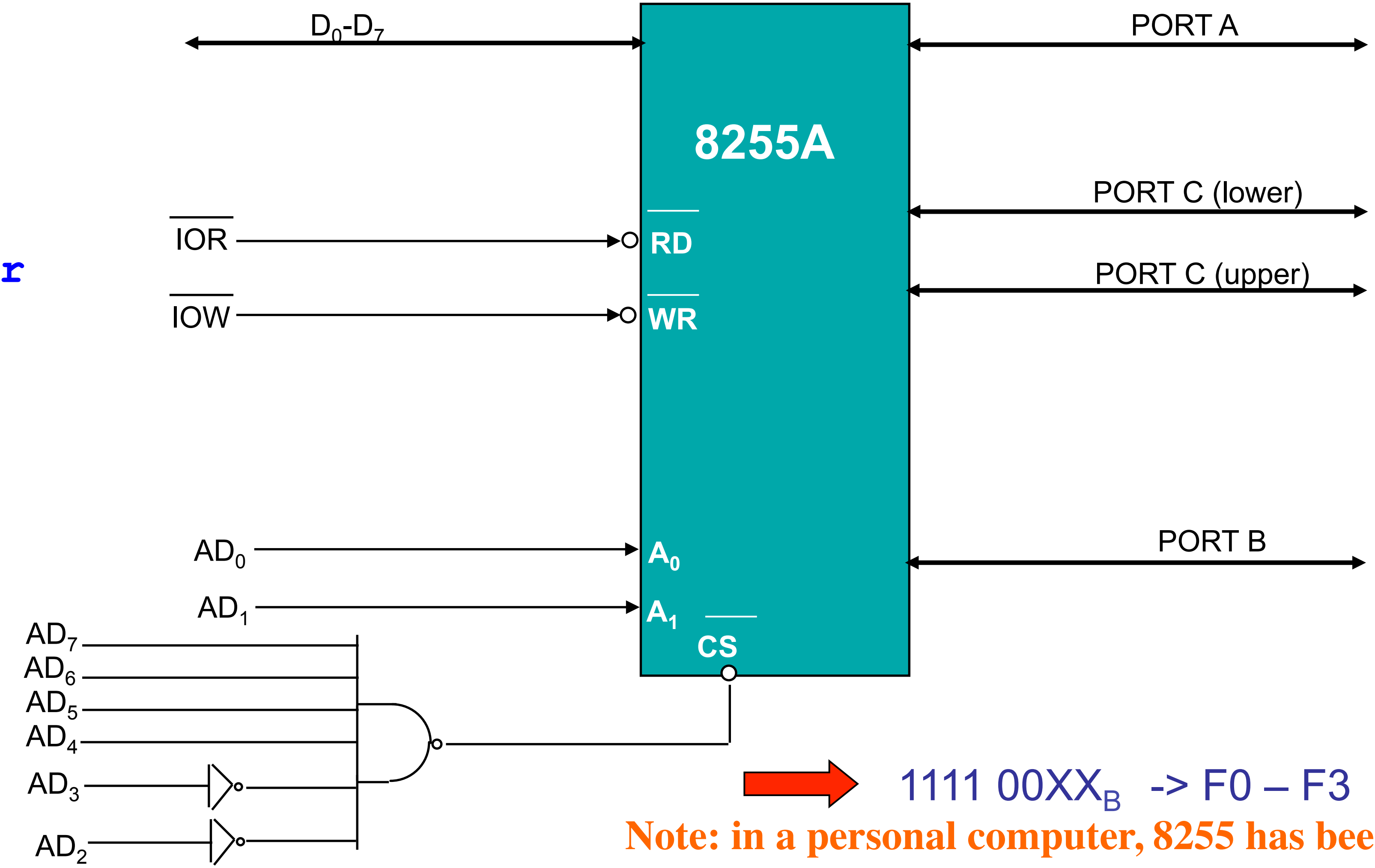
| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | Direction |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | PORT A to data bus |
| 0 | 1 | 0 | 1 | 0 | PORT B to data bus |
| 1 | 0 | 0 | 1 | 0 | PORT C to data bus |
| 1 | 1 | 0 | 1 | 0 | CWR to data bus |
| 0 | 0 | 1 | 0 | 0 | data bus to PORT A |
| 0 | 1 | 1 | 0 | 0 | data bus to PORT B |
| 1 | 0 | 1 | 0 | 0 | data bus to PORT C |
| 1 | 1 | 1 | 0 | 0 | data bus to CWR |
| x | x | x | x | 1 | data bus in tristate |
| x | x | 1 | 1 | 0 | data bus in tristate |



