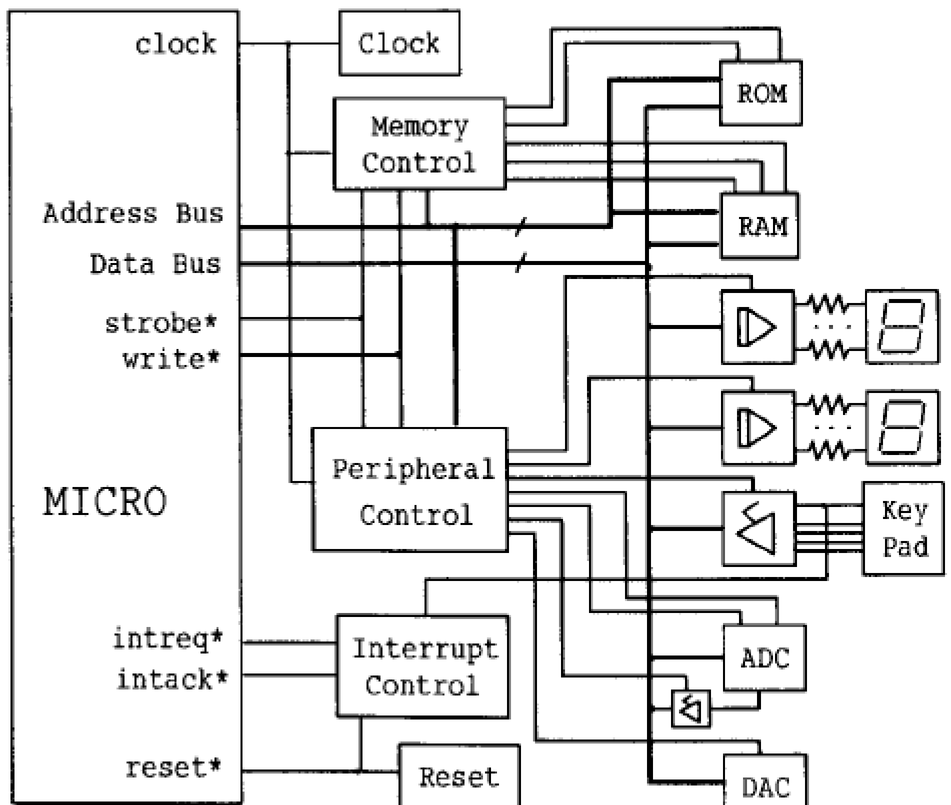
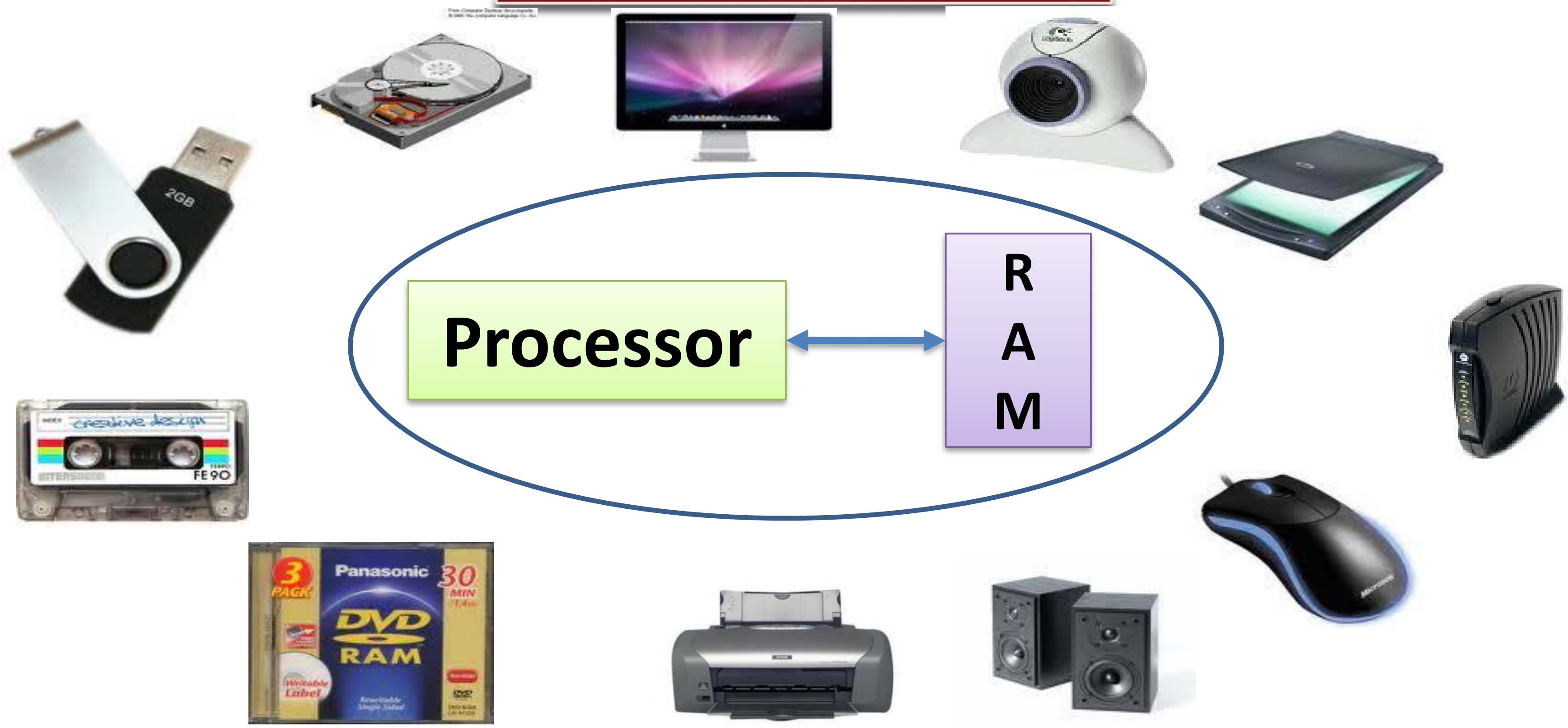
Peripheral and its characteristics

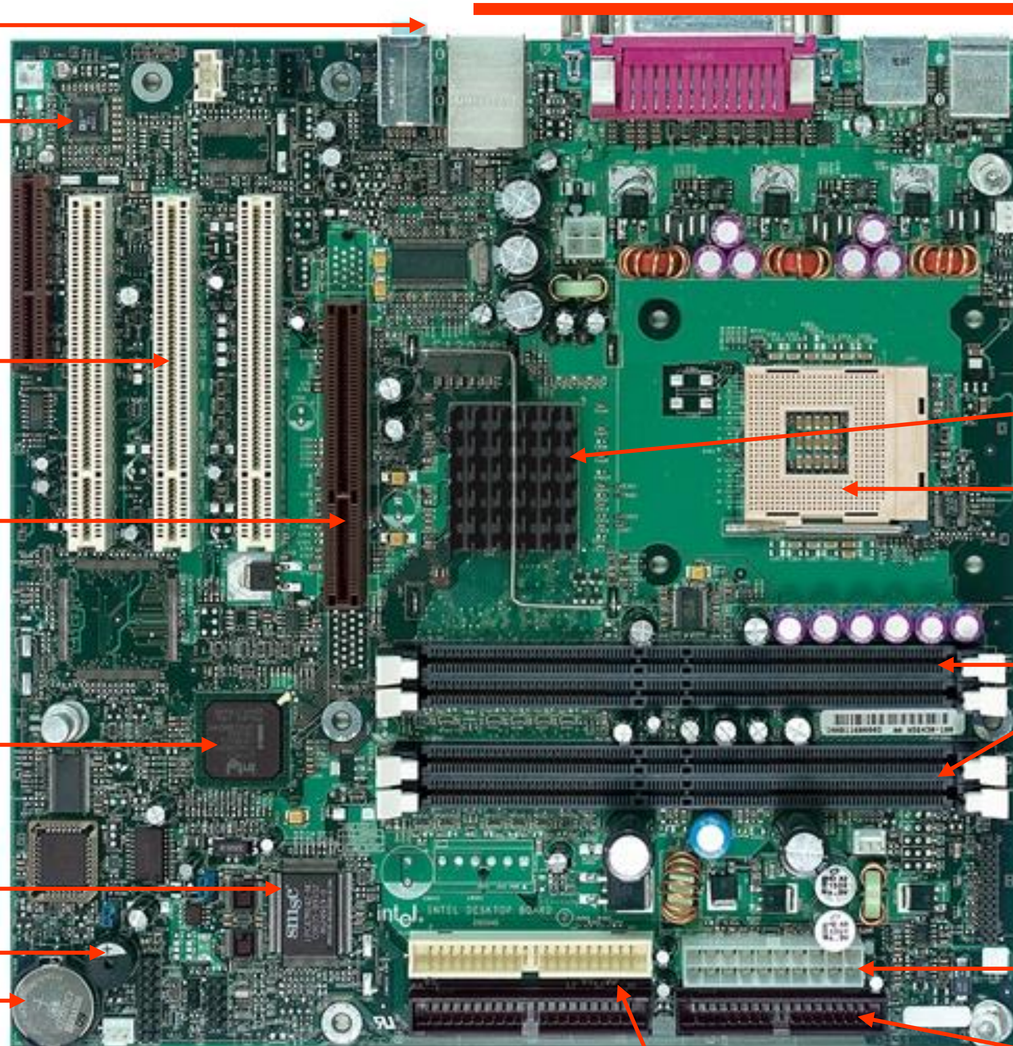


Introduction



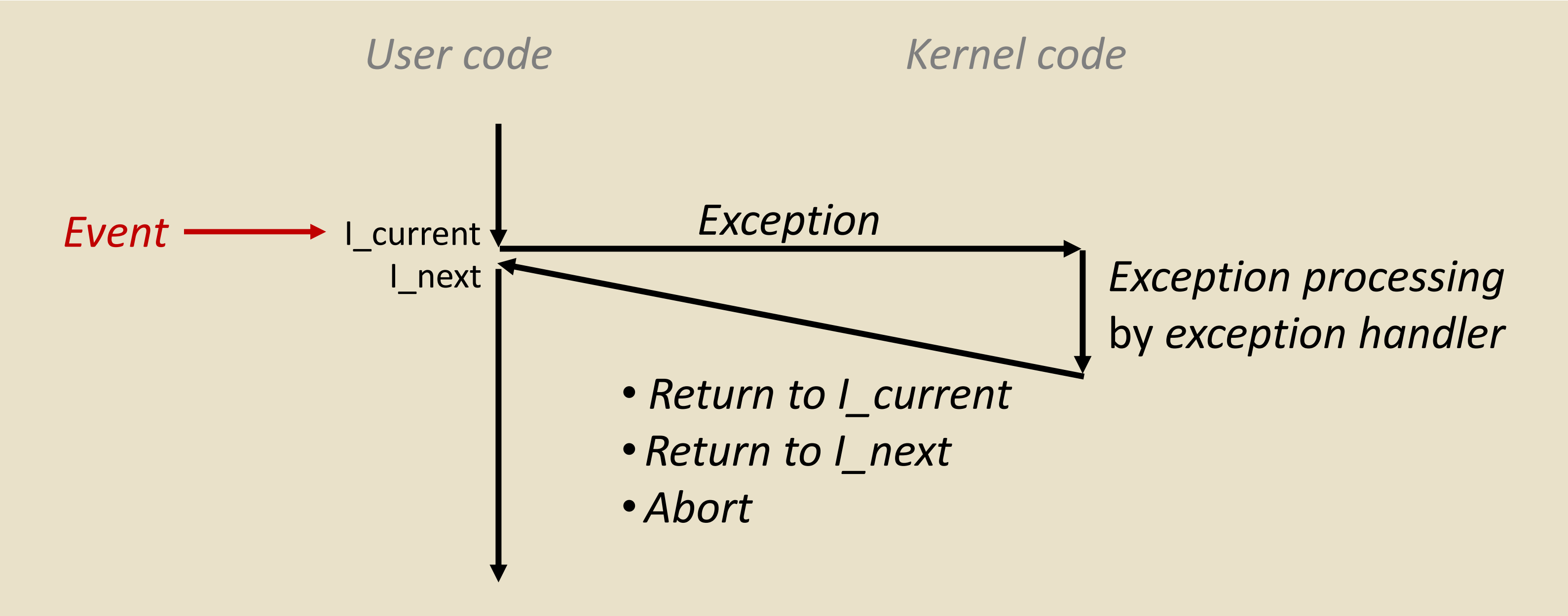
- Computer Systems
  - Internal (processor + memory (RAM))
  - Peripheral (Disk, Display, Audio, Eth,..)

Intel D850MD Motherboard



Exceptions

- An *exception* is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
  - Kernel is the memory-resident part of the OS
  - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



Control Flow

- Processors do only one thing:
  - From startup to shutdown, a CPU simply reads and executes (interprets) a sequence of instructions, one at a time
  - This sequence is the CPU's *control flow* (or *flow of control*)

Physical control flow

Time

<startup>
inst$_1$
inst$_2$
inst$_3$
…
inst$_n$
<shutdown>

Altering the Control Flow

- Up to now: two mechanisms for changing control flow:
  - Jumps and branches
  - Call and return

  React to changes in ***program state***

- Insufficient for a useful system:
  Difficult to react to changes in *system state*
  - Data arrives from a disk or a network adapter
  - Instruction divides by zero
  - User hits Ctrl-C at the keyboard
  - System timer expires

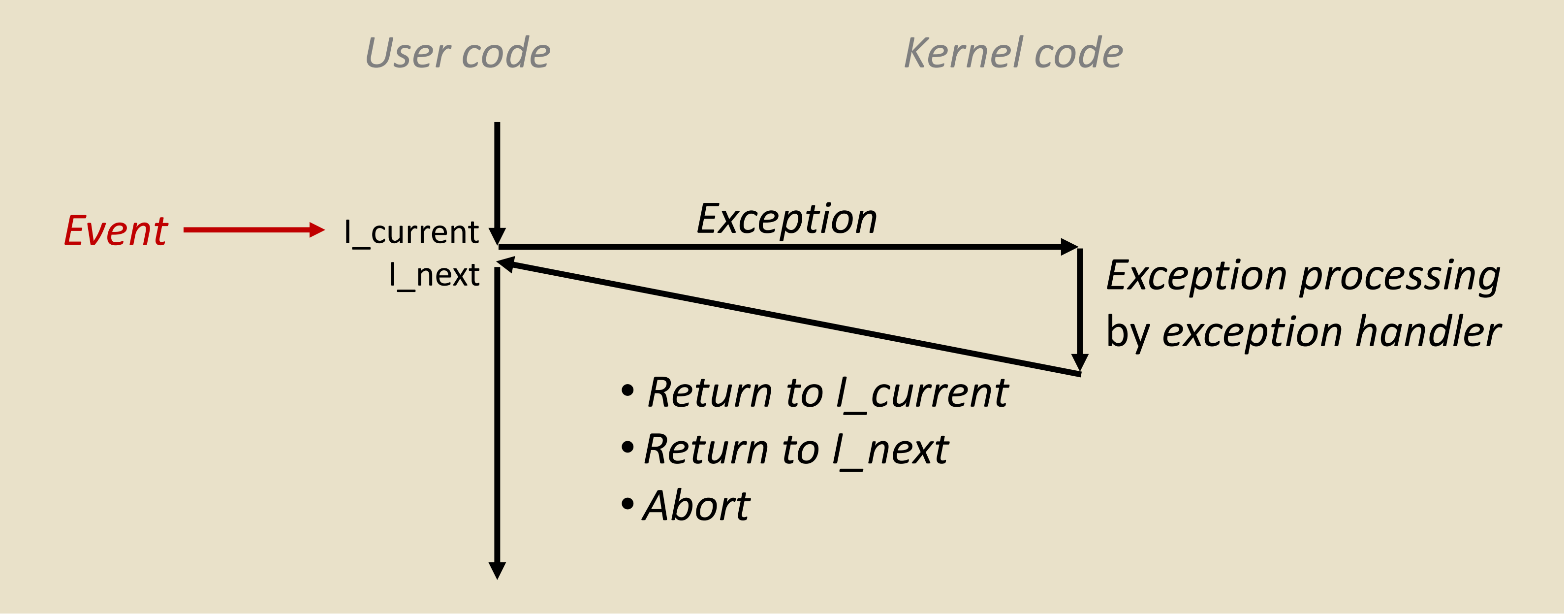
- System needs mechanisms for "exceptional control flow"

Exceptional Control Flow

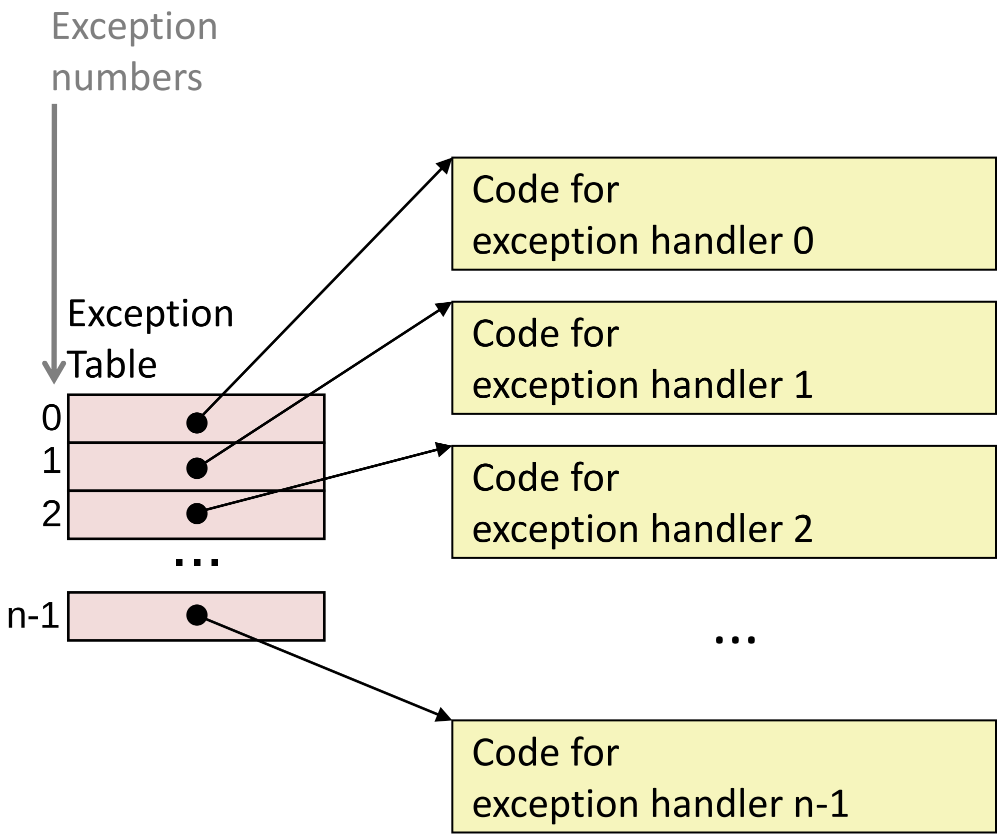
- Exists at all levels of a computer system
- Low level mechanisms
  - 1. **Exceptions**
    - Change in control flow in response to a system event (i.e., change in system state)
    - Implemented using combination of hardware and OS software
- Higher level mechanisms
  - 2. **Process context switch**
    - Implemented by OS software and hardware timer
  - 3. **Signals**
    - Implemented by OS software
  - 4. **Nonlocal jumps**: `setjmp()` and `longjmp()`
    - Implemented by C runtime library

Exceptions

- An *exception* is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
  - Kernel is the memory-resident part of the OS
  - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C

