INTERRUPTS

8086-80486

INTERRUPT SOURCES

Hardware Interrupt

- External input applied at non-maskable interrupt NMI
- External input applied at maskable interrupt INTR

Software Interrupt

- Execution of INT instruction
- Exception in program execution
- Trap

INTERRUPT VECTOR TABLE IVT –REAL MODES

Int Vector No.	Physical Address	Contains
INT 00_H	00000_H	IP0
	00002_H	CS0
INT 01_H	00004_H	IP1
	00006_H	CS1
…………….		
INT FF_H	$003FC_H$	IP255
	$003FE_H$	CS255

IDT – PROTECTED MODE

IDTR – stores physical base address of IDT and length in bytes of IDT

Each Entry (interrupt descriptor) – 8 bytes long

Size- 2KB (256x8 bytes)

Offset (A_{16} – A_{31})			
P	DPL	0 1 1 1 0	0 0 $_H$
Segment selector			
Offset (A_0 – A_{15})			

INTERRUPT VECTORS – 80X86

- ❑ Lowest 17 vectors are dedicated to specific interrupts
- ❑ Interrupts 18 to 31 are reserved by INTEL for complex processors /BIOS
- ❑ Upper 224 interrupt types (32 to 255) available to a user for hardware/software interrupts
- ❑ DOS uses 21_H (33)

INTERRUPTS - 8086

Interrupt Type zero – INT 0

- Divide by zero interrupt
- If the quotient is too large to fit into AL/AX
- Divide by zero interrupt invoked

Interrupt Type one – INT 1

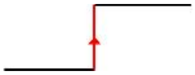
- Single step Interrupt
- If trap flag is set 80X86 will do a type 1 interrupt after every instruction execution

INTERRUPTS- 8086

INT 2

When 8086 receives a low to high transition on its

NMI input



Type 2 interrupt response cannot be disabled (masked) by any program instruction

INTERRUPTS - 8086

Break Point Interrupt – Type 3

- INT 3 instruction – to implement breakpoint routines
- The system execute instruction up to break point and then goes to break point routine
- Debugging

Overflow Interrupt – Type 4

- INTO
- Invoking an interrupt after overflow in an arithmetic operation
- If no overflow it will be a NOP instruction

INTERRUPTS -80286

INT 5

- BOUND
- BOUND AX, [SI]

INT 6

- Invalid opcode

INT 7

- Co-processor not available

INT 8

- Double Fault

INT 9

- Co-processor segment overrun
- Real mode co-processor offset address – $FFFF_H$

INTERRUPTS - 80286

INT A

- Invalid Task Segment

INT B

- Segment not present

INT C

- Stack Segment overrun
- SS not present in protected
- Size exceeded in protected or real

INTERRUPT - 80286

INT D

- GPL
  - Limit Exceeded
  - Privilege rules violated
  - Invalid descriptor segment type
  - Write to code sent
  - Read from execute only segment
  - Write to Read only segment