# Example Connection

- Assume the 8255a is connected to an 8086 in the following manner:

| Port | Addr |
|---|---|
| A | F0 |
| B | F1 |
| C | F2 |
| CWR | F3 |



$1111\ 00XX_B$ -> F0 – F3

**Note: in a personal computer, 8255 has been assigned to the port range of 0060H – 0063H!!**

# Outline: Parallel Input-Output Port

- 8255 Chip
- 8255 I/O Mode 0 Operation

# 8255 I/O Mode 0

- Mode 0 is the basic I/O mode - it is useful when the I/O device is assumed to be ready and does not require handshaking signals.

- Data can be simply read from and written to the input and output ports respectively, after appropriate initialization.

- Two 8-bit ports (port A and port B) and two 4-bit ports (port C upper and lower) are available. The two 4-bit ports can be combined and used as a third 8-bit port.

- Any port can be used as an input or output port.

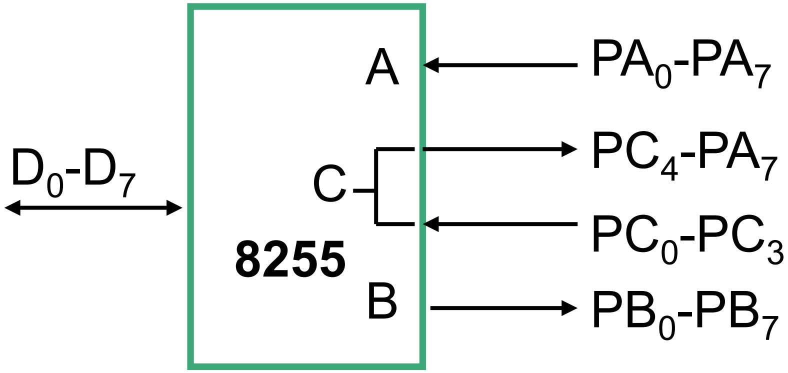
- Output ports are latched. Input ports are not latched.

- A maximum of four ports are available so that overall 16 I/O configurations are possible.

# 8255 I/O Mode 0 Configuration

- **Control Word Register of 8255 should be configured to enable 8255 to work in I/O mode 0.**

- **Configuration: PORT A input**
  **PORT B output**
  **PORT C (upper) output**
  **PORT C (lower) input**


$D_0$-$D_7$

8255

A

C

B

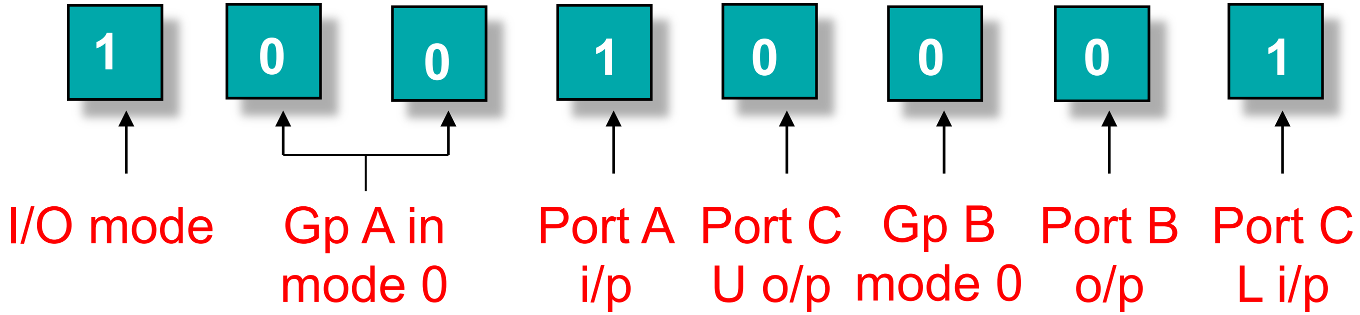
$PA_0$-$PA_7$

$PC_4$-$PA_7$

$PC_0$-$PC_3$

$PB_0$-$PB_7$

Mode Control Word:


1 0 0 1 0 0 0 1

I/O mode

Gp A in mode 0

Port A i/p

Port C U o/p

Gp B mode 0

Port B o/p

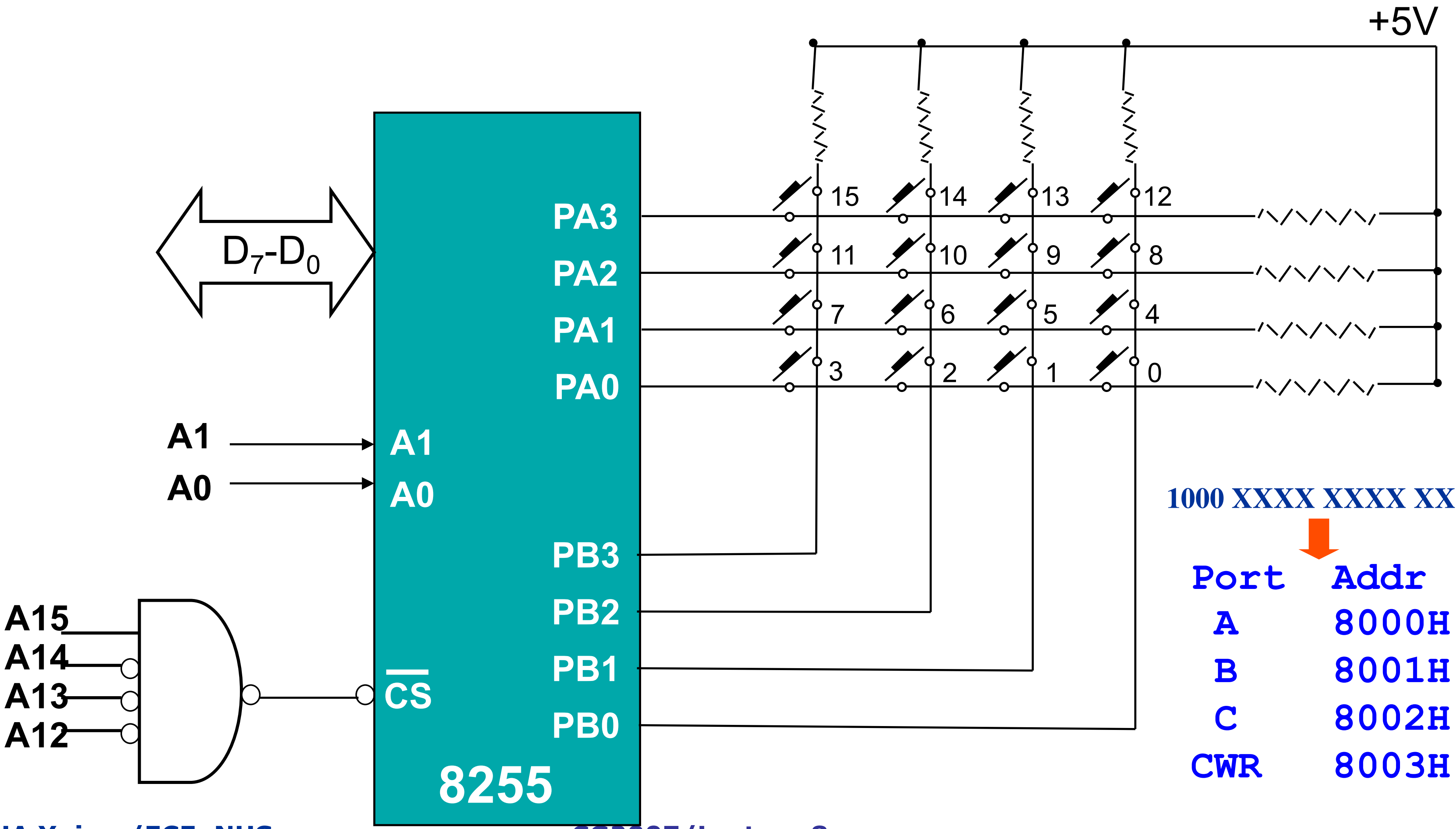
Port C L i/p

Initialize Code (assuming prior implementation in slide 13):

```
MOV AL, 91H
 OUT 0F3H, AL
```

# Example

- PROBLEM: **Use the 8255 to interface a 4*4 keyboard to an 8086 microprocessor**
  - This is both a hardware and a software design problem



**1000 XXXX XXXX XXXX**

| Port | Addr |