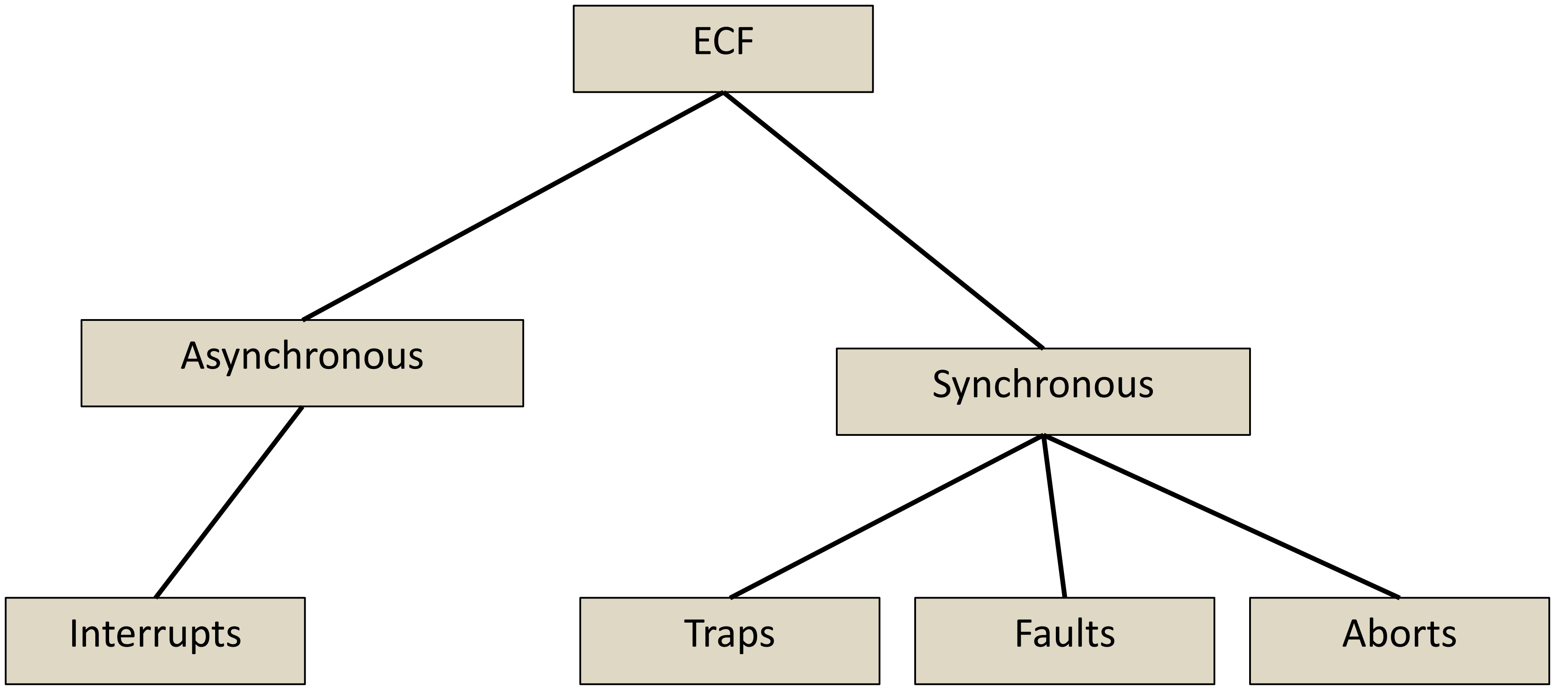


Exception Tables



- Each type of event has a unique exception number k
- k = index into exception table (a.k.a. interrupt vector)
- Handler k is called each time exception k occurs

Taxonomy


ECF
Asynchronous
Synchronous
Interrupts
Traps
Faults
Aborts

Asynchronous Exceptions (Interrupts)

- Caused by events external to the processor
  - Indicated by setting the processor's *interrupt pin*
  - Handler returns to "next" instruction

- Examples:
  - Timer interrupt
    - Every few ms, an external timer chip triggers an interrupt
    - Used by the kernel to take back control from user programs
  - I/O interrupt from external device
    - Hitting Ctrl-C at the keyboard
    - Arrival of a packet from a network
    - Arrival of data from a disk

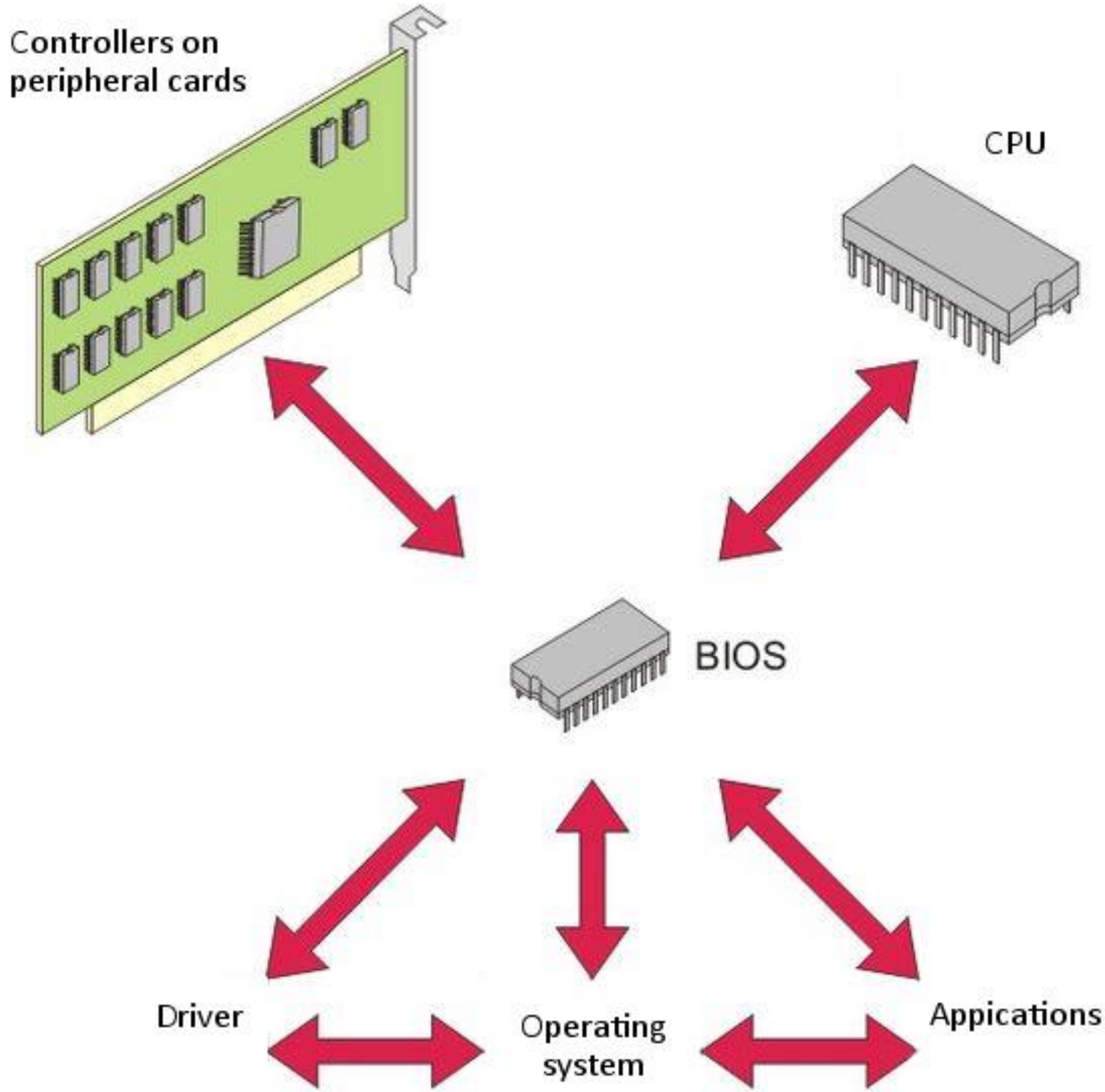
Synchronous Exceptions

- Caused by events that occur as a result of executing an instruction:
  - ***Traps***
    - Intentional
    - Examples: ***system calls***, breakpoint traps, special instructions
    - Returns control to "next" instruction
  - ***Faults***
    - Unintentional but possibly recoverable
    - Examples: page faults (recoverable), protection faults (unrecoverable), floating point exceptions
    - Either re-executes faulting ("current") instruction or aborts
  - ***Aborts***
    - Unintentional and unrecoverable
    - Examples: illegal instruction, parity error, machine check
    - Aborts current program

System Calls

- **Each x86 system call has a unique ID number**
- **Examples:**

Number	*Name*	*Description*
0	`read`	Read file
1	`write`	Write file
2	`open`	Open file
3	`close`	Close file
4	`stat`	Get info about file
57	`fork`	Create process
59	`execve`	Execute a program
60	`_exit`	Terminate process
62	`kill`	Send signal to process

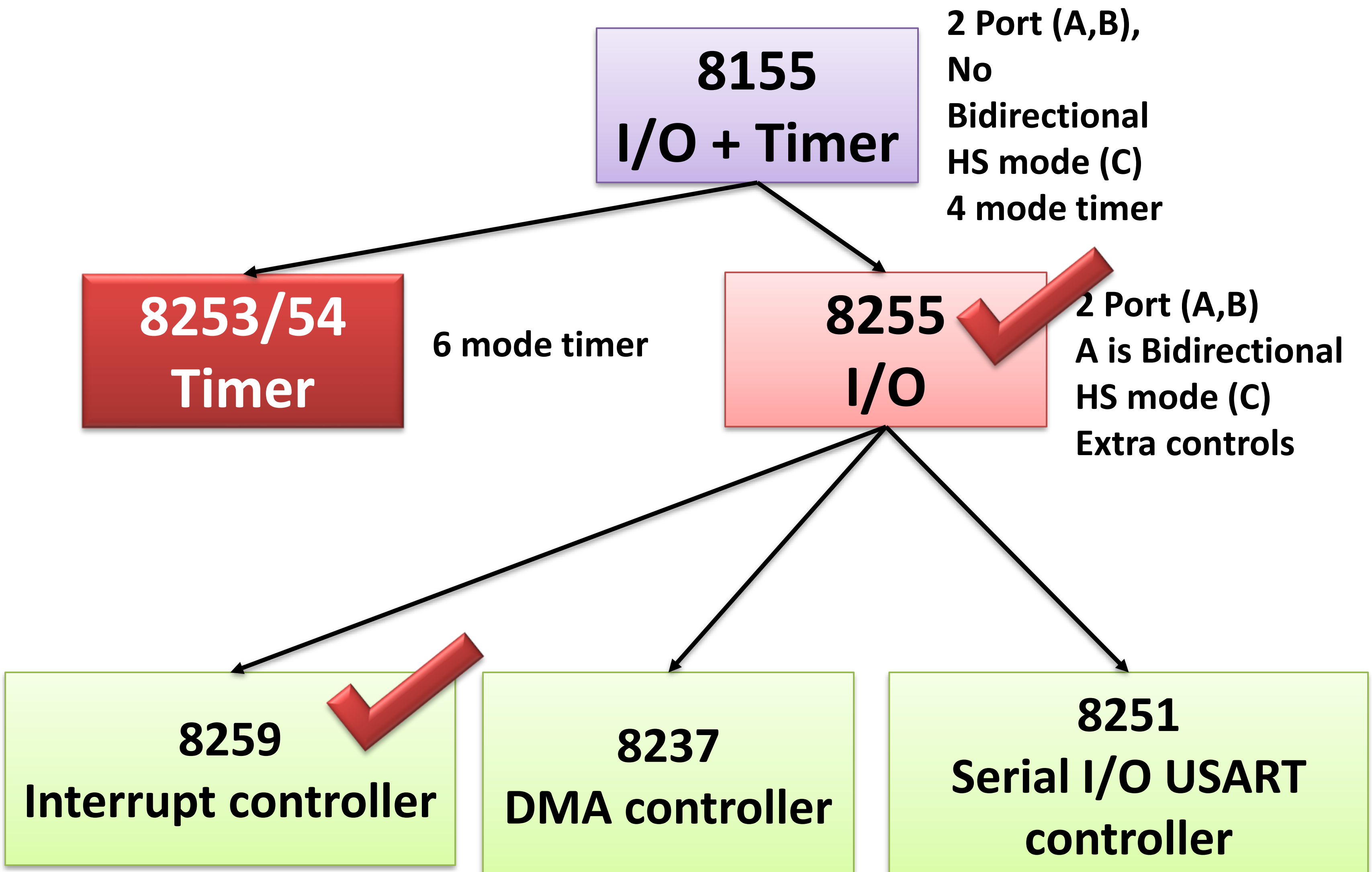


Controllers on
peripheral cards
CPU
BIOS
Driver
Operating
system
Appications

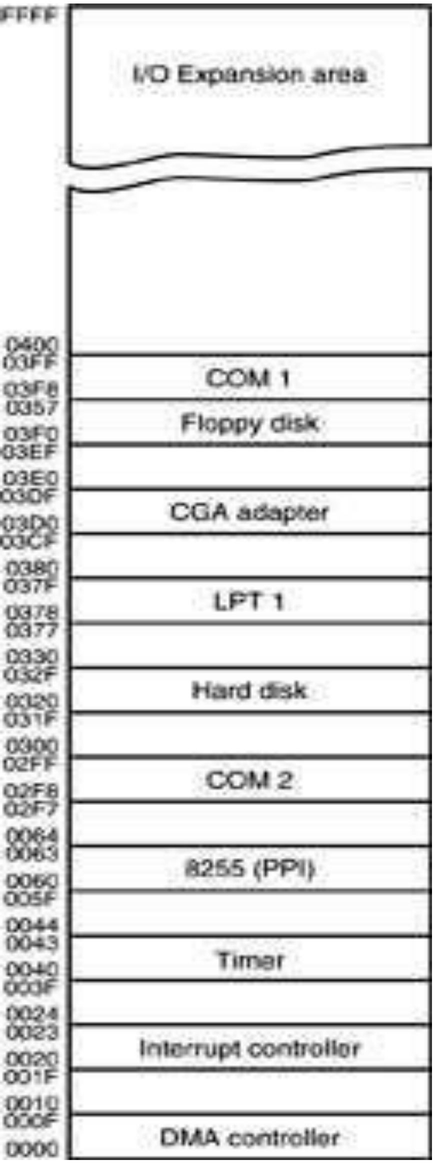
Outline

- Introduction to peripheral
- Type of peripheral (I/O)
- Characteristics of peripheral (I/O)
- Method of getting/sending data from/to I/O
- Programmable Peripheral Interface
- Peripheral controller (8255A)

Hierarchy of I/O Control Devices



Personal Computer I/O Map

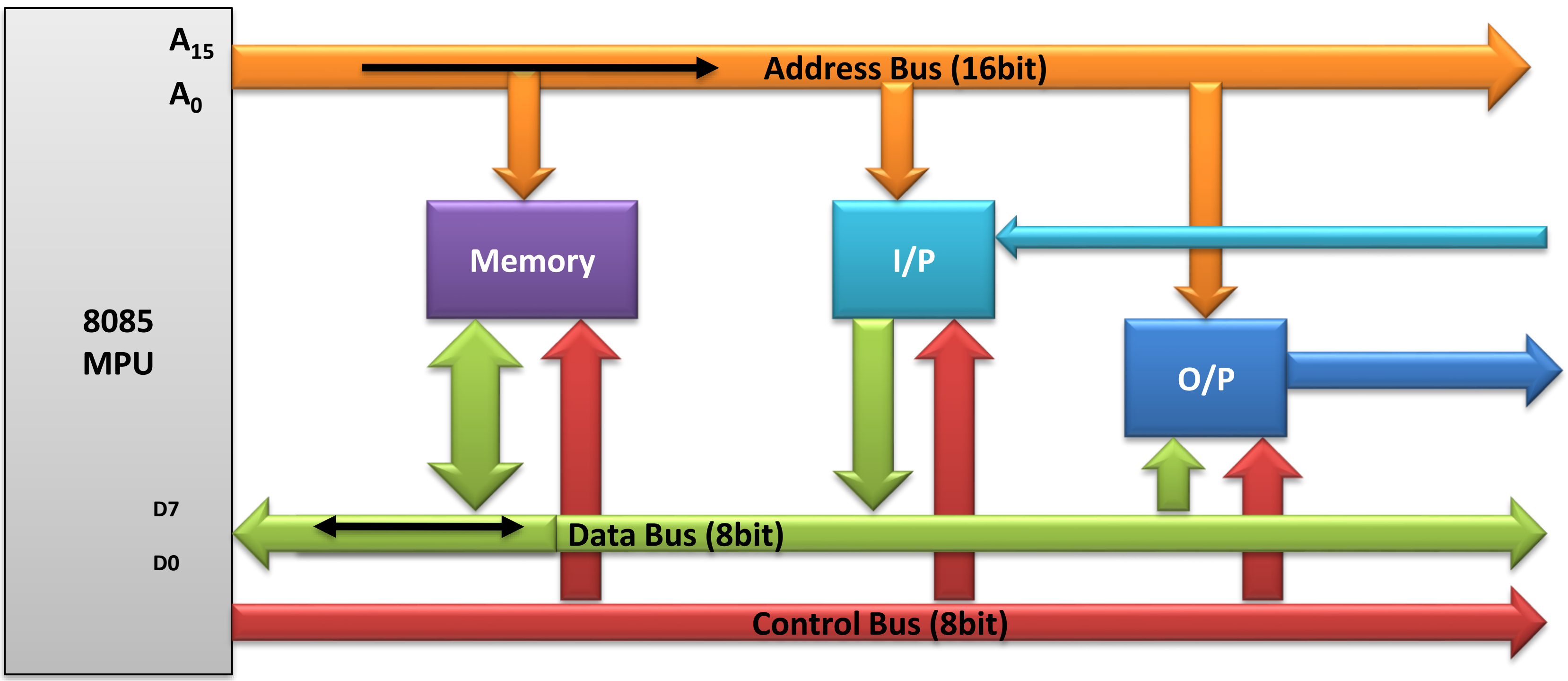


- the PC uses part of I/O map for dedicated functions, as shown here
- I/O space between ports 0000H and 03FFH is normally reserved for the system and ISA bus
- ports at 0400H–FFFFH are generally available for user applications, main-board functions, and the PCI bus
- 80287 coprocessor uses 00F8H–00FFH, so Intel reserves I/O ports 00F0H–00FFH

I/O map of a personal computer illustrating many of the fixed I/O areas.