INTERRUPTS 80386

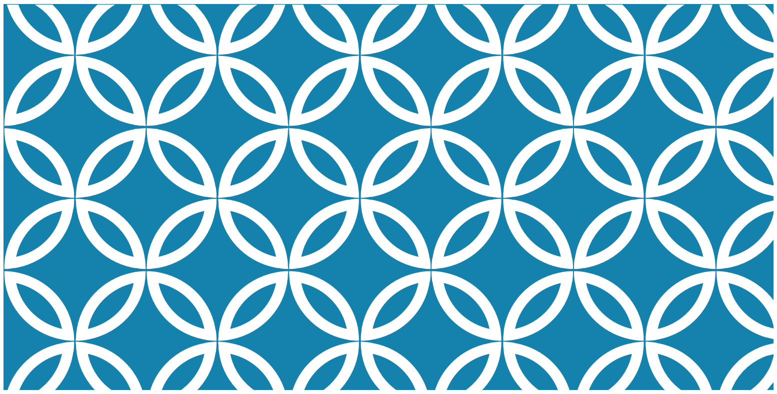
INT E

- Page Fault

INTERRUPT - 80486

INT 17

- Alignment Check



INTERRUPTS

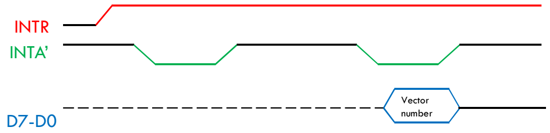
8086-80486

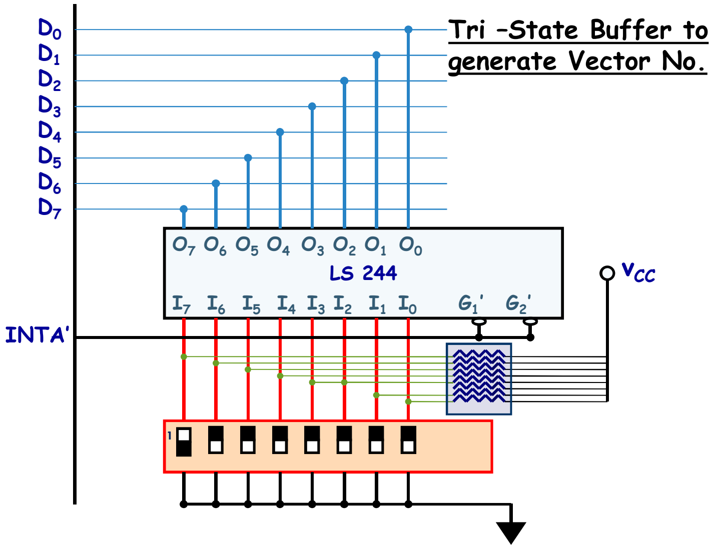
80X86 - INTR

- ❑Allows some external signal to interrupt execution of a program
- ❑INTR can be masked (disabled)
- ❑Clearing IF flag disables INTR
- ❑CLI - clears IF
- ❑STI - sets IF flag
- ❑80x86 when reset IF = 0
- ❑When 80x86 branches to ISR IF - 0
- ❑IRET/IRETD – IF -1

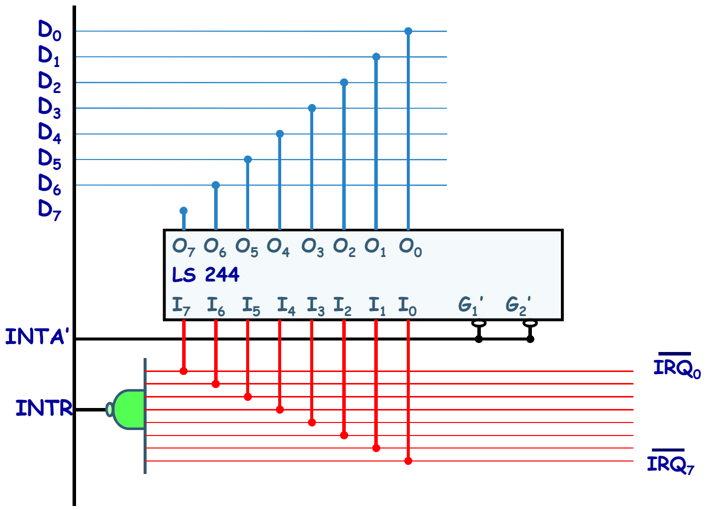
80X86 - INTERRUPTS

- ❑In response to INTR 80X86 expects a **vector number**
- ❑External hardware device required
- ❑It enters into INTA' machine cycle





Tri -State Buffer to generate Vector No.



IRQ'								Vector No.
7	6	5	4	3	2	1	0	
1	1	1	1	1	1	1	0	FE_H
1	1	1	1	1	1	0	1	FD_H
1	1	1	1	1	0	1	1	FB_H
1	1	1	1	0	1	1	1	$F7_H$
1	1	1	0	1	1	1	1	EF_H
1	1	0	1	1	1	1	1	DF_H
1	0	1	1	1	1	1	1	BF_H
0	1	1	1	1	1	1	1	$7F_H$

I/O INTERFACING

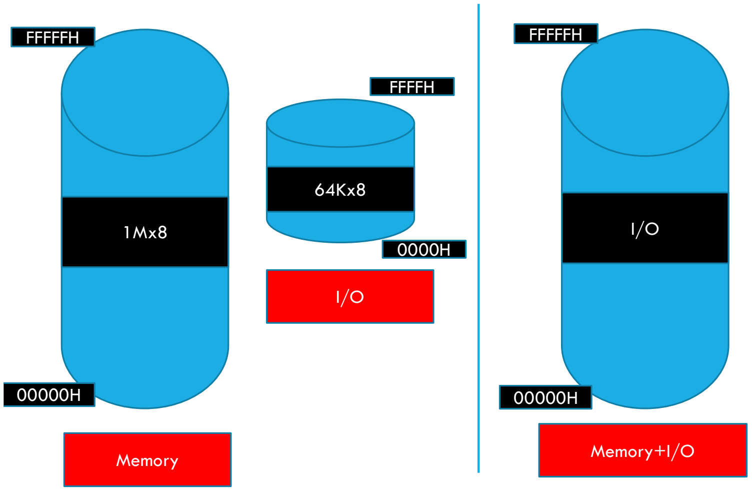
I/O INTERFACING

Isolated I/O (or I/O mapped I/O)