|---|---|
| A | 8000H |
| B | 8001H |
| C | 8002H |
| CWR | 8003H |

# Solution

- PORT A ($PA_0$-$PA_3$) will be used as output port for selecting a row of keys.

- PORT B ($PB_0$-$PB_3$) will be used an input port for sensing the closed key.

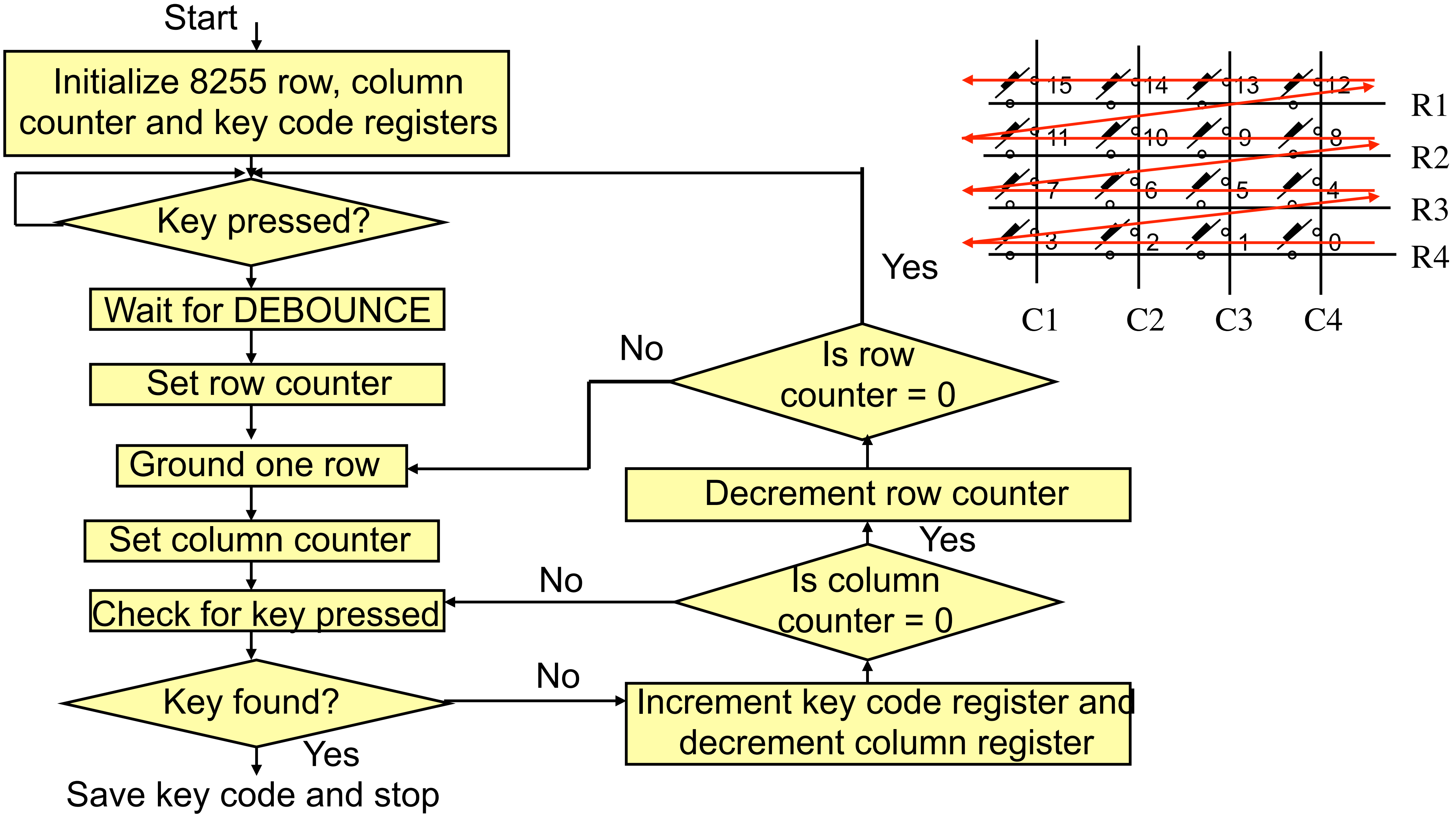
- The keyboard rows are selected one by one through port A and the port B columns are polled continuously till a key closure is sensed.