Outside MPU

- RAM Memory is integral part of MP System
  - MPU fetch instruction from RAM
  - MPU RD and WR data to RAM (same speed as MPU)
- How Ram is interfaced



Primary function of MPU

- Read Instruction from memory
- Execute instruction
- Read/Write data to memory
- Some time send result to output device
  - LEDs, Monitor, Printer
- Interfacing a peripheral
  - Why: To enable MPU to communicate with I/O
  - Designing logic circuit H/W for a I/O
  - Writing instruction (S/W)

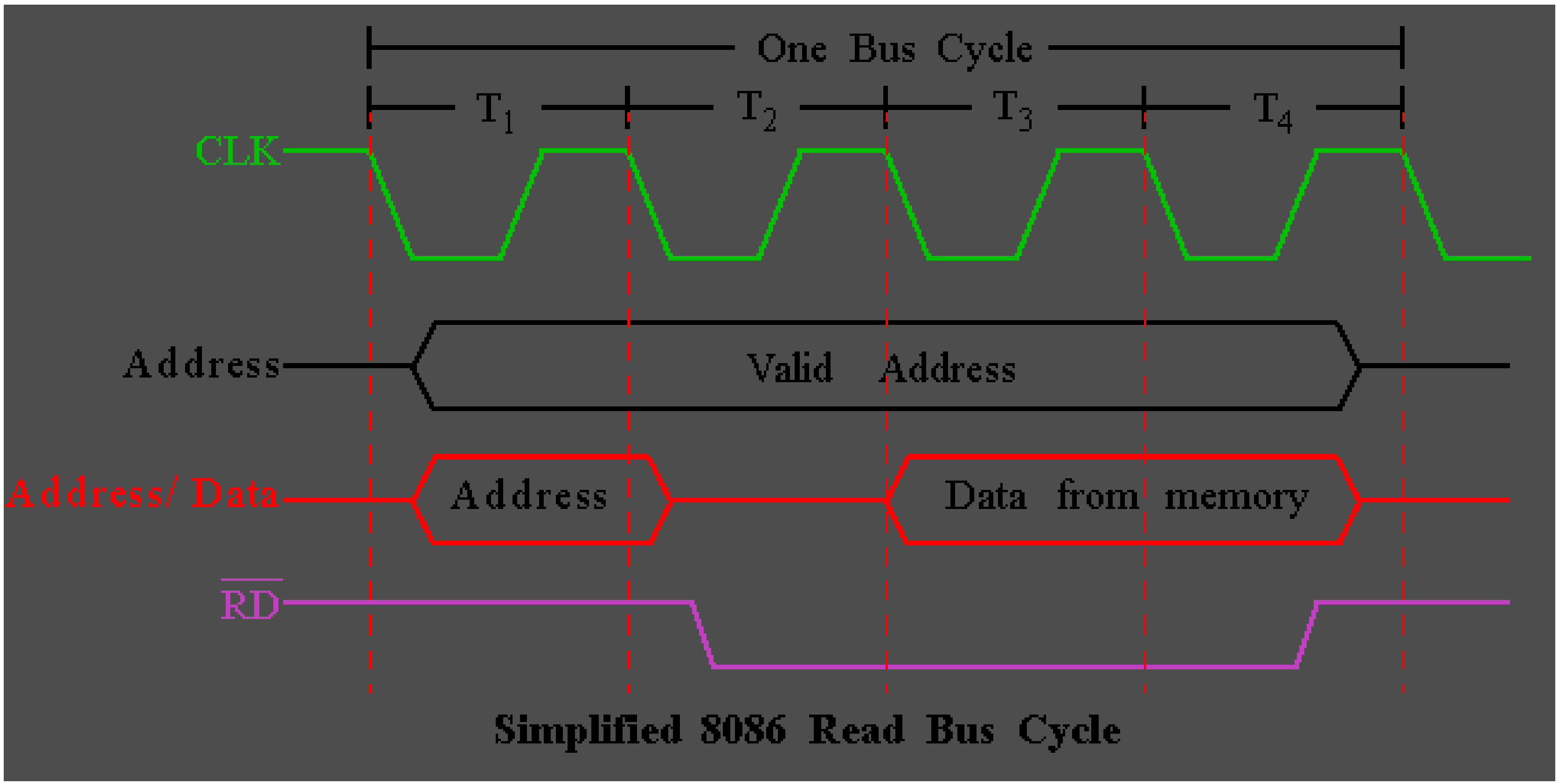
Type of I/O

- Peripheral I/O
  - IN port (Instruction), OUT port (instruction)
  - Identified with 8 bit address (Immediate)
  - Example: IN 01H ; Receive data from port 1
- Memory mapped I/O
  - A peripheral is connected as if it were a memory location
  - Identified with 16 bit address
  - Data transfer by : LDA, STA, MOV M R, MOV R M

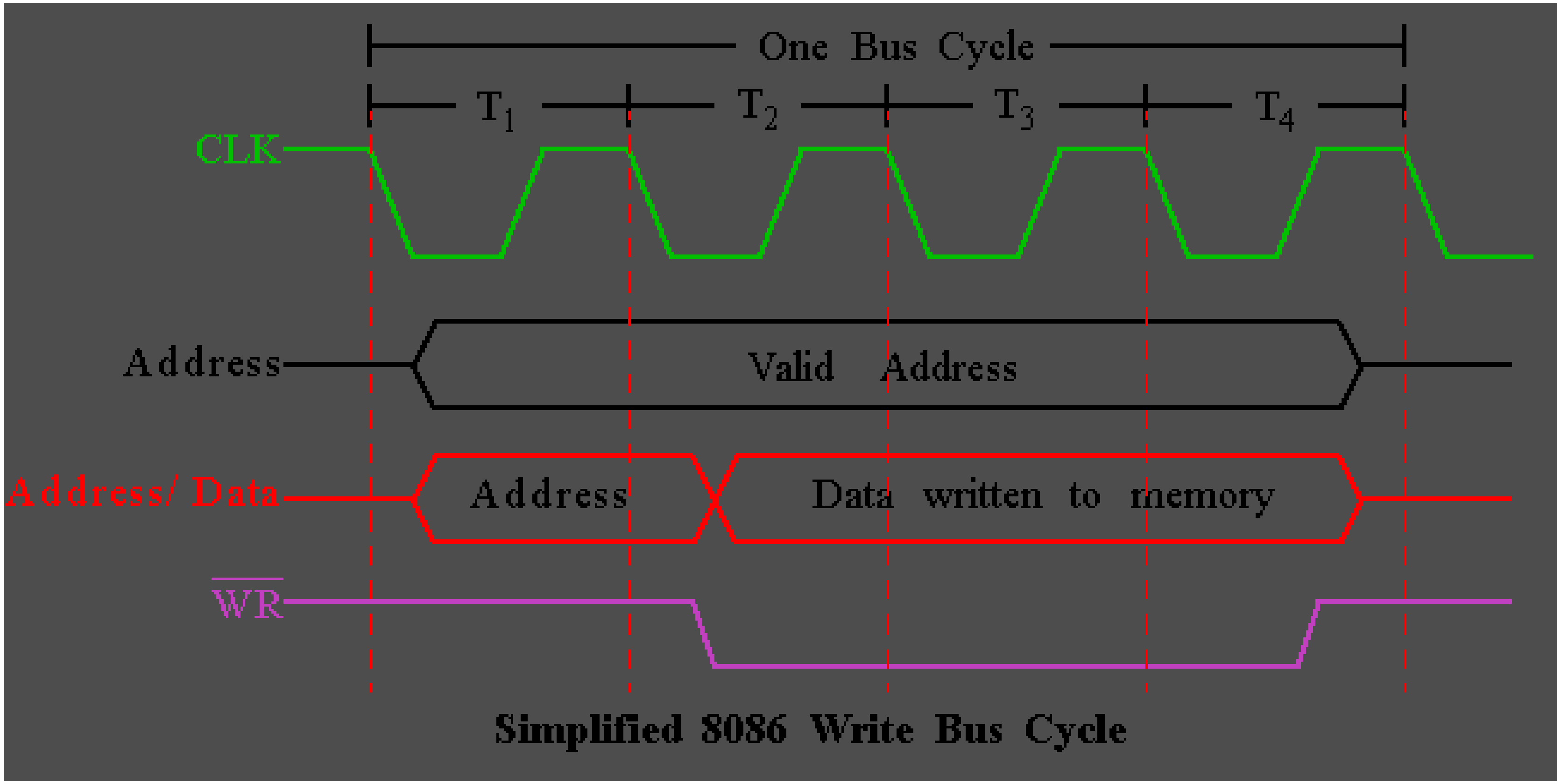
Mode of Data transfer

- Parallel
  - Entire 8bit or 16 bit transfer at one time
  - In 8085 entire 8 bit transferred simultaneous using 8 data lines
  - Seven Segment LEDs, Data converter (ASCIItoHEX), Memory
- Serial
  - Data transferred one bit at a time
  - Parallel to serial conversion (parallel 8 bit to stream of serial 8 bit)
  - Serial to Parallel conversion
  - Modem, USB, SATA, and (sometimes monitor/printer)
- UART: Universal Asynchronous Receiver & Transmitter

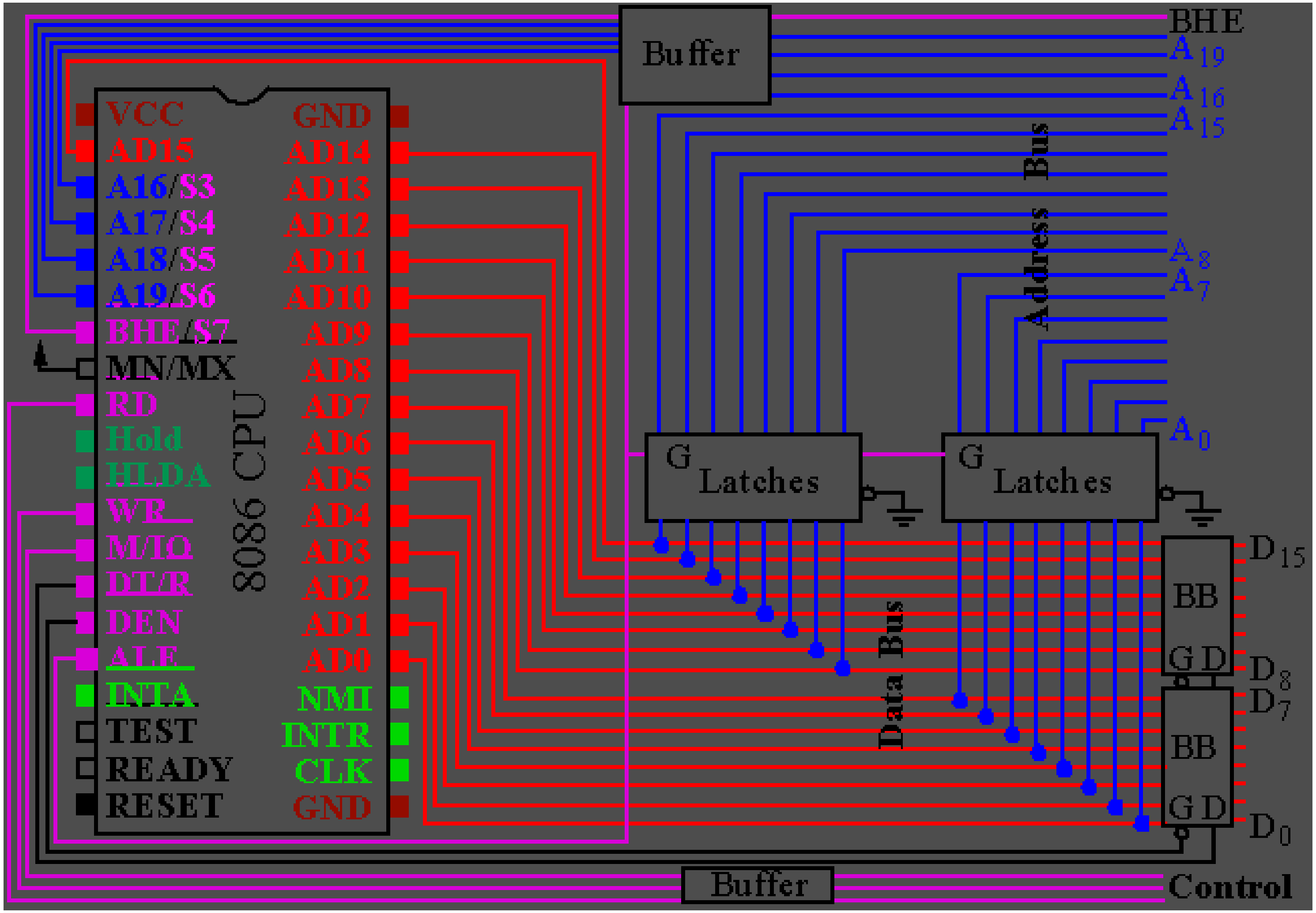
Timing Diagram : Read Cycle



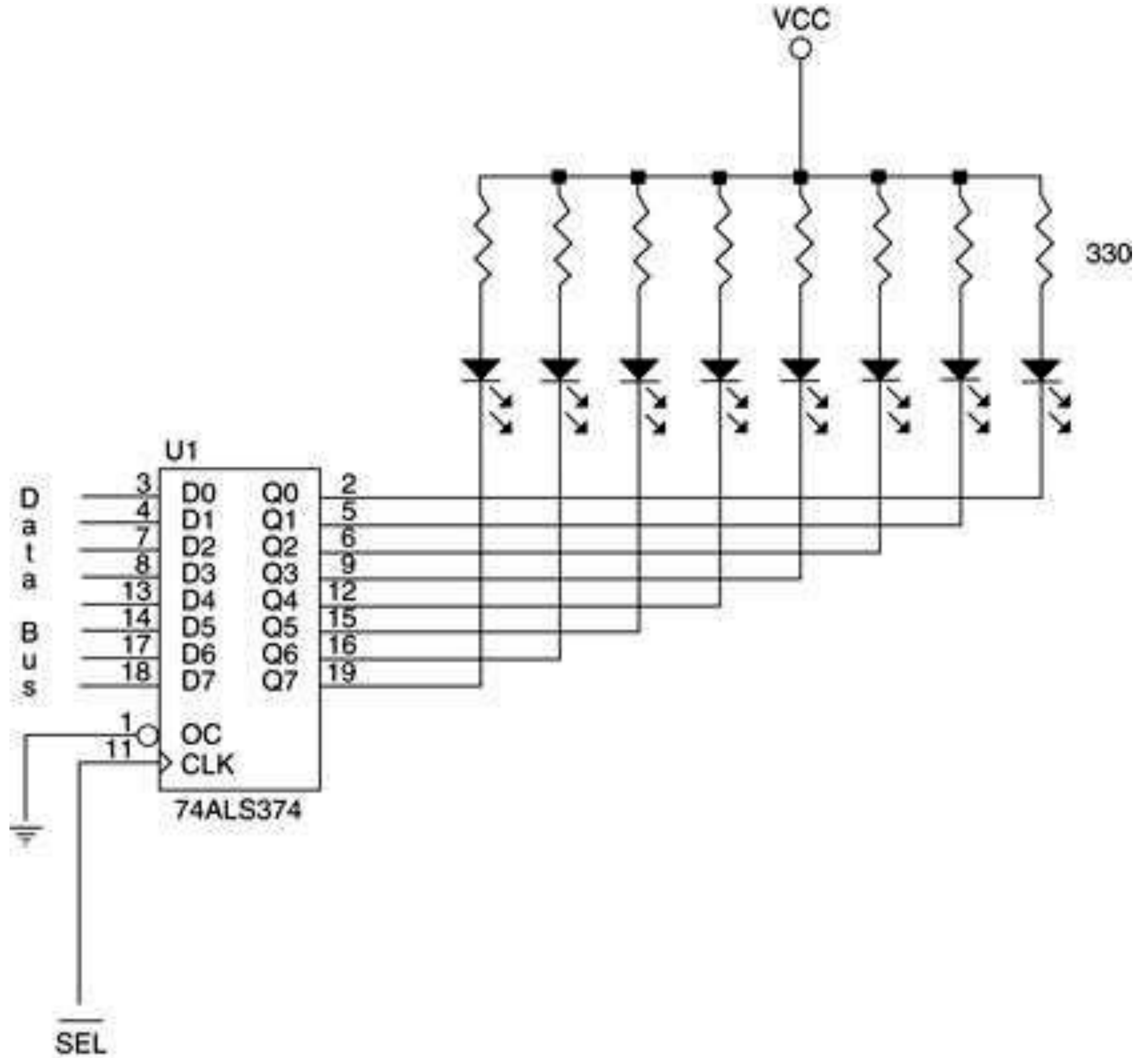
Timing Diagram : Write Cycle



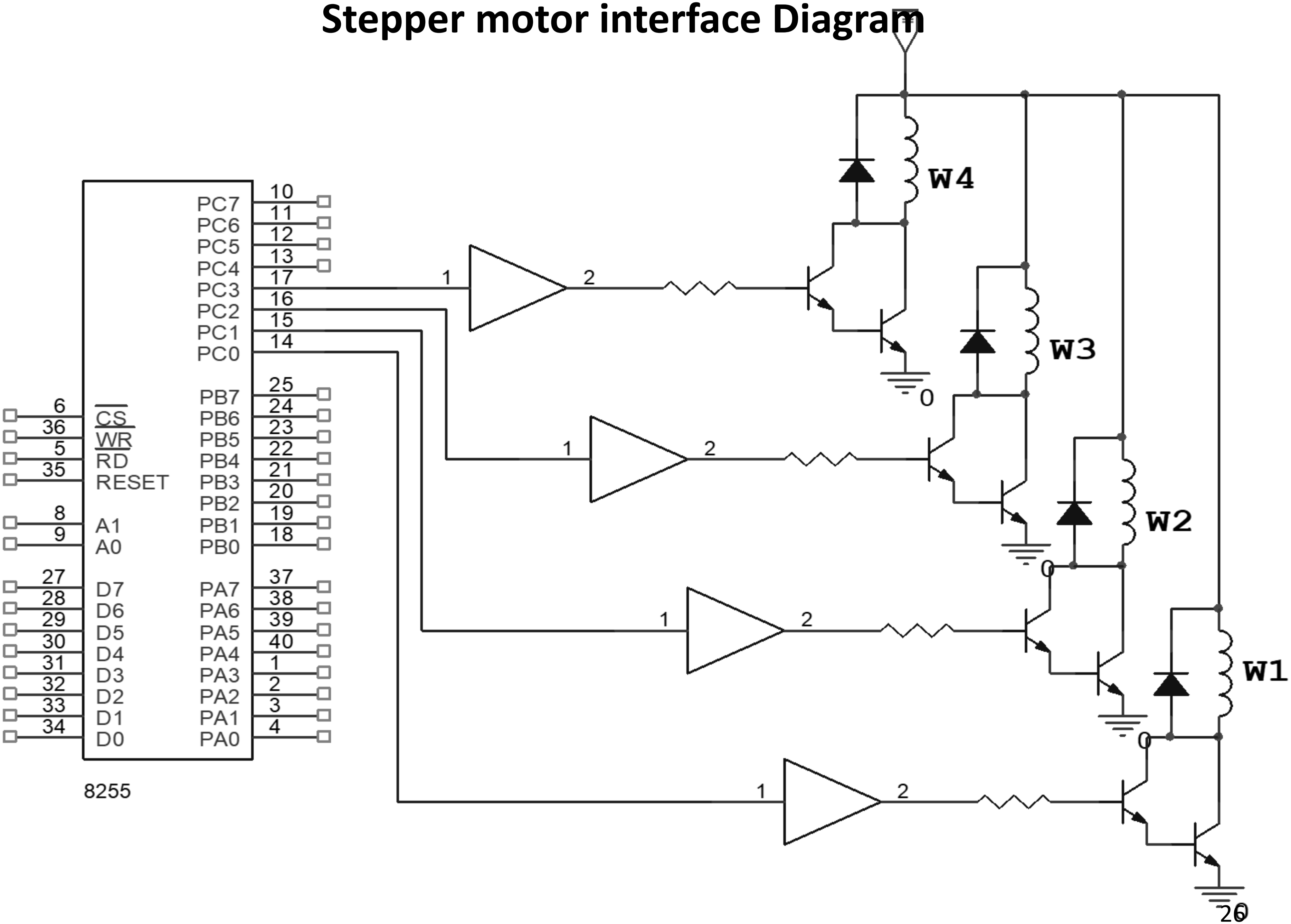
BUS Buffering and Latching



The basic output interface connected to a set of LED displays.



Stepper motor interface Diagram



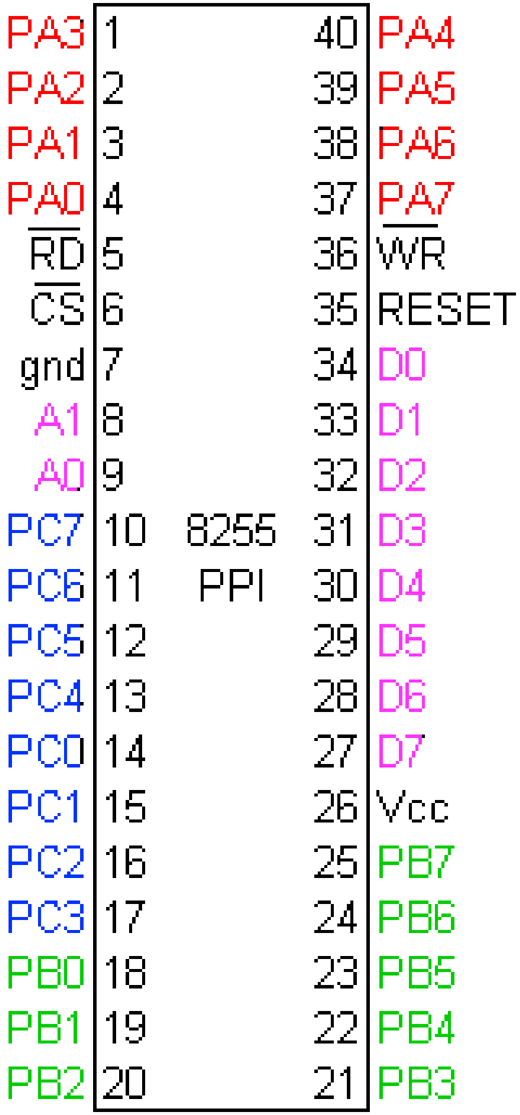
8255 PPI
Programmable Peripheral Interface

Outline

- **8255 PPI**
- **8255 PPI Pin Configuration**
- **8255 operating modes**
- **16-bit data bus to 8-bit peripherals**
- **MODE 0 Application (Keyboard)**
- **MODE 1 Application (Printer)**
- **MODE 2 Application (Printer)**

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purposes I/O component to Interface peripheral equipment to the microcomputer system bush. The functional configuration of the 8255A is programmed by the systems software.