- Memory address space is different from I/O address space
- Signals used are IOR' and IOW' for peripherals and MEMR' and MEMW' for memory
- Instructions used are IN and OUT

Memory mapped I/O

- IN and OUT instructions are not used.
- Signals used are MEMR' and MEMW' both for memory and peripherals
- Memory address space is same as I/O address space



I/O INSTRUCTIONS

IN and OUT instructions

INS and OUTS instructions

IN AL,p8

IN AX,p8

IN EAX,p8

IN AL,DX

IN AX,DX

IN EAX,DX

OUT p8,AL

OUT p8,AX

OUT p8,EAX

OUT DX,AL

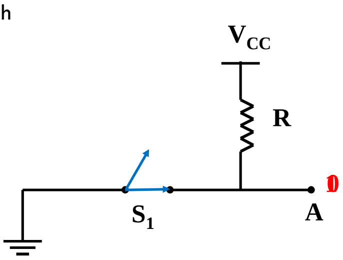
OUT DX,AX

OUT DX,EAX

I/O INTERFACING

Input Device

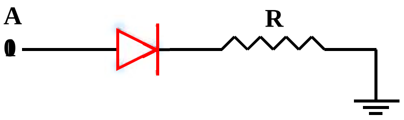
- E.g. Switch



I/O INTERFACING

Output Device

- E.g. LED



WHY BUFFERS ?

Input devices must be isolated from the global data bus

Else unwanted data (garbage) may be transferred on to the data bus

Tri-state buffers – provide isolation as well as strengthen the signal

Why not use latches and buffers with Memory Interfacing ?
Memory has such latches and buffers internally present

I/O DESIGN IN 8086

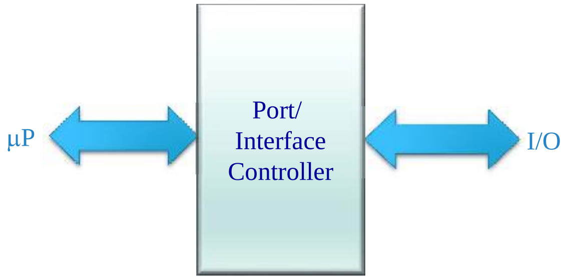
In any µP-based system when data is sent out by µP, the data on the data-bus must be latched by the receiver/output device

Memories have internal latches – store data

Latching system must be designed for ports

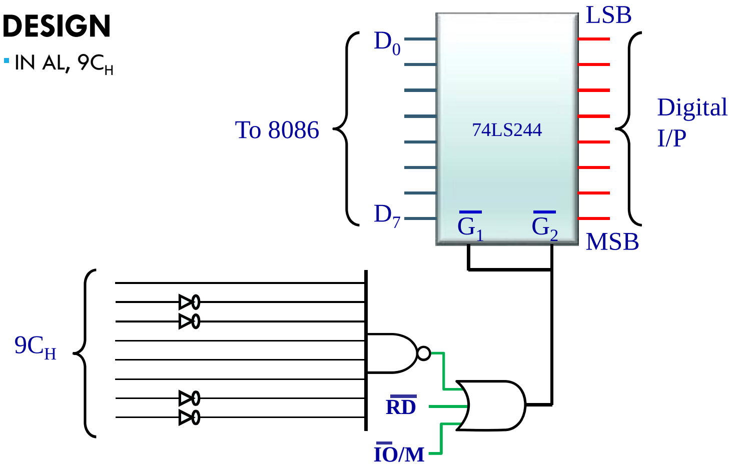
Data provided by the µP is available only for short period of time (50-1000ns) data must be latched else it will be lost

Similarly, when data comes in from a port/memory, data must be input through a tri-state buffer