- When a button is open the input bit of port B will be high, however, when a button is closed and the output bit from port A is low, the input bit of port B will be low.

- Whenever a mechanical switch is closed or opened the contacts will "bounce" for several milliseconds, so there is a requirement to "debounce" the switch.

- According to the interfacing shown in previous slide, the addresses of port A and port B are 8000H and 8001H respectively while the address of CWR is 8003H.

# Flow Chart for Keyboard Interfacing



Start

Initialize 8255 row, column counter and key code registers

Key pressed?

Wait for DEBOUNCE

Set row counter

Ground one row

Set column counter

Check for key pressed

Key found?

Yes

Save key code and stop

No

Increment key code register and decrement column register

Is column counter = 0

No

Yes

Decrement row counter

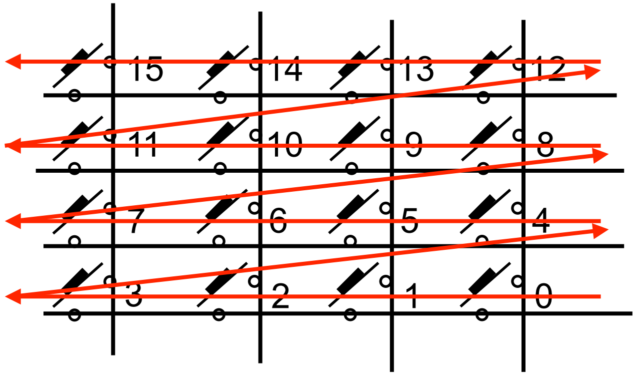
Is row counter = 0

No

Yes

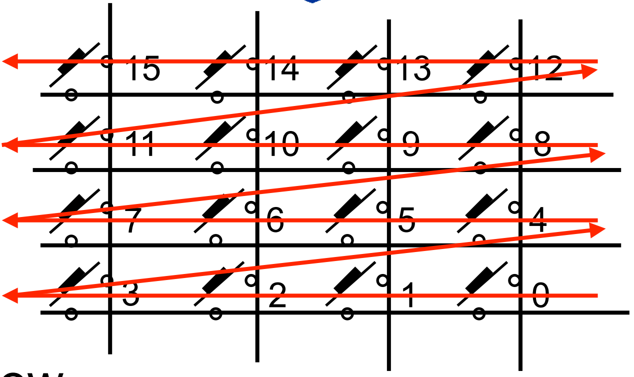
# Example Code I

```
CODE         SEGMENT
ASSUME       CS: CODE
START:       MOV AL, 82H      ;mode 0, A - out, B-in
             MOV DX, 8003H
             OUT DX, AL       ;send the control word

             MOV BL, 00H      ;initialize BL for key code
             XOR AX, AX       ;clear al flags
             MOV DX, 8000H    ;port A address to DX
             OUT DX, AL       ;ground all rows

             MOV DX, 8001H    ;Port B address to DX

WAIT:        IN AL, DX        ;read all columns
             AND AL, 0FH      ;Mask data lines D7-D4
```