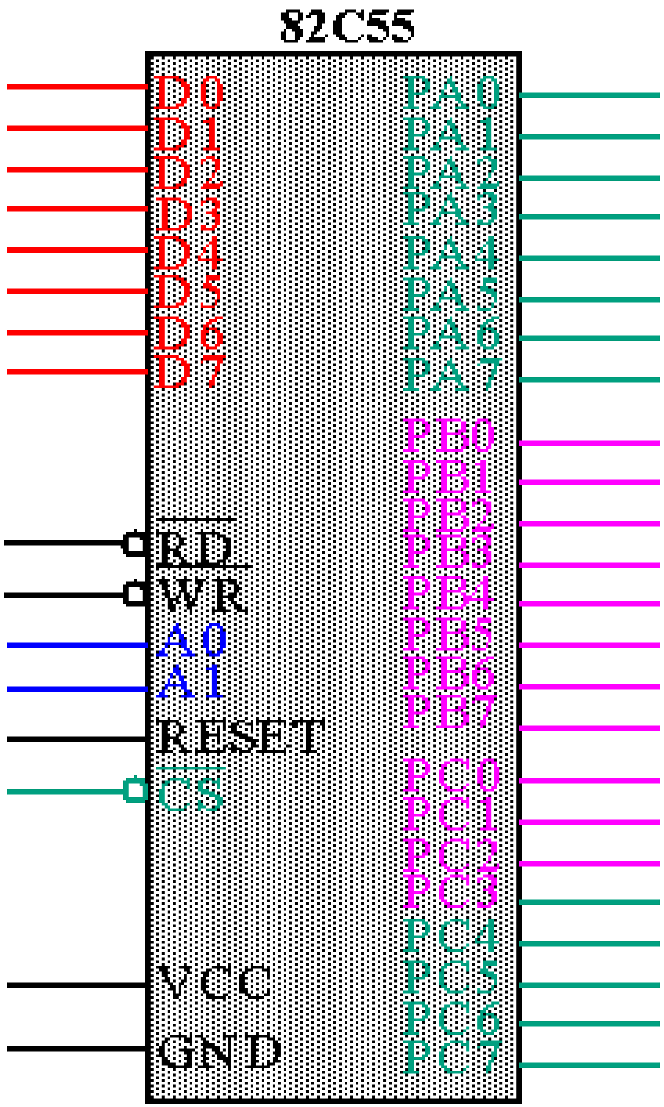
Pin Configuration



Pin Names

D_7–D_0	Data Bus (Bidirectional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0, A1	Port Address
PA7–PA0	Port A (bit)
PB7–PB0	Port B (bit)
PC7–PC0	Port C (bit)
V_{cc}	+5V
GND	0V

82C55 : Pin Layout



Group A

Port A (PA7-PA0) and upper half of port C (PC7 - PC4)

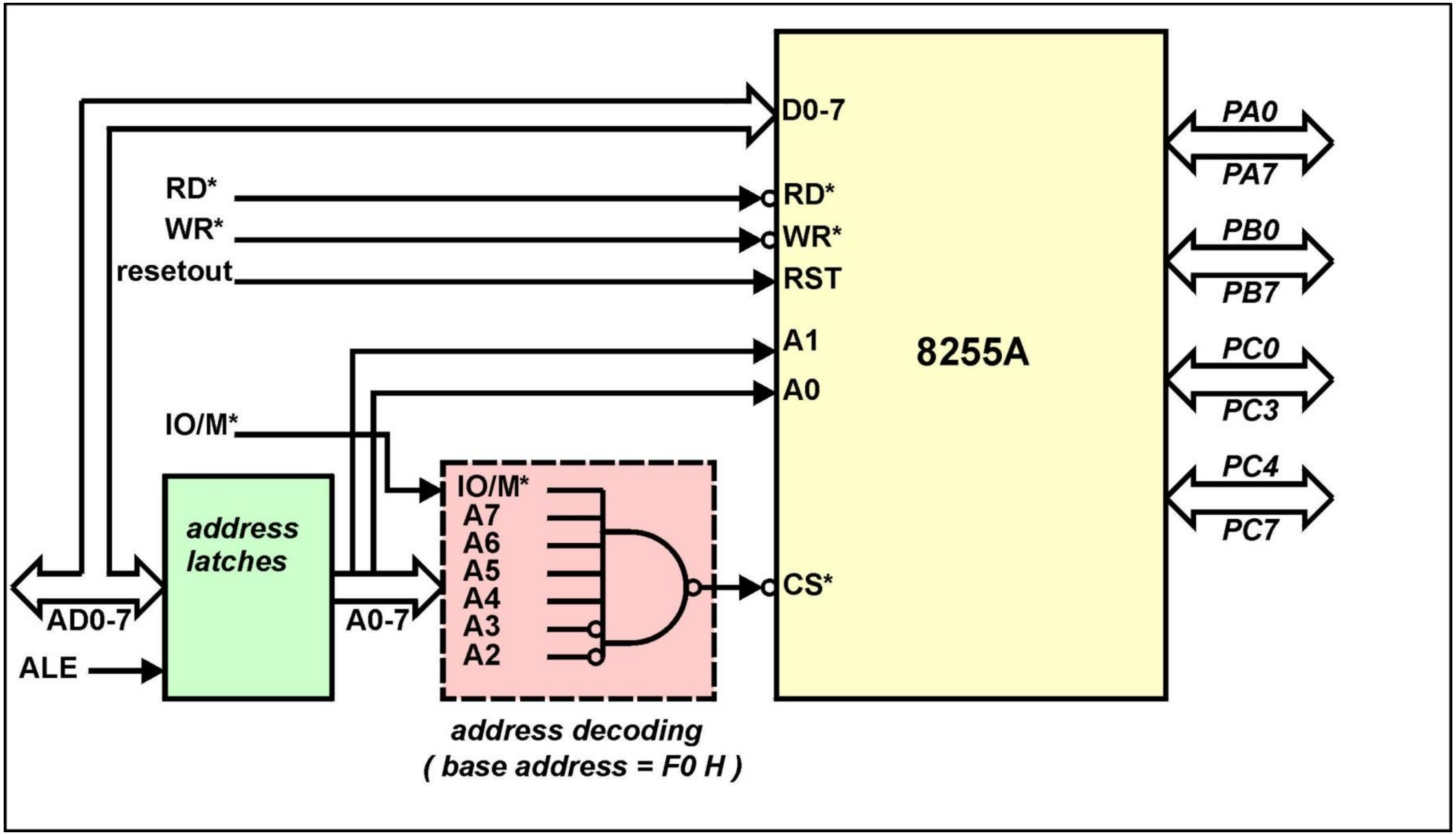
Group B

Port B (PB7-PB0) and lower half of port C (PC3 - PC0)

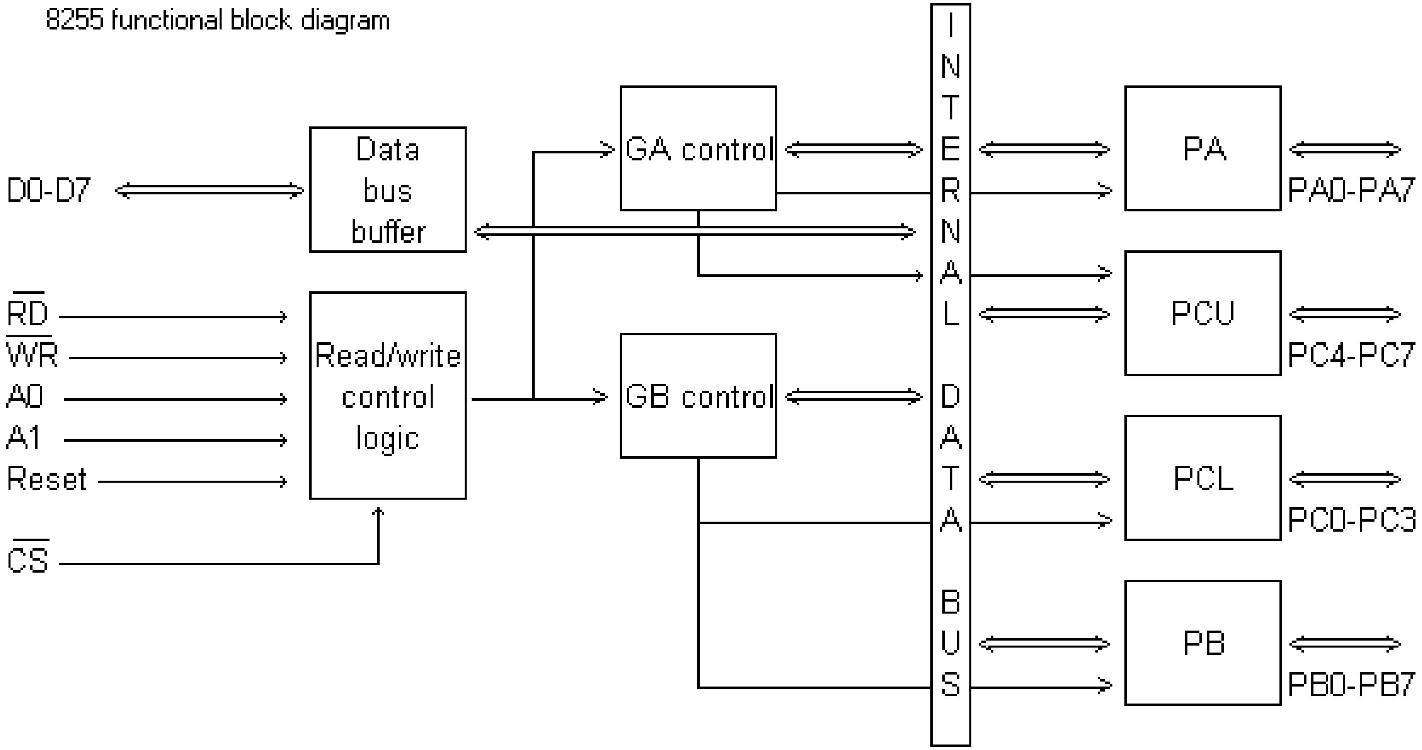
I/O Port Assignments

A_1	A_0	Function
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

8255A - 8085A Interface



8255 A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



Pin Configuration

- (**CS)Chip Select.** A “low’ on this input pin enables the communication between the 8255A, and the CPU.

- **(RD) Read.** A “low” on this Input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to “read from the 8255A.

- **(WR) Write.** A. “ low” on the input pin enables the CPU to write data or control words into the 8255A.

- **(A0 and A1)**

  Port Select 0 and Port Select 1. The Input signals, in conjunction with the RD and WR Inputs, controls the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A0 and A1).

Interface Registers

A1	A0	RD	WR	CS	Input Operation (Read)
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus

					Output Operation (Write)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control

Ports A, B and C

- **Ports A, B, and C**

  The 8255A contains three 8-bit ports (A , B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or personally to further enhance the power and flexibility of the 8255A.

- **Port A.** One 8 bit data output latch/buffer and one 8-bit data input latch.
- **Port B.** One 8-bit data output latch/buffer and one 8-bit data input buffer.
- **Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the controls signal outputs and status signal inputs in conjunction with ports A and B.

8255A OPERATIONAL DESCRIPTION

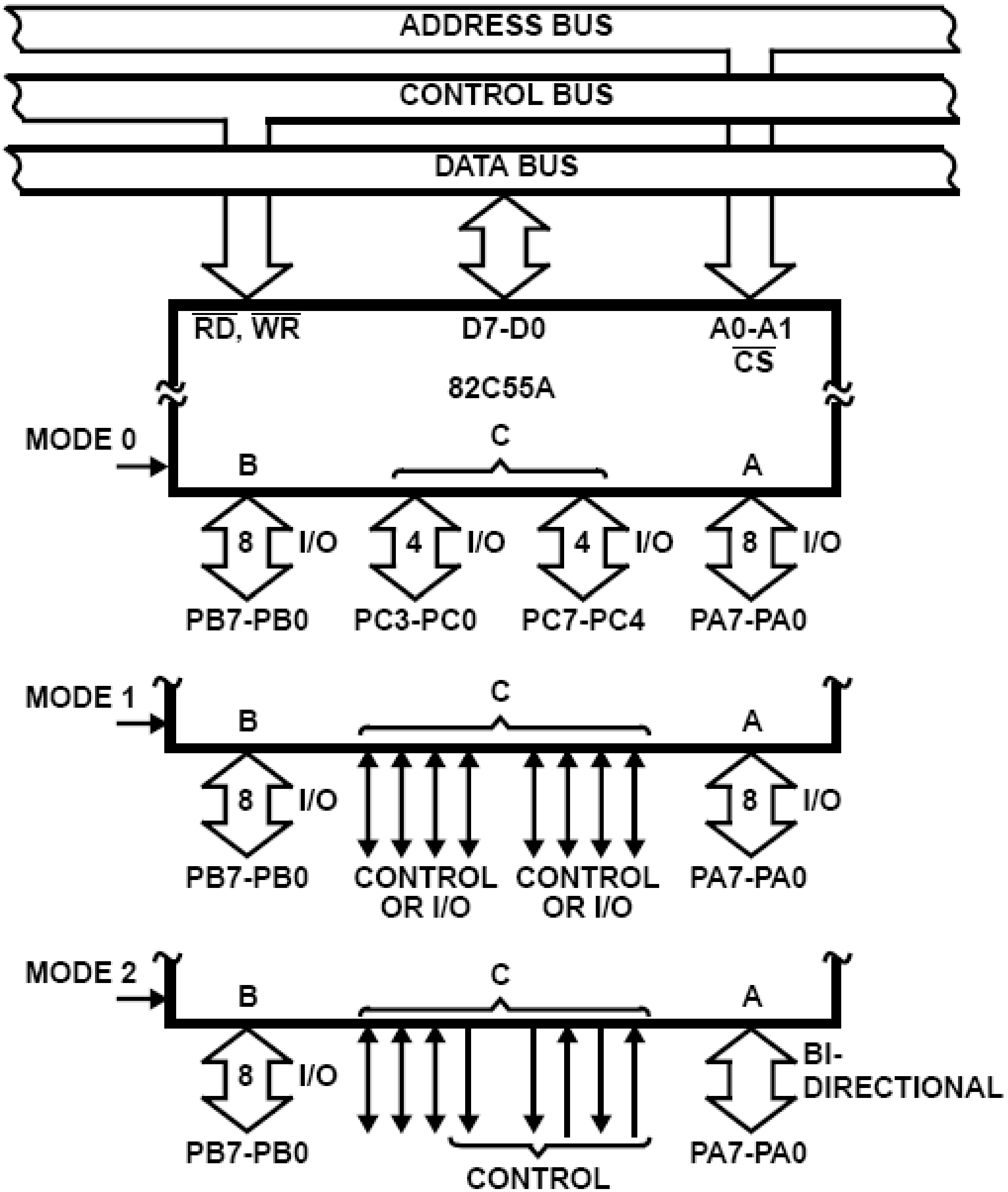
- **Mode Selection**

  There are three basic modes of operation that can be selected by the systems software:

Mode O – Basic Input/Output

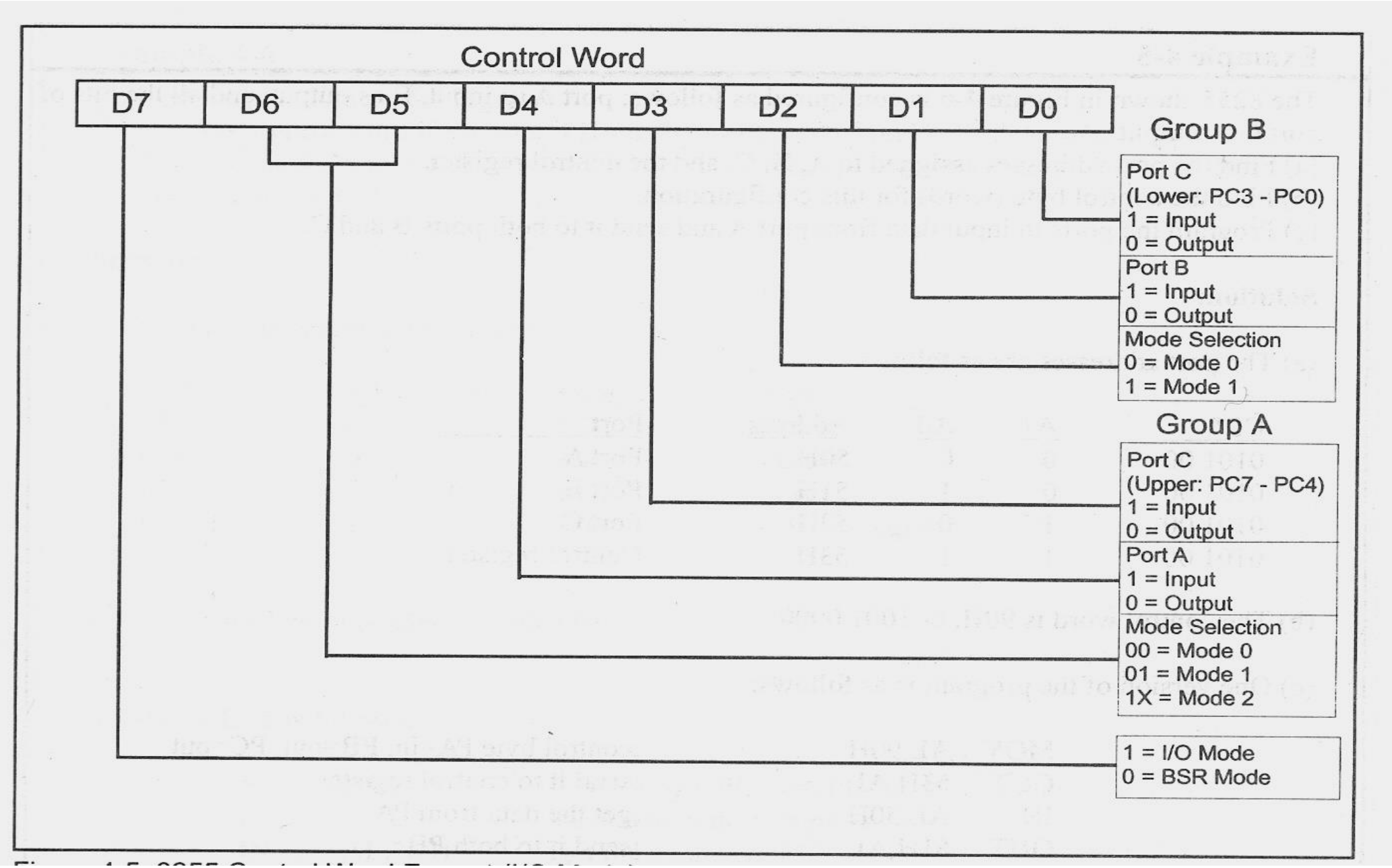
Mode 1 – Strobed Input/Output

Mode 2 – Bi-Directional Bus



8255 Control Word

Mode Definition Format

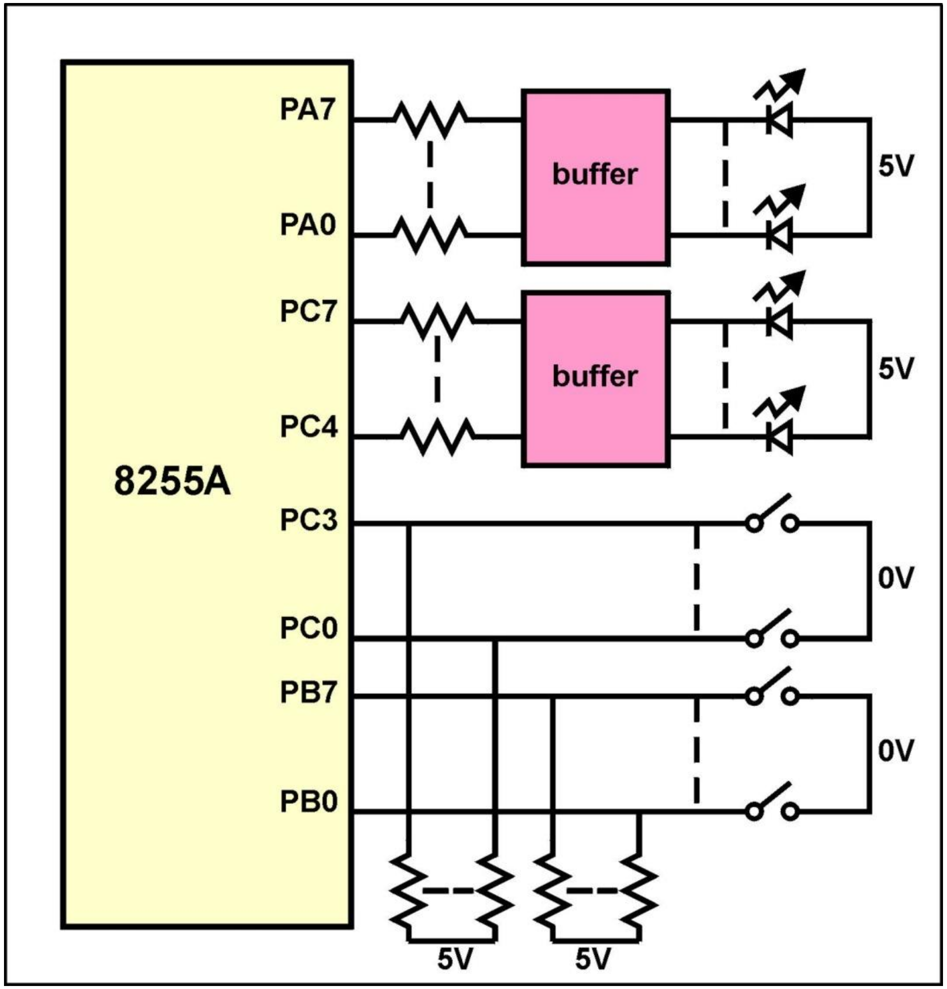


8255 Control Word

Control word bits								Control word	Port A	Port C upper	Port B	Port C lower
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0					
1	0	0	1	1	0	1	1	9B	input	input	input	input
1	0	0	1	1	0	1	0	9A	input	input	input	output
1	0	0	1	1	0	0	1	99	input	input	output	input
1	0	0	1	1	0	0	0	98	input	input	output	output
1	0	0	1	0	0	1	1	93	input	output	input	input
1	0	0	1	0	0	1	0	92	input	output	input	output
1	0	0	1	0	0	0	1	91	input	output	output	input
1	0	0	1	0	0	0	0	90	input	output	output	output
1	0	0	0	1	0	1	1	8B	output	input	input	input
1	0	0	0	1	0	1	0	8A	output	input	input	output
1	0	0	0	1	0	0	1	89	output	input	output	input
1	0	0	0	1	0	0	0	88	output	input	output	output
1	0	0	0	0	0	1	1	83	output	output	input	input
1	0	0	0	0	0	1	0	82	output	output	input	output
1	0	0	0	0	0	0	1	81	output	output	output	input
1	0	0	0	0	0	0	0	80	output	output	output	output

Example of 8255A Operating in Mode 0

- **Write a program that reads the state of the 12 switches and displays the switch state on the 12 LEDs**

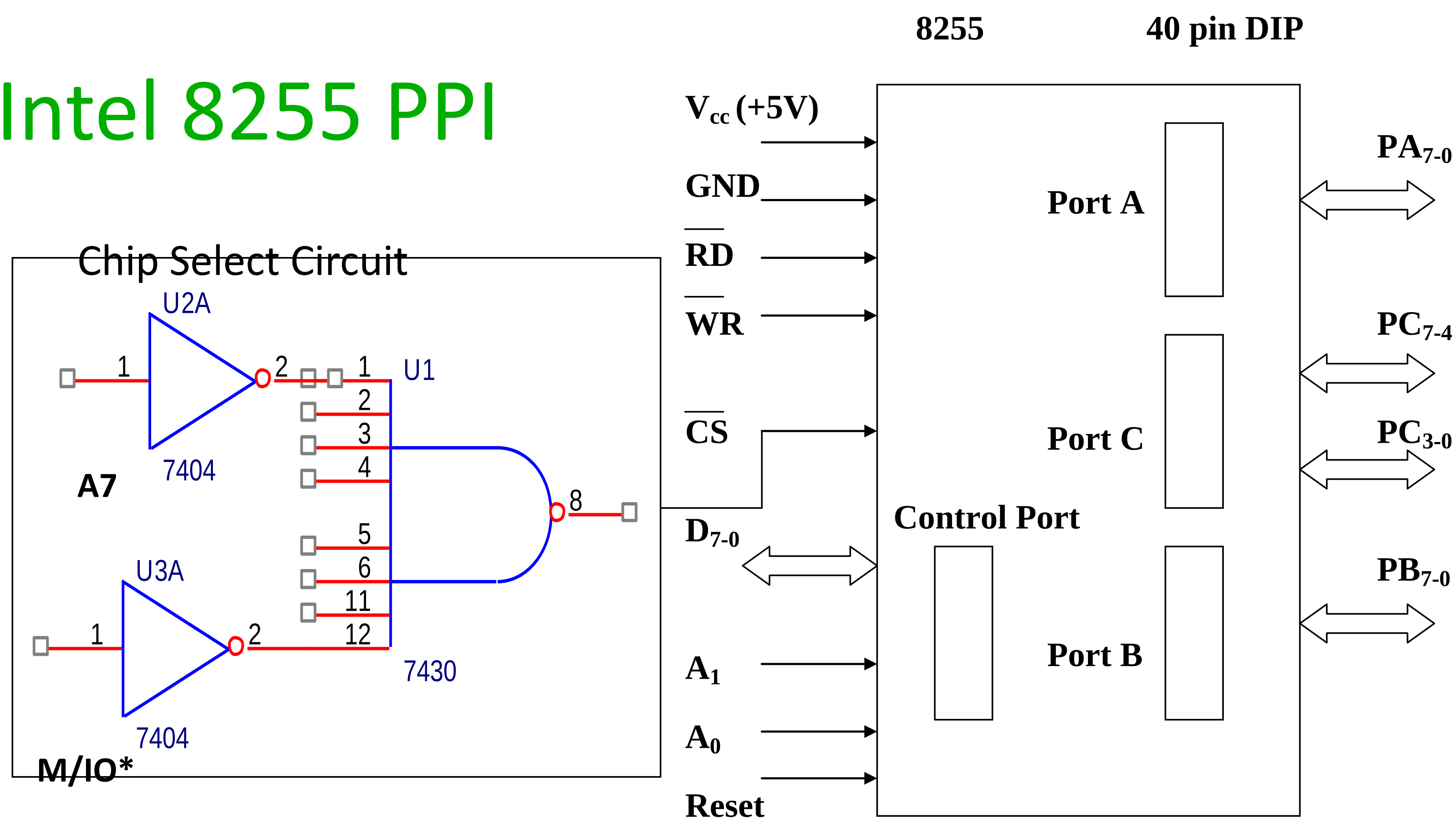


Example of 8255A Operating in Mode 0

- **The assembly language program is :**

```
MVI   A, 83 H   ;A= out  B= in  C(lo)= in  C(hi)= out
OUT   F3 H      ; write the control word
IN    F1 H      ; read switches on port B
OUT   F0 H      ; display on port A
IN    F2 H      ; read switches on port C (lo)
ANI   0F H      ; mask top 4-bits  -  not switch data
RLC
RLC
RLC
RLC             ; move bottom 4-bits of A to top
OUT   F2 H      ; output reg A to port C
```

Intel 8255 PPI



A7=0, A6=1, A5=1, A4=1, A3=1, A2=1, & M/IO*= 0

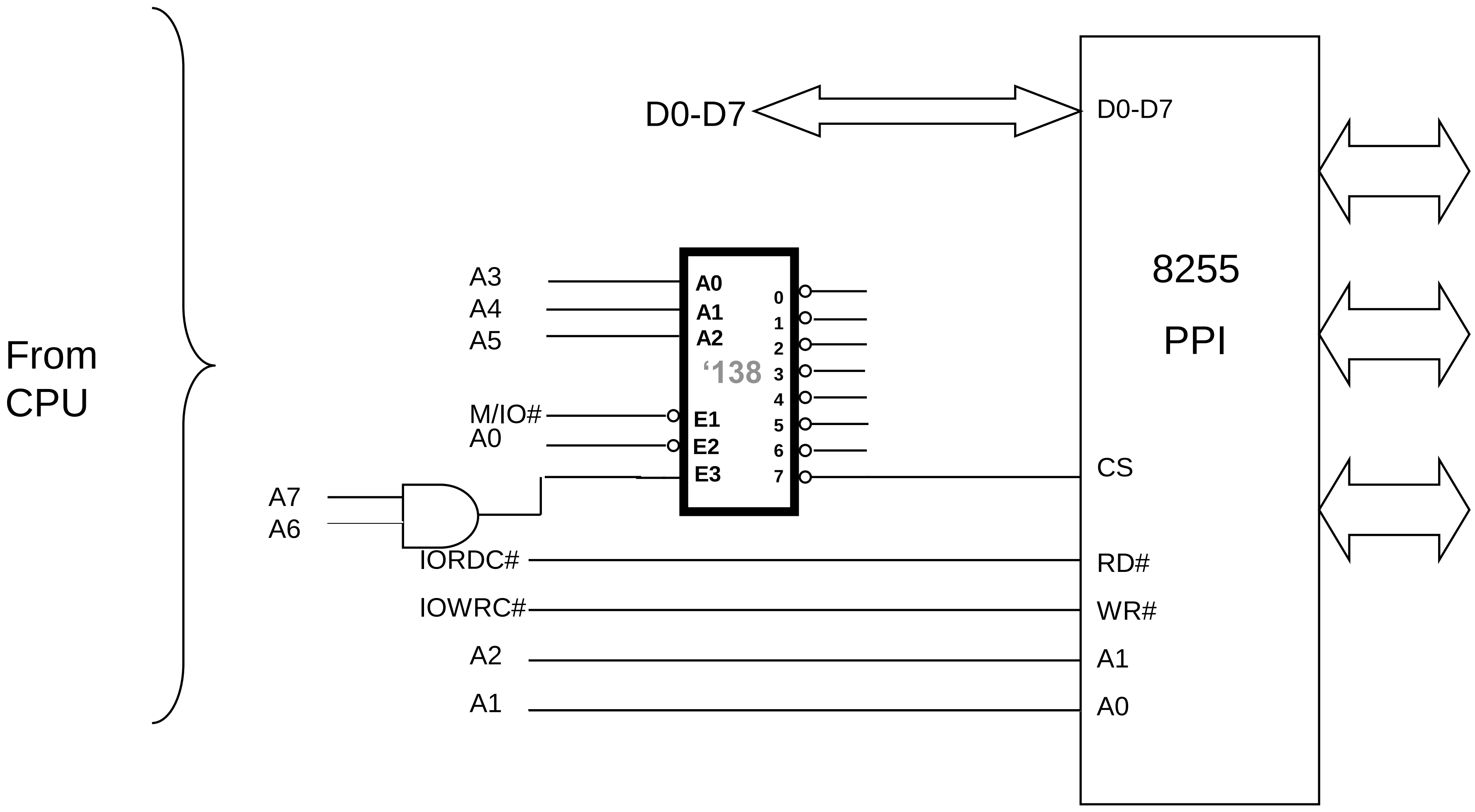
8255 –port address

When CS (Chip select) is 0, 8255 is selected for communication by the processor. The chip select circuit connected to the CS pin assigns addresses to the ports of 8255.

For the chip select circuit shown, the chip is selected when A7=0, A6=1, A5=1, A4=1, A3=1, A2=1, & M/IO*= 0

Port A, Port B, Port C and Control port will have the addresses as 7Ch, 7Dh, 7Eh, and 7Fh respectively.

Ex. :Circuit Diagram



Use only even addresses

- Example: We want to use a 8255 PPI with the starting I/O address of F8h. Use even adresses only.

	A7	A6	A5	A4	A3	A2	A1	A0		
f8h	1	1	1	1	1	0	0	0	B	: Port A
fah	1	1	1	1	1	0	1	0	B	: Port B
fch	1	1	1	1	1	1	0	0	B	: Port C
feh	1	1	1	1	1	1	1	0	B	: Control Reg.

Register Select

Access to Interface Registers

- Port B and C are programmed as Mode 0 input port.
- Port A is programmed as Mode 0 simple latched output port.
- Write a code to implement the operation

PortA=**PortB-PortC**

```
mov AL,8Bh      ;control word
out  FEh,AL     ;written to control reg.
in   AL,FCh     ;Read Port C
mov BL,AL       ;
in   AL,FAh     ;Read Port B
sub  AL,BL      ;PortB-PortC
out  F8h,AL     ;write PortA
```

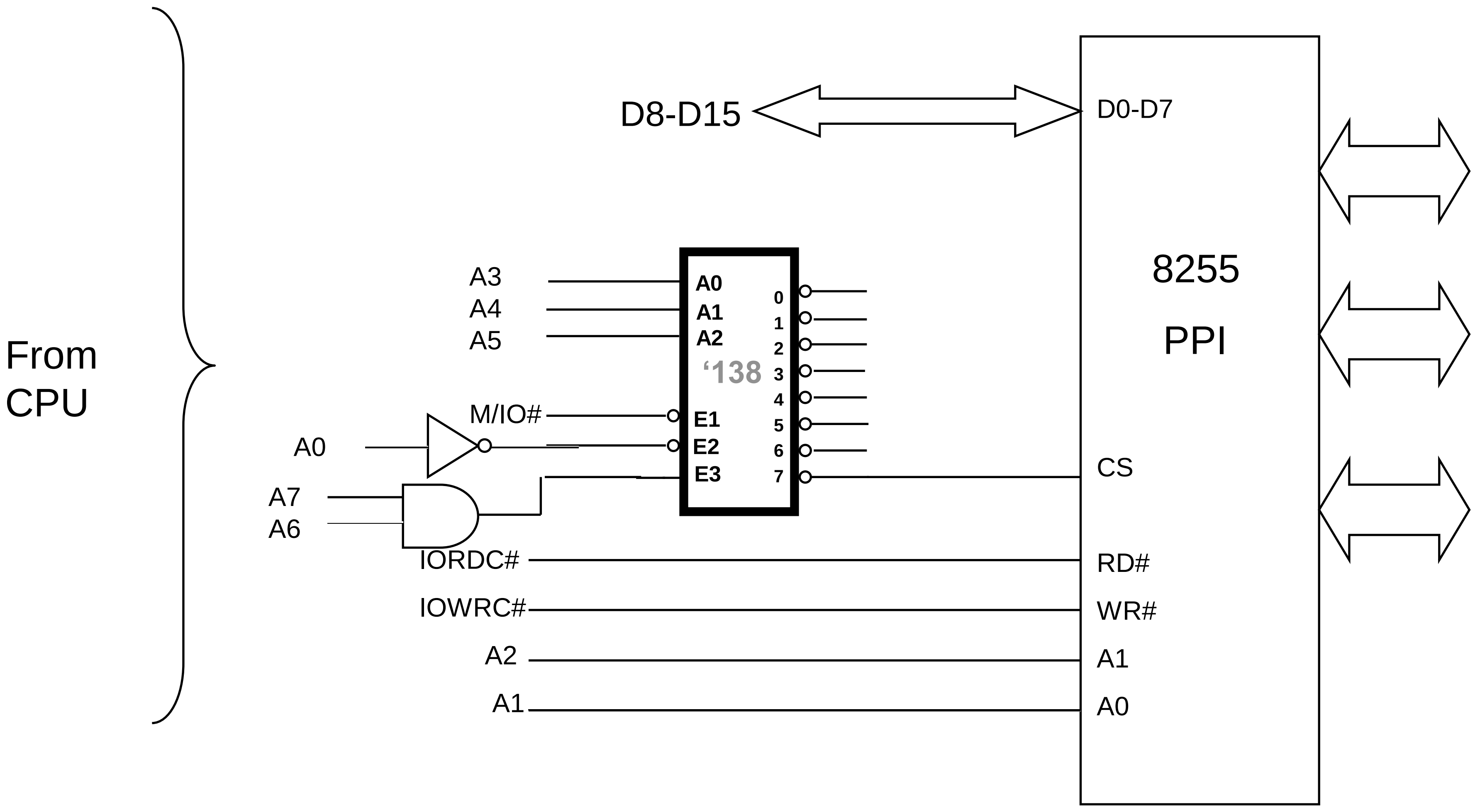
Solution 2: Use only odd addresses

Example: We want to use a 8255 PPI with the starting I/O address of F9h. Use odd adresses only.

	A7	A6	A5	A4	A3	A2	A1	A0		
f9h	1	1	1	1	1	0	0	1	B	: Port A
fbh	1	1	1	1	1	0	1	1	B	: Port B
fdh	1	1	1	1	1	1	0	1	B	: Port C
ffh	1	1	1	1	1	1	1	1	B	: Control Reg.

Register Select

Circuit Diagram



Solution 3: Use consecutive even and odd address

Example: We want to use a 8255 PPI with the starting I/O address of C0h. Use even and odd adresses.

	A7	A6	A5	A4	A3	A2	A1	A0		
C0h	1	1	0	0	0	0	0	0	B	: Port A
C1h	1	1	0	0	0	0	0	1	B	: Port B
C2h	1	1	0	0	0	0	1	0	B	: Port C
C3h	1	1	0	0	0	0	1	1	B	: Control Reg.

Register Select

8255 MD Control word Contd.

Ex. 1: Configure Port A as i/p in Mode 0, Port B as o/p in mode 0, Port C (Lower) as o/p and Port C (Upper) as i/p ports.

Required MD control word:

1	0	0	1	1	0	0	0
MD control	PA in Mode 0		PA as i/p	PC Upper as i/p	PB in Mode 0	PB as o/p	PC Lower as o/p

= 98H

Reqd. instrns.

```
MOV AL, 98H
OUT 7FH, AL
```

8255 MD Control word Contd.

Ex. 2: Configure Port A as i/p in Mode 1, Port B as o/p in mode 1, Port C7-8 as i/p ports. (PC5-0 are handshake lines, some i/p lines and others o/p. So they are shown as X)

Required MD control word:

1	0	1	1	1	1	0	X

= BCH or BDH

MD control

PC3-0 as don't care

PA in Mode 1

PB as o/p

PA as i/p

PB in Mode 1

PC Upper(C7-8) as i/p

Reqd. Instrns.

MOV AL,BCH

OUT 7FH, AL

Example - Port addresses

Example 4-5

The 8255 shown in Figure 4-6 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.