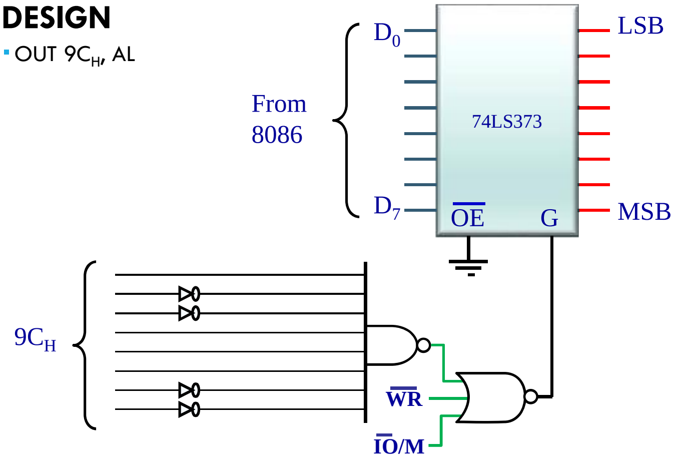
DESIGN

- IN AL, $9C_H$



DESIGN

- OUT $9C_H$, AL



- Interfacing *input devices* like switches require *buffers.*
- Interfacing *output devices* like LEDs require *latches*.
- Programmable Peripheral Interface (PPI) device provides these features.

I/O INTERFACING

8255

8255 – PROGRAMMABLE PERIPHERAL INTERFACE (PPI)

Intel has developed several peripheral control chips for 80x86 family

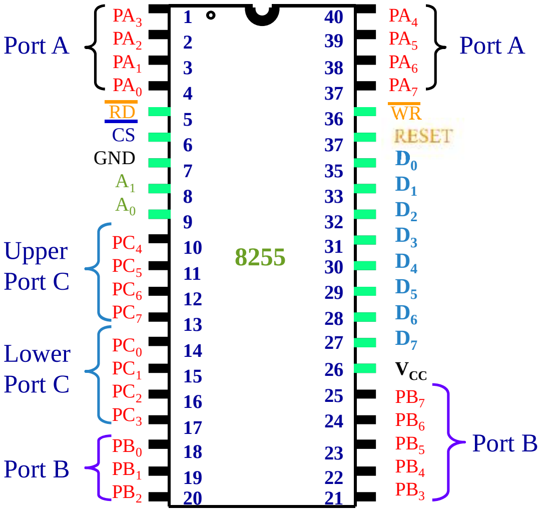
Intent – provide complete I/O interface to x86 chip

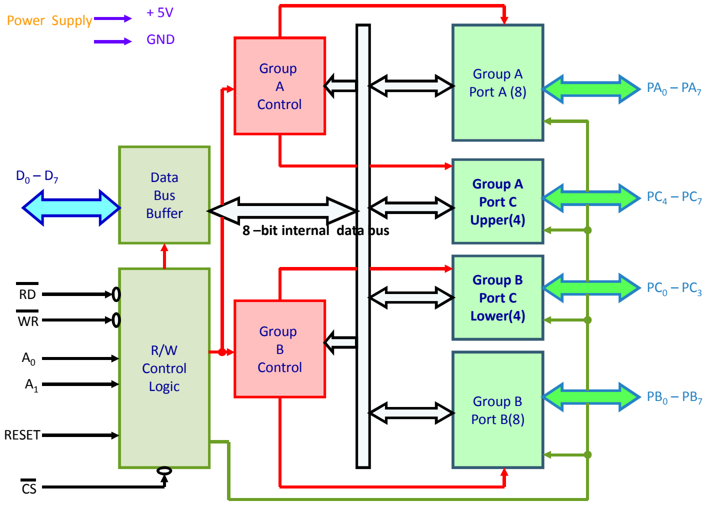
8255 PPI

- PPI provides 3, 8-bit I/O ports in one package
- Chip can be directly interfaced to the data bus of 8086

Other Peripheral Devices

- 8253/8254 – Programmable Interval Timer (PIT)
- 8259 – Programmable Interrupt Controller (PIC)
- 8237 – Direct memory Access Controller (DMAC)





8255A Internal

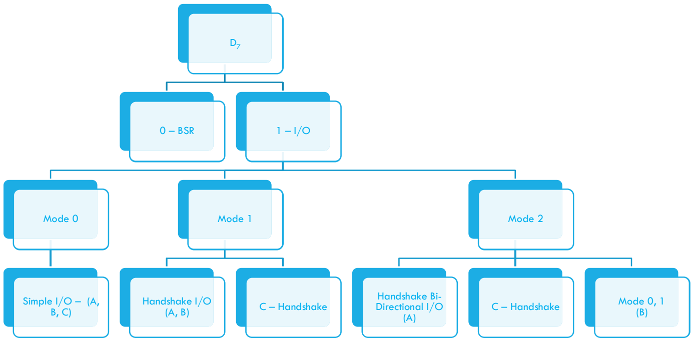
CS’	A_1	A_0	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255 Not Selected