DX: port address
BL: key code

# Example Code II



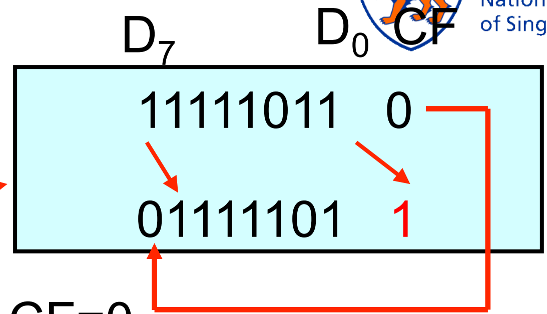
```
            CMP AL, 0FH       ;any key pressed?
            JZ WAIT           ;if not, wait till key press
            CALL DEBOUNCE ;wait for 10ms if key press
            MOV AL, 07FH      ;load data byte to ground a row
            MOV BH, 04H       ;set row counter

NXTROW:     ROL AL, 01H       ;rotate AL to ground next row
            MOV CH, AL        ;save data byte to ground next row
            MOV DX, 8000H     ;port A address to DX
            OUT DX, AL        ;ground one of the rows

            MOV DX, 8001H     ;port B address to DX
            IN AL, DX         ;read input port for key closure
            AND AL, 0FH       ;Mask D4-D7
            MOV CL, 04H       ;set column counter
```

ROL AL, 01

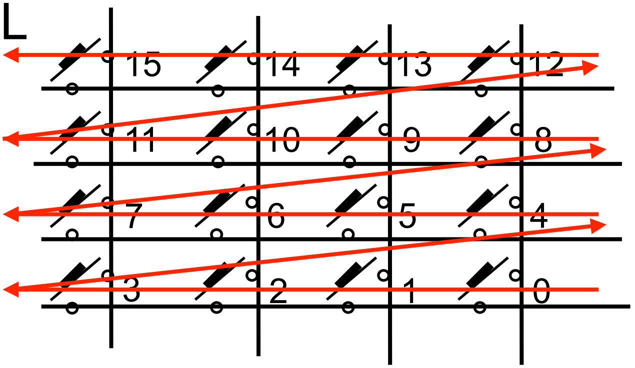
01111111 ⇨ 11111110

DX: port address
BH: row counter
BL: key code
CL: column counter

# Example Code III



```
NXTCOL:     RCR AL, 01H      ;move D0 to CF
            JNC CODEKY       ;key closure is found, if CF=0
            INC BL           ;inc BL for next binary key code
            DEC CL           ;dec column counter if no key closure
            JNZ NXTCOL       ;check for key closure in next column
            MOV AL, CH       ;Load data byte to ground next row
            DEC BH           ;if no key closure found in all columns
                              in this row, go to ground next row
            JNZ NXTROW       ;go back to ground next row
            JMP WAIT         ;back to check for key closure again
CODEKY      MOV AL, BL       ;key code is transferred to AL
            MOV AH, 4CH      ;return to DOS prompt
            INT 21H          ;
```



CH: data byte to ground a row
CL: column counter
BL: key code

# Example Code IV

```
;This procedure generates 10ms delay at 5MHz
;operating frequency, which corresponds to
;50,000 clock cycles.
 DEBOUNCE:     PROC   NEAR
               PUSH   CX
 Option1:      MOV CX, 094Ch ; 2380 dec
 BACK:         NOP       ; 3 clocks
               LOOP BACK; 18 clocks

 ;Option2:     MOV CX, 08E0h ; 2272 dec
 ;BACK2:       NOP        ; 3 clocks
 ;             DEC CX     ; 3 clocks
 ;             JNZ BACK2 ; 16 clocks
               POP CX
               RET
 DEBOUNCE      ENDP

 CODE          ENDS
 END           START
```