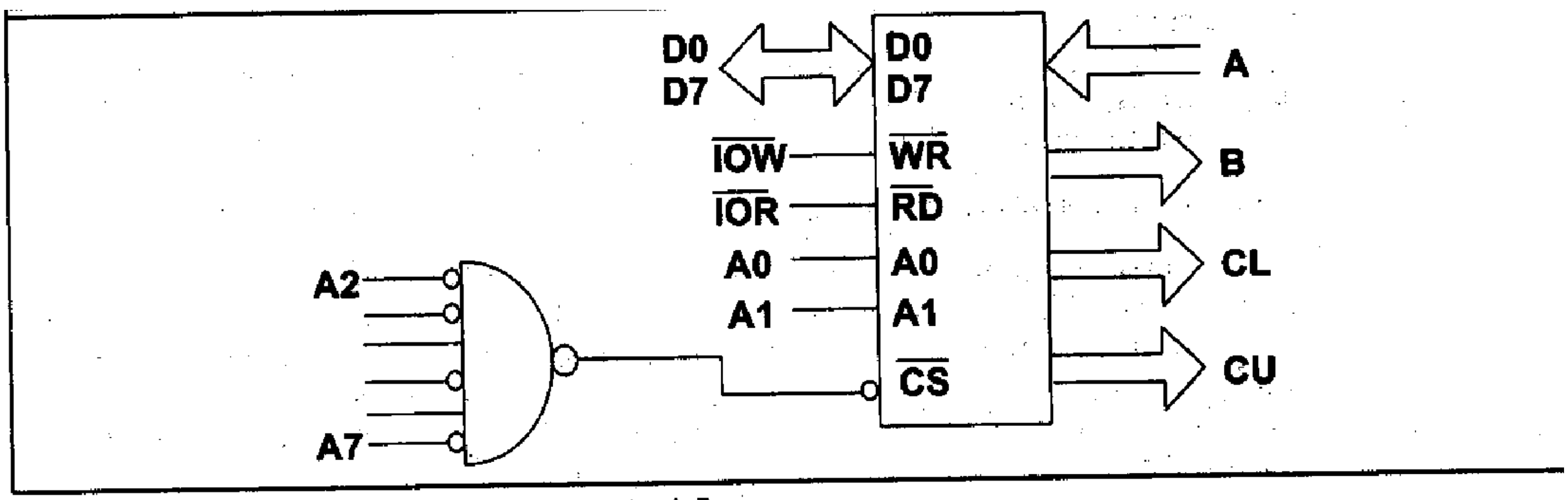


Figure 4-6. 8255 Configuration for Example 4-5

Solution

Solution:

(a) The port addresses are as follows:

CS*	A1	A0	Address	Port
0101 00	0	0	50H	Port A
0101 00	0	1	51H	Port B
0101 00	1	0	52H	Port C
0101 00	1	1	53H	Control register

(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
        OUT   53H,AL          ;send it to control register
        IN    AL,50H          ;get the data from PA
        OUT   51H,AL          ;send it to both PB
        OUT   52H,AL          ; and PC
```

Using the EQU directive one can rewrite the above program as follows:

```
PORTA       EQU   50H
PORTB       EQU   51H
PORTC       EQU   52H
CNTLREG     EQU   53H
            ...
            MOV   AL,90H          ;control byte PA=in, PB=out, PC=out
            OUT   CNTLREG,AL      ;send it to control register
            IN    AL,PORTA        ;get the data from PA
            OUT   PORTB,AL        ;send it to both PB
            OUT   PORTC,AL        ; and PC
```

Example – Programming 8255

Example 4-6

(a) Find the port address for Figure 4-7.

(b) Find the control word if PA =out, PB=in, PC0 - PC3 =in, and PC4 - PC7=out.

(c) Program the 8255 to get data from port A and send it to port B. In addition, data from PCL is send out to the PCU.

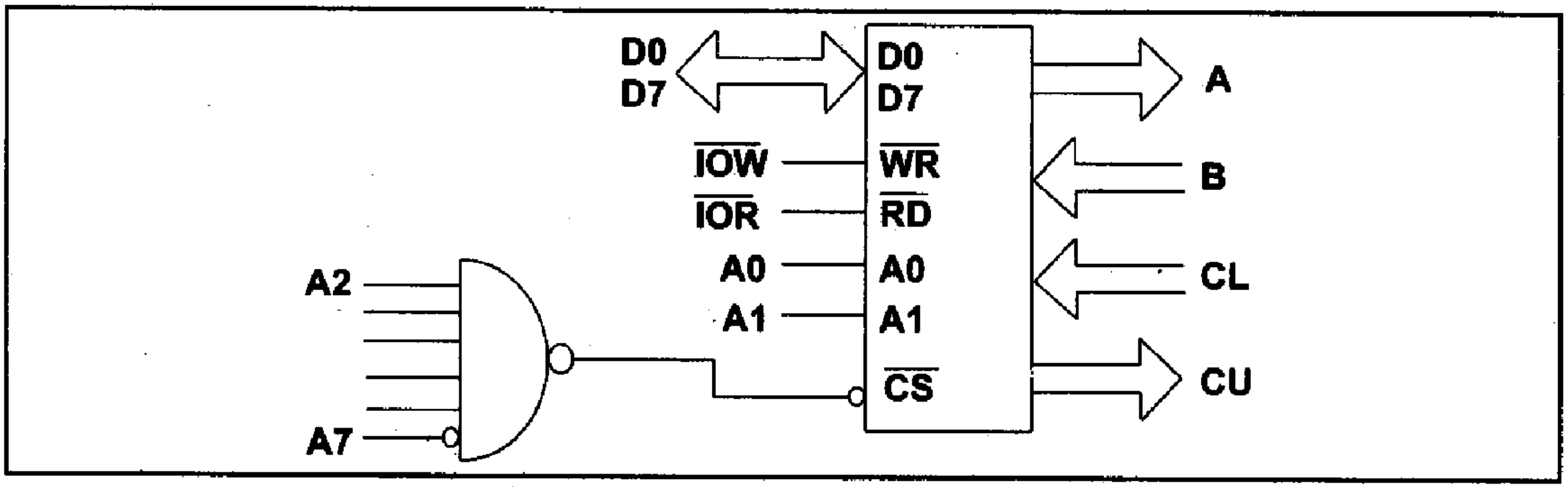


Figure 4-7. Configuration for Example 4-6

Solution

(a) The port addresses are as follows:

CS*	A1	A0	Address	Port
0111 11	0	0	7CH	Port A
0111 11	0	1	7DH	Port B
0111 11	1	0	7EH	Port C
0111 11	1	1	7FH	Control register

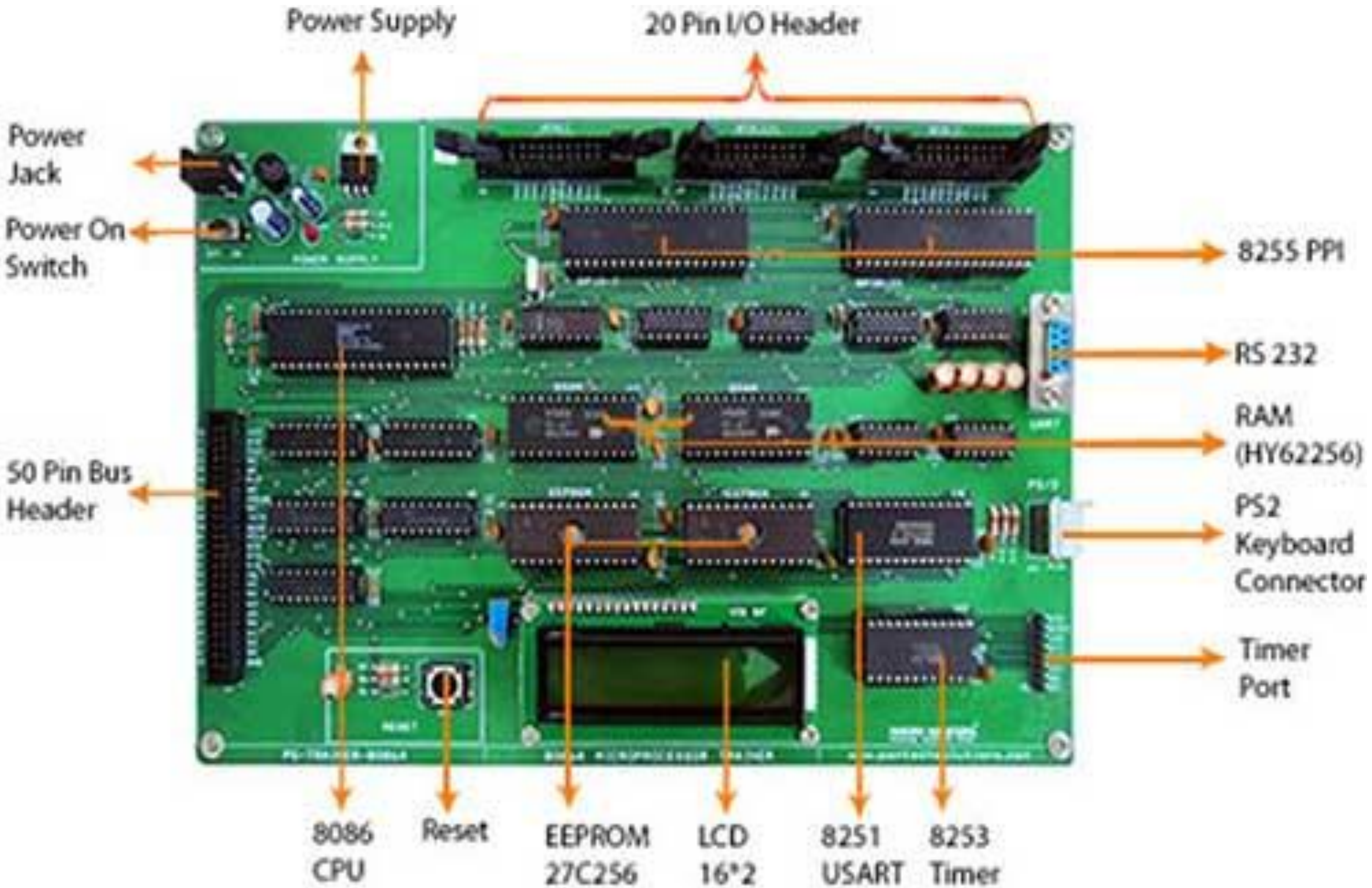
(b) The control word is 83H, or 1000 0011.

(c) The code is as follows.

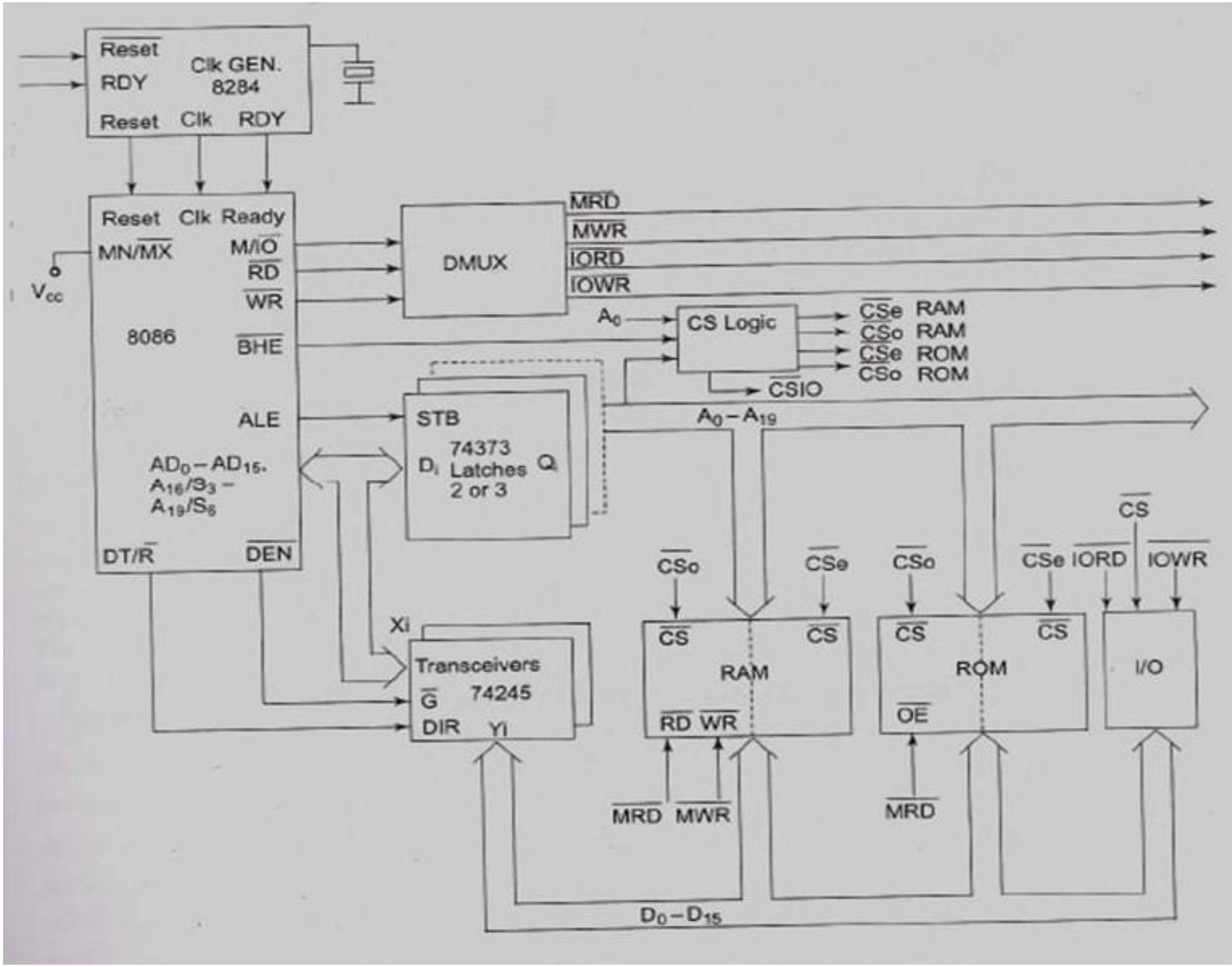
```
MOV   AL,83H     ;control byte PA=out, PB=in, PCL=in, PCU=out
OUT   7FH,AL     ;send it to control register
IN    AL,7DH     ;get the data from PB
OUT   7CH,AL     ;send it to PA
IN    AL,7EH     ;get the bits from PCL
AND   AL,0FH     ;mask the upper bits
ROL   AL,1
ROL   AL,1       ;shift the bits
ROL   AL,1       ;to upper position
ROL   AL,1
OUT   7EH,AL     ;send it to PCU
```

Alternately, the four instructions above of "ROL AL,1" could be replaced with the following two instructions:

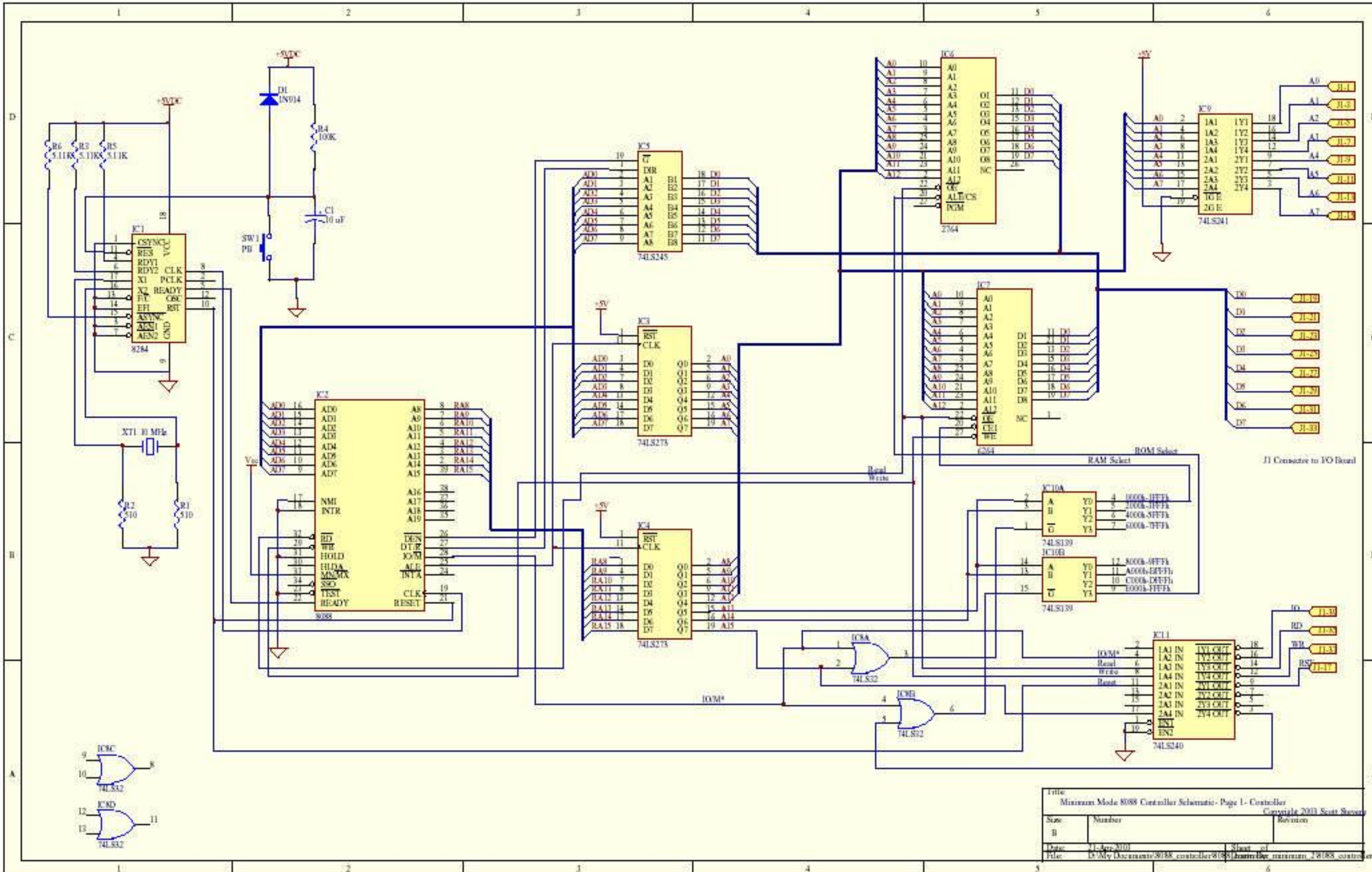
```
MOV   CL,4       ;count = 4
ROL   AL,CL      ;rotate 4 times
```



Power Supply
20 Pin I/O Header
Power
Jack
Power On
Switch
8255 PPI
RS 232
RAM
(HY62256)
50 Pin Bus
Header
PS2
Keyboard
Connector
Timer
Port
8086
CPU
Reset
EEPROM
27C256
LCD
16*2
8251
USART
8253
Timer

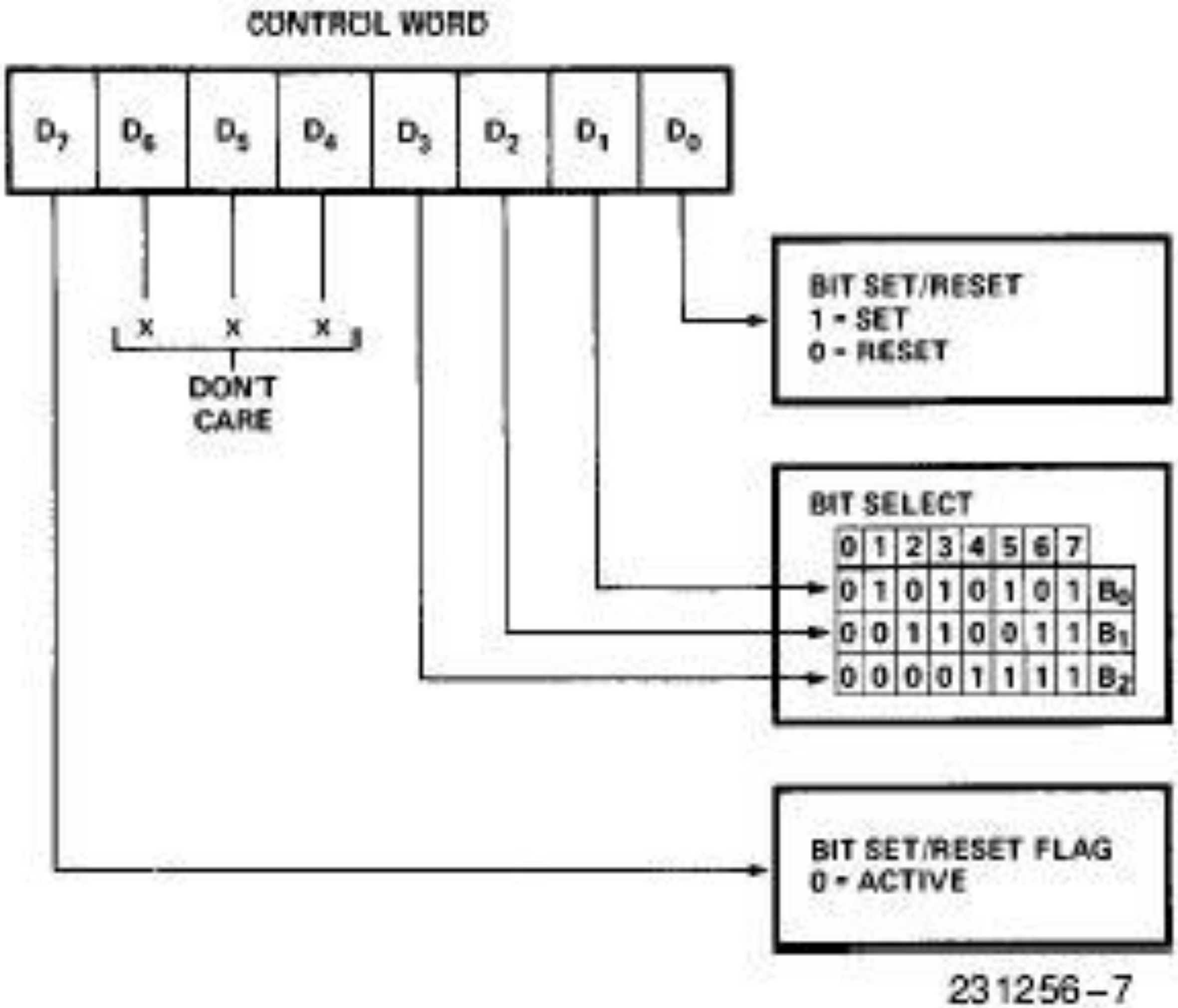


Reset
RDY
Clk GEN.
8284
Reset Clk RDY
Reset Clk Ready
MN/MX
M/IO
RD
WR
Vcc
8086
BHE
ALE
AD0 – AD15,
A16/S3 –
A19/S6
DT/R
DEN
DMUX
MRD
MWR
IORD
IOWR
A0
CS Logic
CSe RAM
CSo RAM
CSe ROM
CSo ROM
CSIO
A0 – A19
STB
Di
74373
Latches
2 or 3
Q
Xi
Transceivers
G
74245
DIR
Yi
CSo
CSe
CS
RAM
RD WR
MRD MWR
ROM
OE
I/O
IORD
IOWR
D0 – D15



IC1
8284
IC2
8088
IC3
74LS273
IC4
74LS273
IC5
74LS245
IC6
2764
IC7
6264
IC8A
74LS32
IC8B
74LS32
IC8C
74LS32
IC8D
74LS32
IC9
74LS241
IC10A
74LS139
IC10B
74LS139
IC11
74LS240
XT1 10 MHz
D1 1N914
R4 100K
C1 10 uF
SW1 PB
R6 5.11K
R3 5.11K
R5 5.11K
R2 510
R1 510
+5VDC
+5V
Vcc
ROM Select
RAM Select
Read
Write
IO/M*
Reset
0000h-1FFFh
2000h-3FFFh
4000h-5FFFh
6000h-7FFFh
8000h-9FFFh
A000h-BFFFh
C000h-DFFFh
E000h-FFFFh
J1 Connector to I/O Board
Title
Minimum Mode 8088 Controller Schematic - Page 1 - Controller
Copyright 2003 Scott Stevens
Size
B
Number
Revision
Date
21-Apr-2003
Sheet of
File
D:\My Documents\8088_controller\8088_controller_minimum_2\8088_controller.ddb

Bit Set Reset (BSR) mode



Example for BSR

- Program 8255 for the following
  - A) set PC2 to high
  - B) Use PC6 to generate a square wave of 66% duty cycle
- Solution
- A)