Selecting Port / Programming 8255

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	Port A Mode		**Port A**	**Port C Upper**	**Port B Mode**	**Port**	**Port C Lower**
Always 1 for I/O Mode	0 0 - Mode 0 0 1 - Mode 1 1 x – Mode 2		1 - I/P 0 - O/P	1 - I/P 0 - O/P	0-Mode0 1-Mode1	1 - I/P 0 -O/P	1 - I/P 0 - O/P
	Group A				**Group B**		

Control Word Format for I/O Mode

MODES OF OPERATION – 8255



D_7

0 – BSR

1 – I/O

Mode 0

Mode 1

Mode 2

Simple I/O – (A, B, C)

Handshake I/O (A, B)

C – Handshake

Handshake Bi-Directional I/O (A)

C – Handshake

Mode 0, 1 (B)

I/O INTERFACING

BSR Mode Example

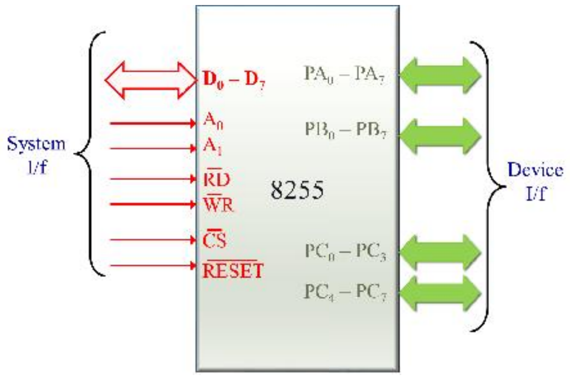
BSR MODE OF 8255 – PORT C

Example: Connect 3 LEDs to Port C. Blink one LED after another at regular intervals of 1ms

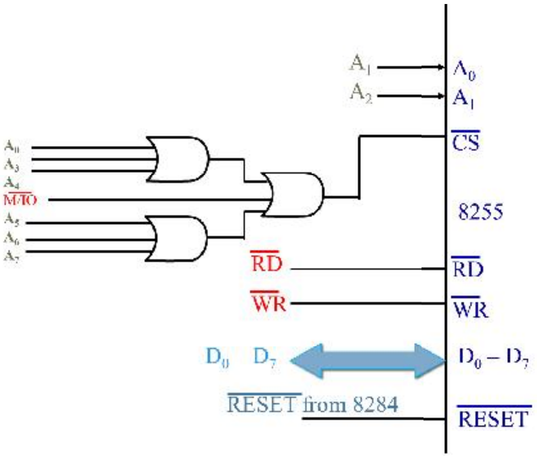
8255- Base address 00_H

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0 -BSR	x	x	x	Bit_2	Bit_1	Bit_0	Bit Set/Reset
	Don't Care Condition						1 - Set 0 - Reset

PC	0	1	2	3	4	5	6	7
B_0	0	1	0	1	0	1	0	1
B_1	0	0	1	1	0	0	1	1
B_2	0	0	0	0	1	1	1	1



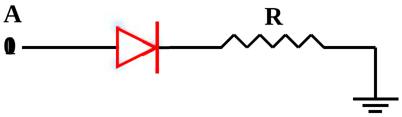
I/F – 8255



Interface to the Processor

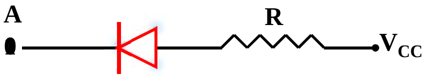
OUTPUT DEVICE

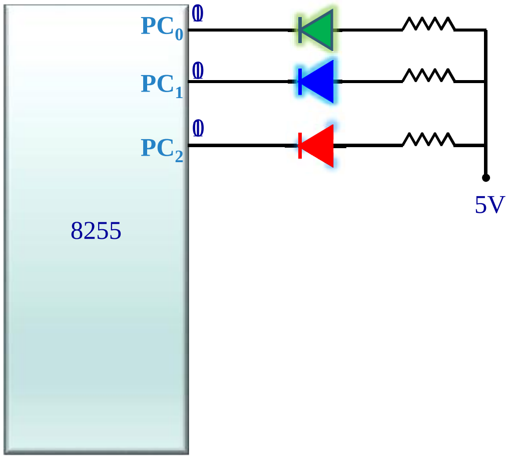
- E.g. LED



OUTPUT DEVICE

- E.g. LED

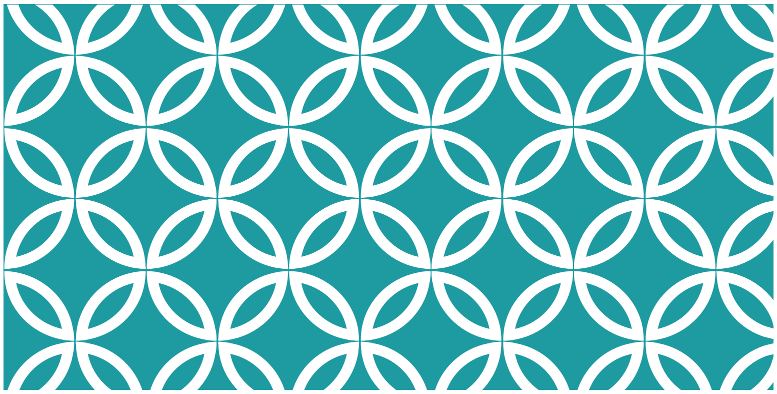




Interface to the I/O Devices

```
creg    equ 06h
        moval,80h
        out     creg,al
x1:moval,00
        out     creg, al
        moval,03
        out     creg, al
        moval,05
        out     creg, al
calldelay_1ms
        mov     al,01
        out        creg, al
        mov     al,02
```

```
        out     creg, al
        mov     al,05
        out     creg, al
call    delay_1ms
      mov     al,01
        out     creg, al
        mov     al,03
        out     creg, al
        mov     al,04
        out     creg, al
call    delay_1ms
        jmp     x1
```



I/O INTERFACING

Basic I/O example

MODE 0 : SIMPLE INPUT/OUTPUT

O/P

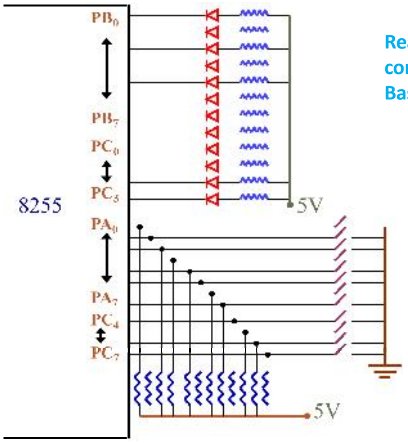
- Are Latched
- Any of the Ports – A, B & C can be used

I/P

- Buffered
- Any of the Ports – A, B & C can be used

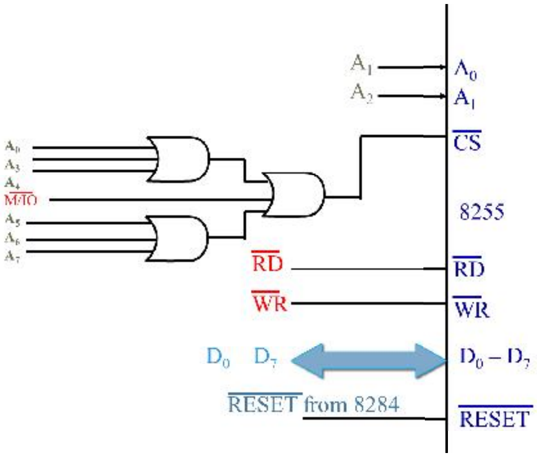
Interrupts/Handshake

- Not Possible



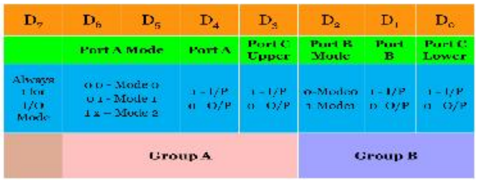
Read 12 switches and display switch condition on 12 LEDs with 8255H and Base Address – 00H

Interface to the 8255



Interface to the Processor

```
creg      equ   06h
porta     equ   00h
portb     equ   02h
portc     equ   04h
mov    al,10011000b
out    creg,al
in     al,porta
out    portb,al
in     al,portc
and    al,0f0h
mov    cl,04h
ror    al,cl
out    portc,al
```



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	Port A Mode		Port A	Port C Upper	Port B Mode	Port B	Port C Lower
Always 1 for I/O Mode	0 0 - Mode 0 0 1 - Mode 1 1 x – Mode 2		1 – I/P 0 O/P	1 – I/P 0 O/P	0-Mode0 1 Mode1	1 - I/P 0 O/P	1 – I/P 0 O/P
	Group A				Group B		