```
MOV AL, 00000101B
OUT 92H,AL
```

- B)

```
AGAIN      MOV AL, 0xxx1101
           OUT 92H, AL
           CALL Delay
           CALL Delay
           MOV AL, 0xxx1100
           OUT 92H, AL
           CALL Delay
           JMP AGAIN
```

Mode 0: Example

This functional configuration provides simple input and output operations for each of the three ports. No ``handshaking'' is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 Application (Keyboard Interface)

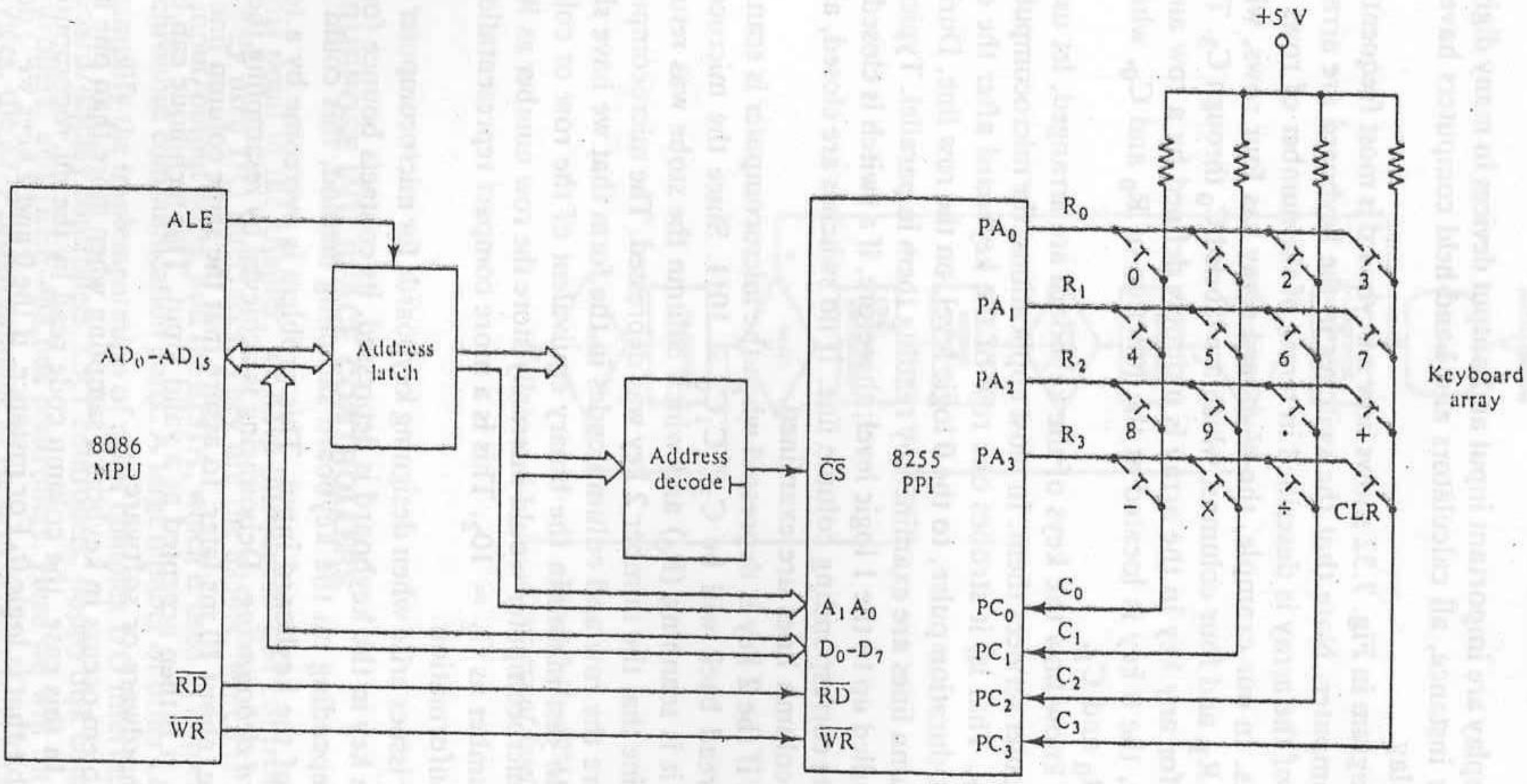


Figure 7.32 Keyboard interfaced to a microcomputer.

MODE 0 Application (Keyboard Interface)

- The switches in the keyboard are arranged in an array. The size of the array is described in terms of the number of rows and the number of the columns.
- In our example, the keyboard array has four rows, which are labeled R_0 through R_3, and four columns, which are labeled C_0 through C_3. The location of the switch for any key in the array is uniquely defined by a row and a column.
- For instance, the 0 key is located at the junction of R_0 and C_0, while the 1 key is located at R_0 and C_1.
- In most applications, the microcomputer scans the keyboard array. That is, it strobes one row of the keyboard after the other by sending out a short-duration pulse, to the 0 logic level, on the row line. During each row strobe, all column lines are examined by reading them in parallel.
- Typically, the column lines are pulled up to the 1 logic level; therefore, if a switch is closed, a logic 0 will be read on the corresponding column line. If no switches are closed, all 1s will be read when the lines are examined.

MODE 0 Application (Keyboard Interface)

- The starting address for this I/O interface is 10H and consecutive even addresses are used.

10h:	0 0 0 1	0 0 0 0B	-Port A (Output port)
12h:	0 0 0 1	0 0 1 0B	-Port B (Unused output port)
14h:	0 0 0 1	0 1 0 0B	-Port C (lower and higher input)
16h:	0 0 0 1	0 1 1 0B	-Control Reg.

```
PORTA   EQU   10h
PORTB   EQU   12h
PORTA   EQU   14h
CREG    EQU   16h
CWD     EQU   10001001b
MOV     AL, CWD
OUT     CREG,AL
```

MODE 0 Application (Keyboard Interface)

```
                    .