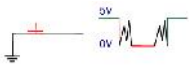
I/O INTERFACING KEYPAD

MATRIX KEYPAD INTERFACING

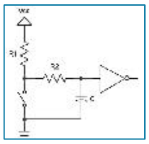
- ❑In most keyboards, key switches are connected in a matrix of rows and columns. Getting meaningful data from keyboard, requires following steps:
- ❑Detect a key press
- ❑De-bounce the key press
- ❑Encode the key press (produce a standard code for the key press)

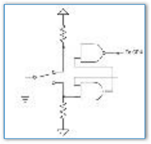
KEY DE-BOUNCE

- When a mechanical key is pressed or released- the metallic contacts bounce before they make steady state contact
- Bouncing is noise and should not be treated as i/p

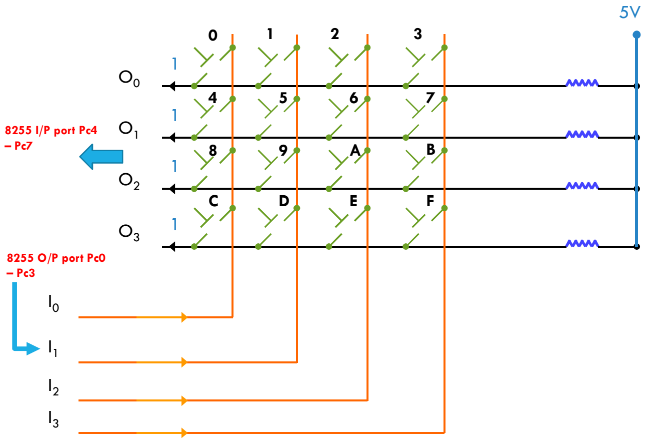


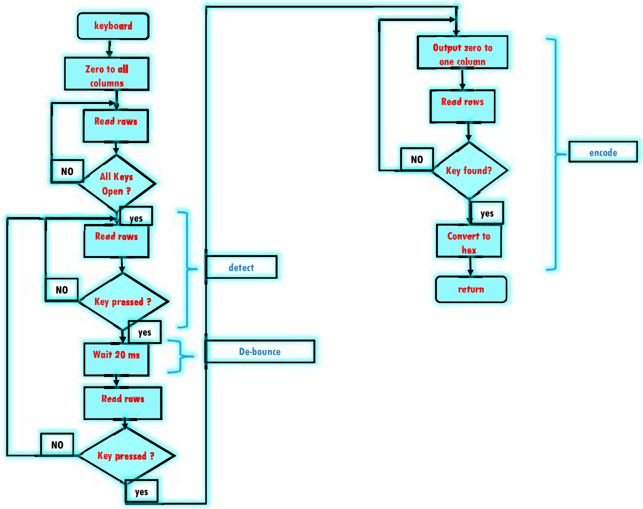
- De-bounce
  - s/w – using software delay of 10-20 ms
  - h/w –dedicated hardware device