          .
SCAN:     MOV   BL,FEH      ; send a short-duration pulse, to the 0 logic level,
SCAN1:    MOV   AL,BL         ; on the row line0.
          OUT   PORTA,AL
          IN    AL,PORTC      ;Read PortC
          XOR   AL,FFH       ;Complement AL
          AND   AL,0FH        ;Mask unused nibble
          CMP   AL,0
          JNE   KEY           ;if a key pressed go to KEY
          ROL   BL,1          ; if no key pressed, shift the ruration pulse to next row
          CMP   BL,FEH
          JNE   SCAN1
          JMP   SCAN
          .
          .
KEY:      .
          .
```

Mode 0 Application: Display Interface

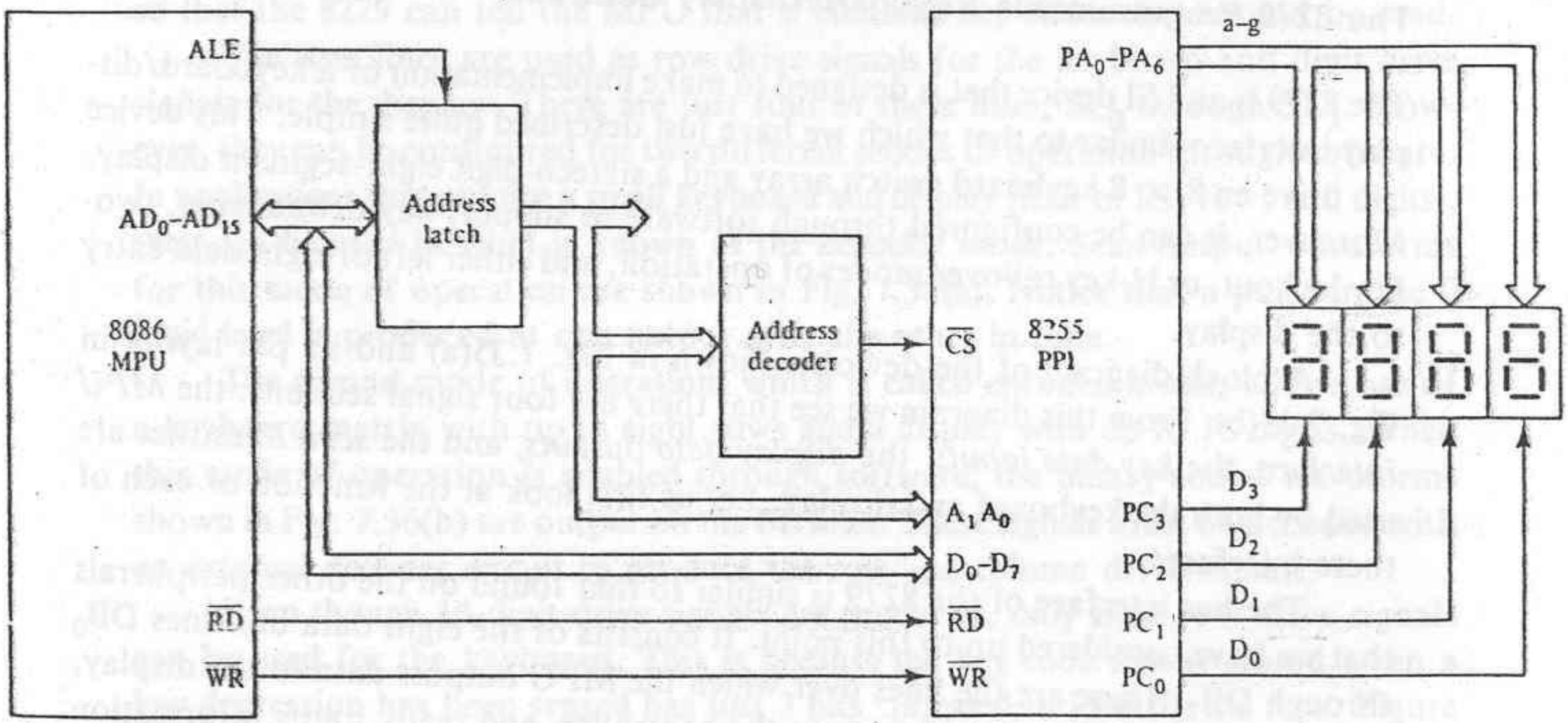


Figure 7.33 Display interfaced to a microcomputer.

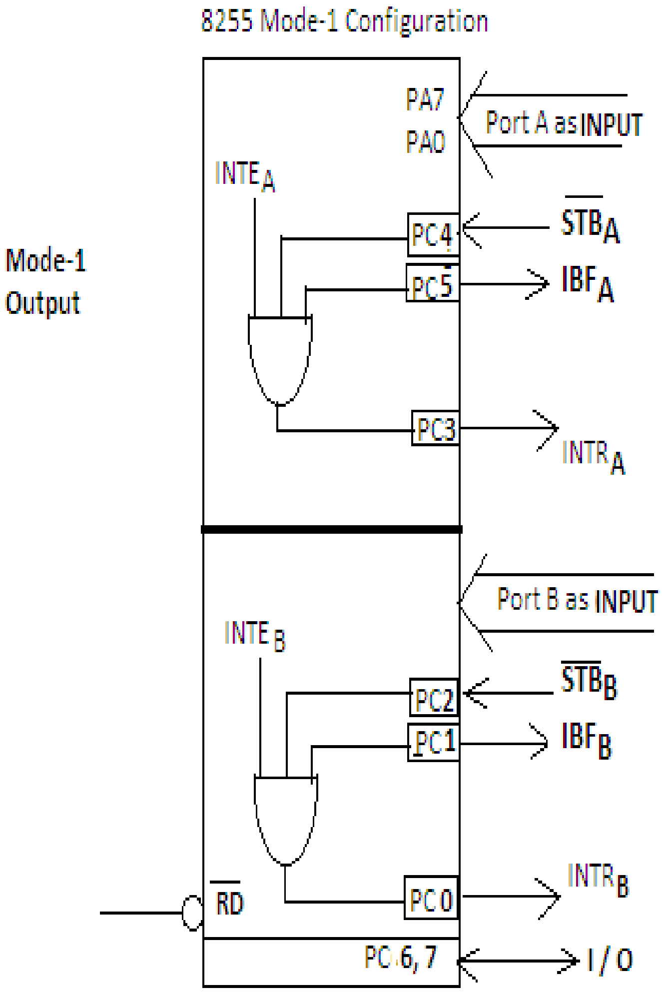
MODE 1 (Strobed Input/Output).

This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or ``handshaking'' signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these ``handshaking'' signals.

Mode 1 Basic functional Definitions:

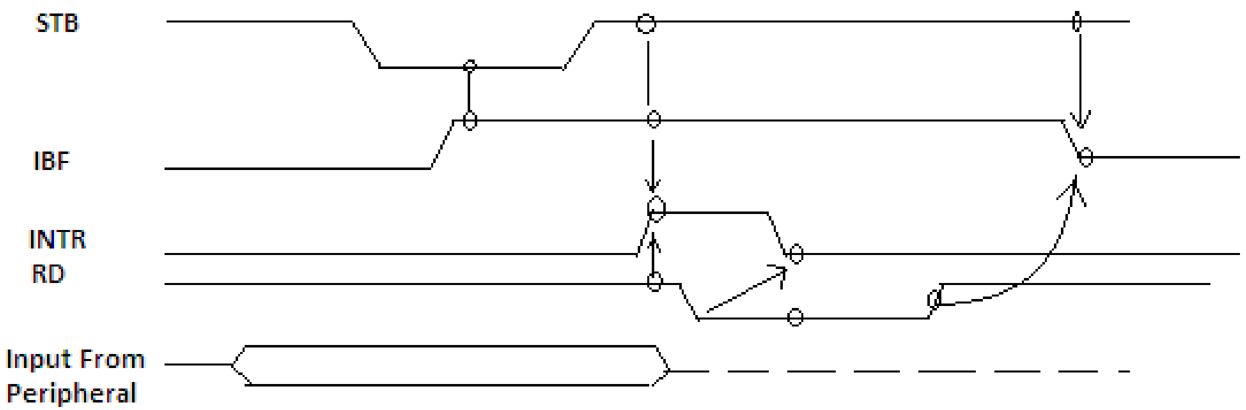
- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output
- Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

MODE 1 Input Operation



STB: This is an active low input to 8255A from the device to indicate that the device has transmitted a byte of data. 8255A in response generates an IBF and INTRA.
IBF: In response to STB, 8255A generates and transmits an active high signal 'IBF' as an acknowledgement for receipt of data. This is reset ('0') after the MPU reads the data.
INTE: 8255 has two internal flip-flops INTEA and INTEB. These are used to enable or disable the generation of INTR signal. These two FF are set/reset using the BSR mode. INTEA is enabled or disabled through PC4 and INTEB through PC2
INTR: For an interrupt driven I/O, an active high INTR signal is generated by 8255A that may be used to interrupt the MPU. As seen from figure, three signals STB', IBF and INTE all at logic '1' are input to an AND gate INTR is dropped when the data is read by the MPU at the falling edge of RD' signal

MODE 1 Timing (Input)



Waveform with strobed Input (with Handshake)

1. When a device is ready to send data using 8255 in mode-1 it sends the data on either port A or port B, the device also sends STB' signal to the 8255A to indicate that it has sent the data.
2. 8255A sends an IBFA or IBFB signal depending on the port used for data transmission as an acknowledgement.
3. If the INTEA of INTEB flip-flop is set the 8255A generates an INTR signal which may be used to interrupt the MPU in interrupt driven I/O operation.
4. In status check I/O MPU continuously checks IBFA and/or IBFB for it to become high. When it finds IBF(A or B) high, MPU reads the data from PA or PB and resets the IBF(A or B). If mechanism of IO is interrupt based, in that case when the MPU is interrupted it reads the data from the port and disables the interrupt.

MODE 1 Timing (Input)

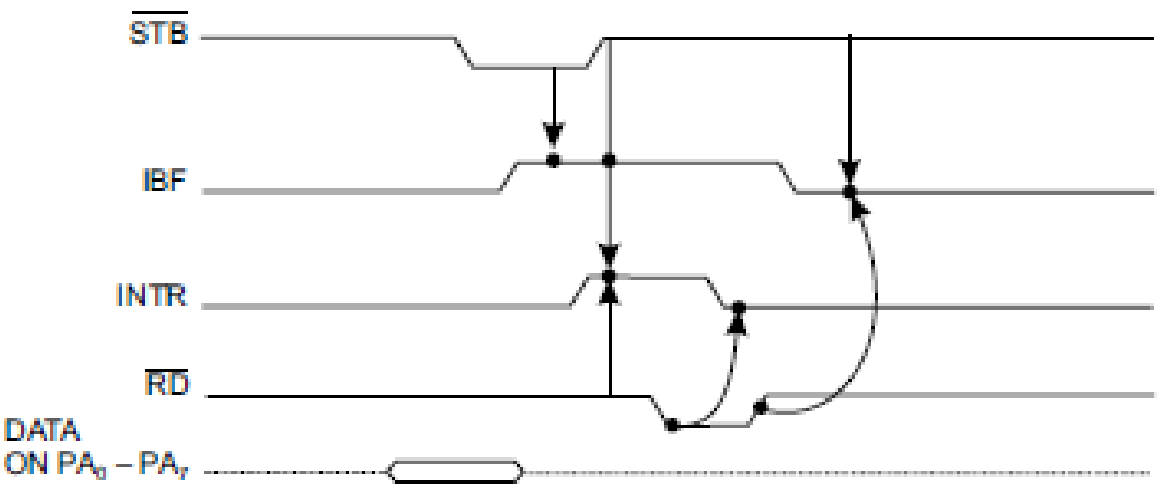
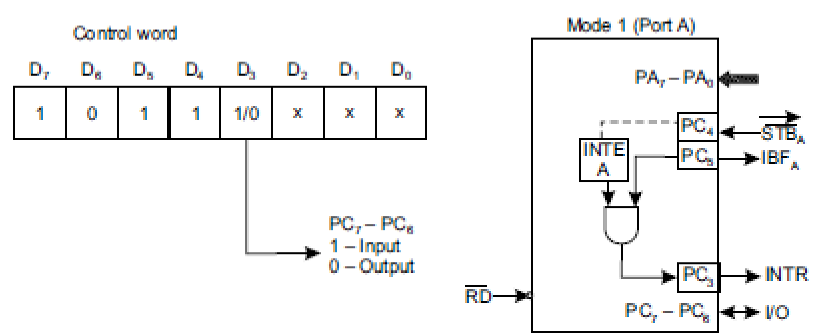
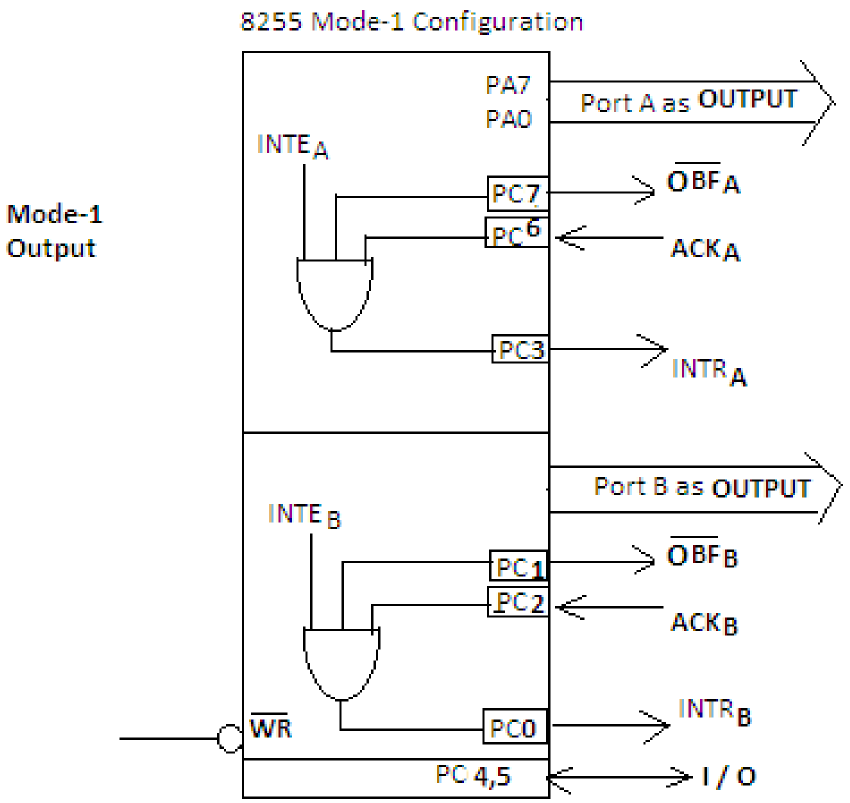


Fig. 9a.5: Port A in Mode 1 (Input) (*Source:* Intel Corporation)

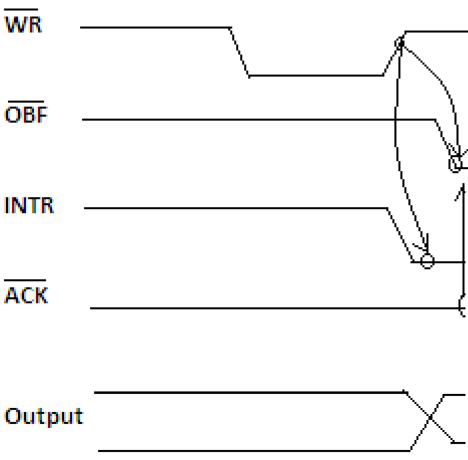


MODE 1 output Operation



INTE A :Controlled by bit set/reset of PC6.
INTE B: Controlled by bit set/reset of PC2.

MODE 1 Timing (output)



8255 Mode-1 C

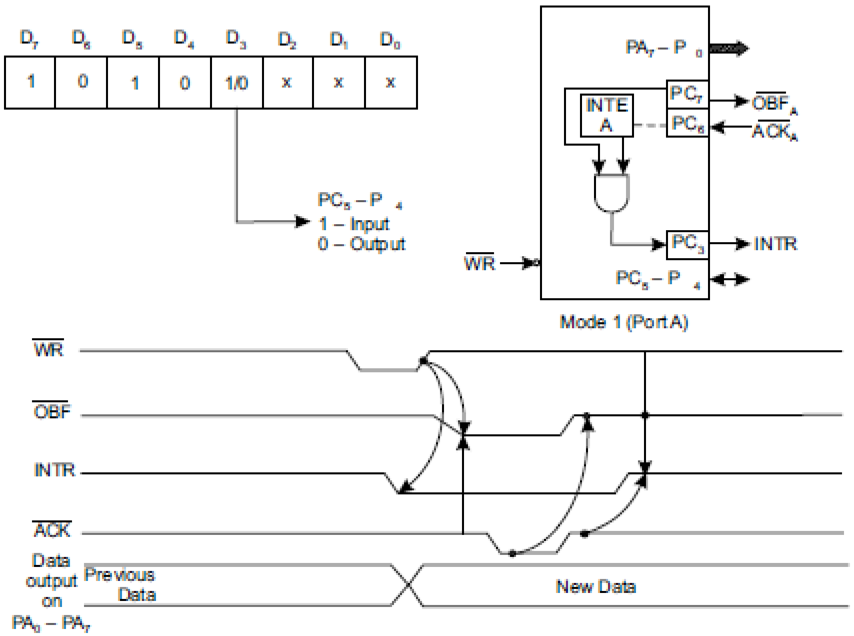


Fig. 9a.6: Port A in mode 1 (Output) (*Source:* Intel Corporation)

1. During a write operati… of port A or Port B. As the byte is written into the … output latch is full. The status of the oBF i.e. l… …able in latch.
2. Now the MPU can trar…

   I. If transmittin… …dicate to the device that there is data b… …the latch, it acknowledges it by sending a l… …data read by peripheral device. The MPU can now transmit new data byte.

   II. if transmitting using interrupt I/O when there is no data in output latch, then (OBF', ACK, INTE) are all high resulting in generation of INTR for the MPU for transmitting new data byte.

MODE 1 Timing (output)

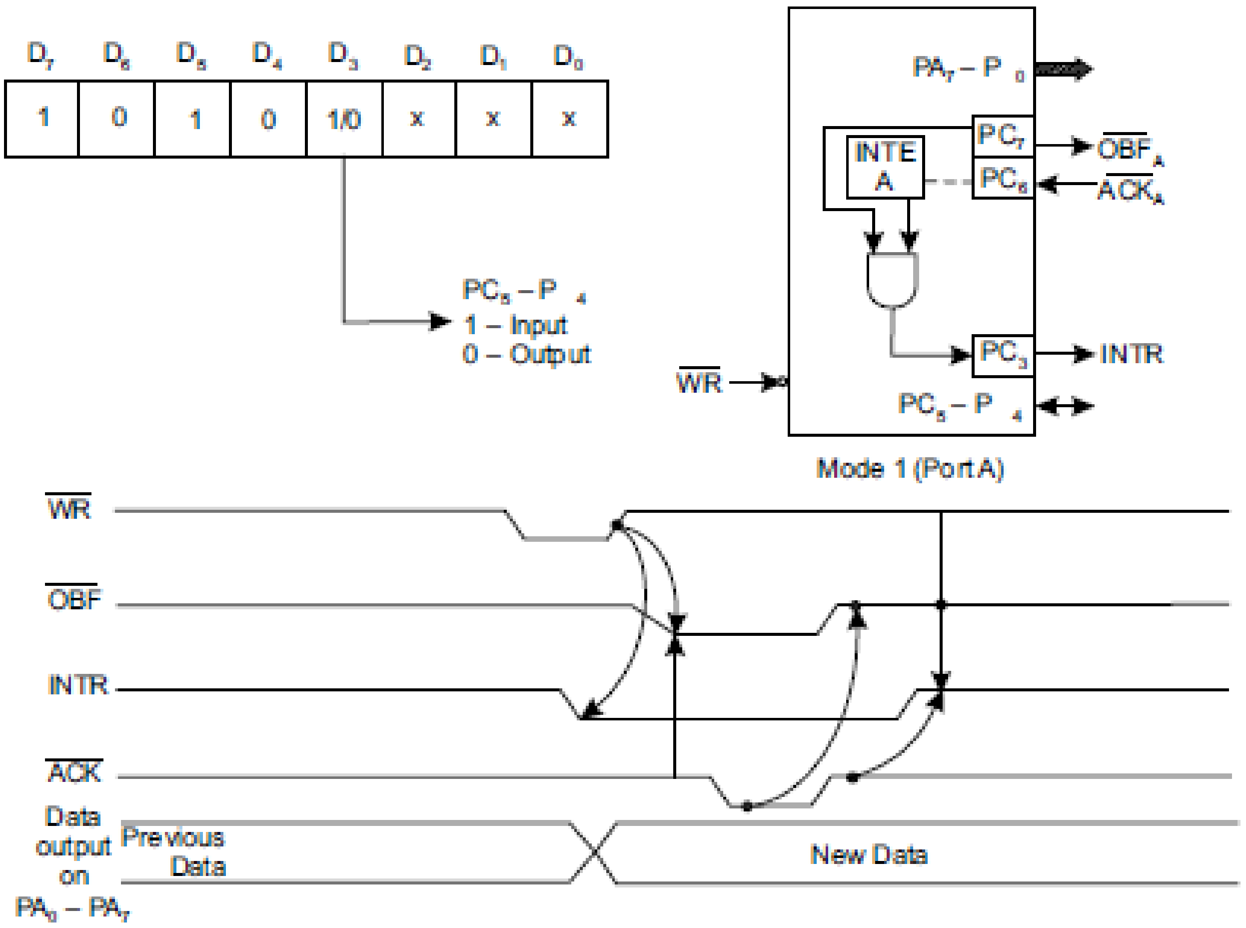


Fig. 9a.6: Port A in mode 1 (Output) (*Source:* Intel Corporation)

Sample mode 1

From the data segment:

```
MY_DATA     DB    "Hi. How are you?",CR,LF
            DB    "I am fine. How are you?",CR,LF,"$"
PA          EQU   300H                    ;port A address
PB          EQU   301H                    ;port B address
PC          EQU   302H                    ;port C address
CWP         EQU   303H                    ;control word port
LF          EQU   0AH                     ;line feed
CR          EQU   0DH                     ;carriage return
```

From the code segment:

```
            MOV   AL,10100000B            ;control word PA=out mode 1
            MOV   DX,CWP                  ;DX = 303 control word port
            OUT   DX,AL                   ;issue control word
            MOV   AL,00001101B            ;PC6=1 for INTEa
            OUT   DX,AL                   ;using BSR mode
            MOV   SI,OFFSET MY_DATA       ;SI = data address
    AGAIN:  MOV   AH,[SI]                 ;get a character
            CMP   AH,'$'                  ;is it the end?
            JZ    OVER                    ;if yes, exit
            MOV   DX,PC                   ;DX=302 port C address
    BACK:   IN    AL,DX                   ;get status byte from port C
            AND   AL,08                   ;is INTRa high?
            JZ    BACK                    ;if no, keep checking
            MOV   DX,PA                   ; if yes, make DX=300 data port
            MOV   AL,AH                   ;address and
            OUT   DX,AL                   ;send char to printer
            INC   SI                      ;increment the data pointer
            JMP   AGAIN                   ;keep doing it
    OVER:   ...                           ;go back to DOS
```

MODE 2 Operation

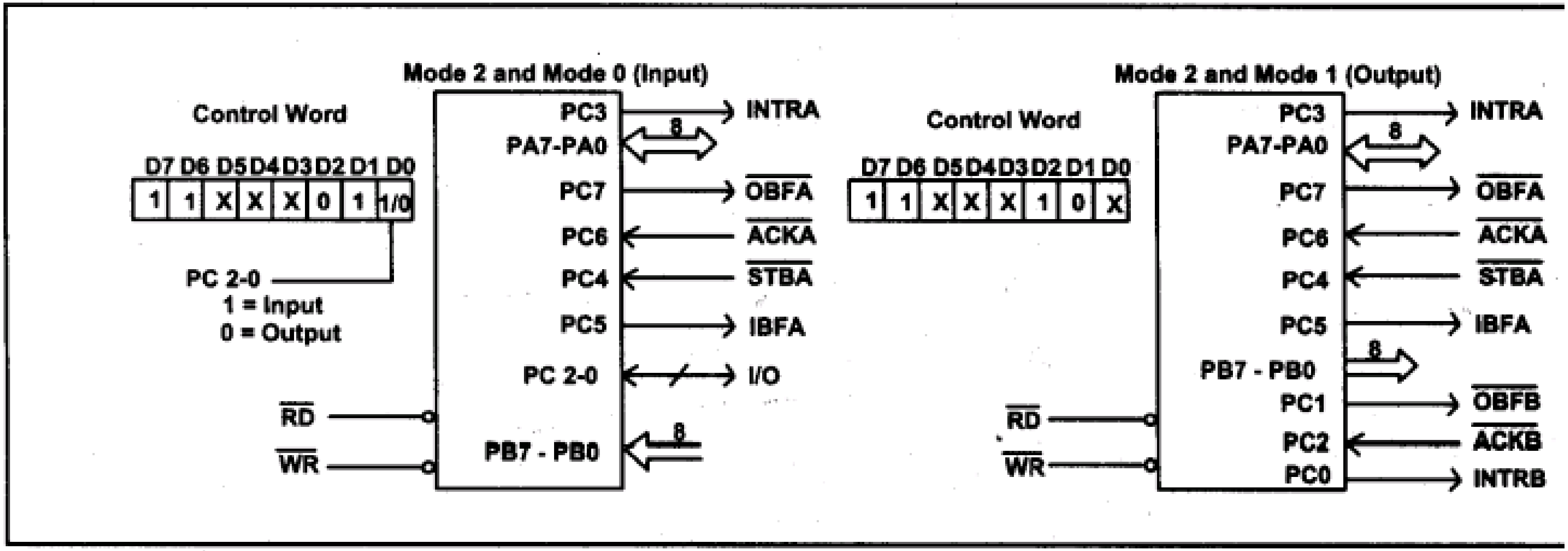
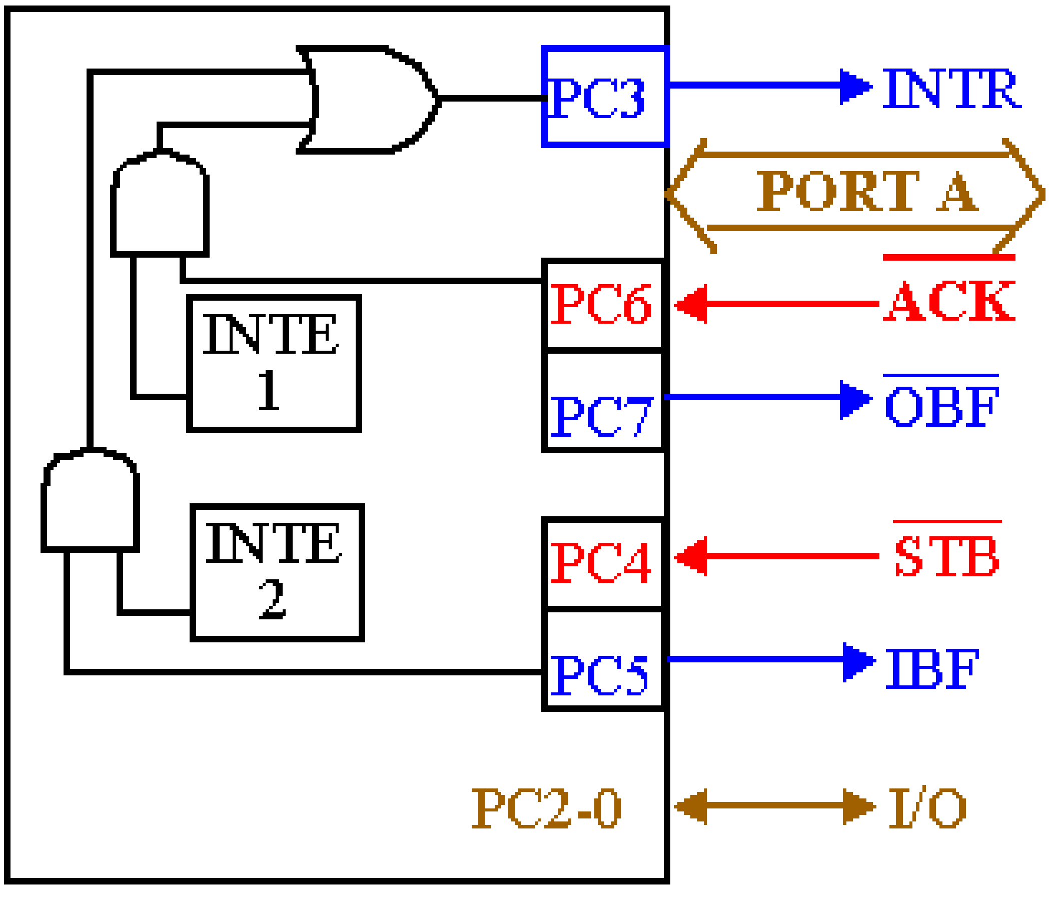


Figure 4-14. 8255A Modes 1 and 2 Input/Output Diagram
(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)

82C55: Mode 2 Bi-directional Operation



•Timing diagram is a combination of the Mode 1 Strobed Input and Mode 1 Strobed Output Timing diagrams.

Summary of Port Connection for 8255

		Mode 0		Mode 1		Mode 2
Port A		IN	OUT	IN	OUT	I/O
Port B		IN	OUT	IN	OUT	Not Used
	0			$INTR_B$	$INTR_B$	I/O
	1			IBF_B	$\overline{OBF}_B$	I/O
	2			$\overline{STB}_B$	$\overline{ACK}_B$	I/O
Port C	3	IN	OUT	$INTR_A$	$INTR_A$	INTR
	4			$\overline{STB}_A$	I/O	$\overline{STB}$
	5			IBF_B	I/O	IBF
	6			I/O	$\overline{ACK}_A$	$\overline{ACK}$
	7			I/O	$\overline{OBF}_B$	$\overline{OBF}$