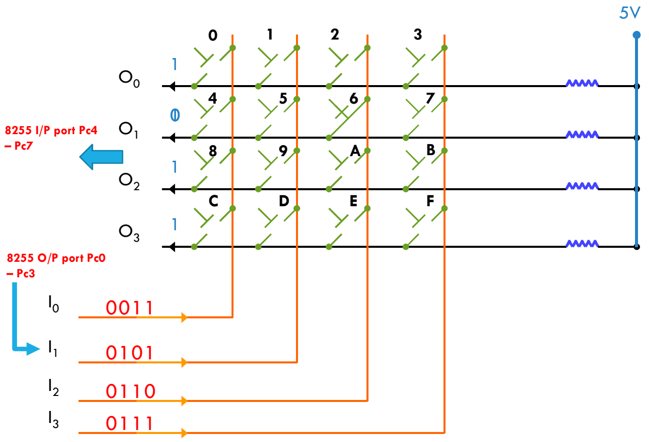


Mode 0 – Matrix Keypad

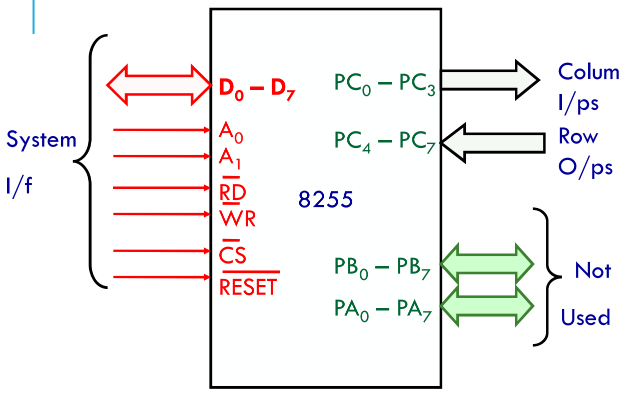


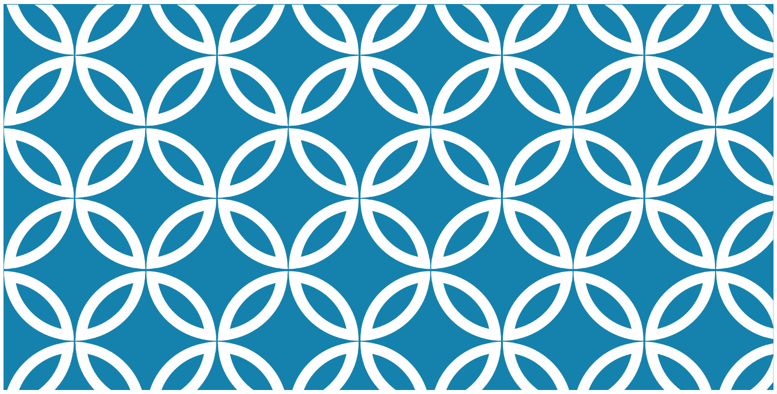


E.g. Mode 0 – Keypad I/f



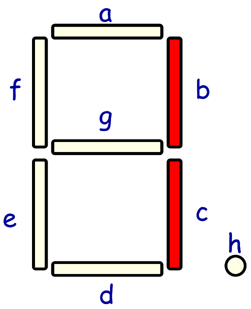
Key	O_3	O_2	O_1	O_0	I_3	I_2	I_1	I_0	HEX
0	1	1	1	0	1	1	1	0	EE
1	1	1	1	0	1	1	0	1	ED
2	1	1	1	0	1	0	1	1	EB
3	1	1	1	0	0	1	1	1	E7
4	1	1	0	1	1	1	1	0	DE
5	1	1	0	1	1	1	0	1	DD
6	1	1	0	1	1	0	1	1	DB
7	1	1	0	1	0	1	1	1	D7
8	1	0	1	1	1	1	1	0	BE
9	1	0	1	1	1	1	0	1	BD
A	1	0	1	1	1	0	1	1	BB
B	1	0	1	1	0	1	1	1	B7
C	0	1	1	1	1	1	1	0	7E
D	0	1	1	1	1	1	0	1	7D
E	0	1	1	1	1	0	1	1	7B
F	0	1	1	1	0	1	1	1	77

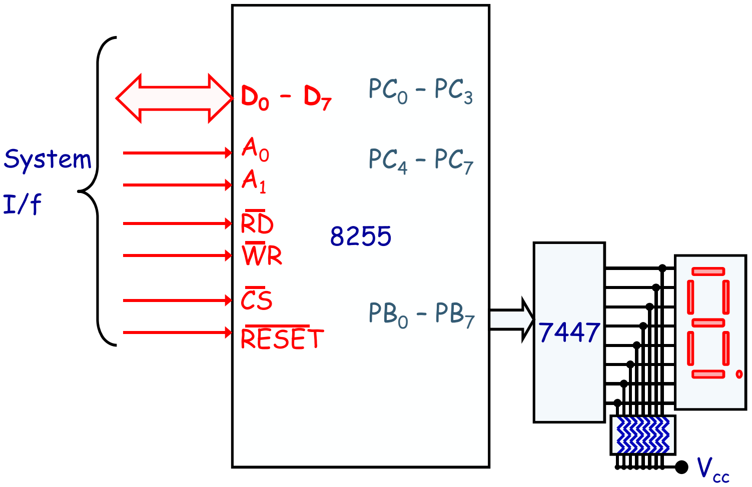




I/O INTERFACING

Display Interfacing

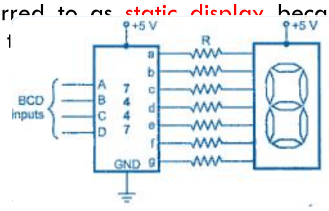




STATIC DISPLAY

- For a common anode display, a segment is turned on by applying a logic low to it. The 7447 converts a BCD code to its inputs to the pattern of lows required to display the number represented by the BCD code.

- This is referred to as static display because current is being passed t

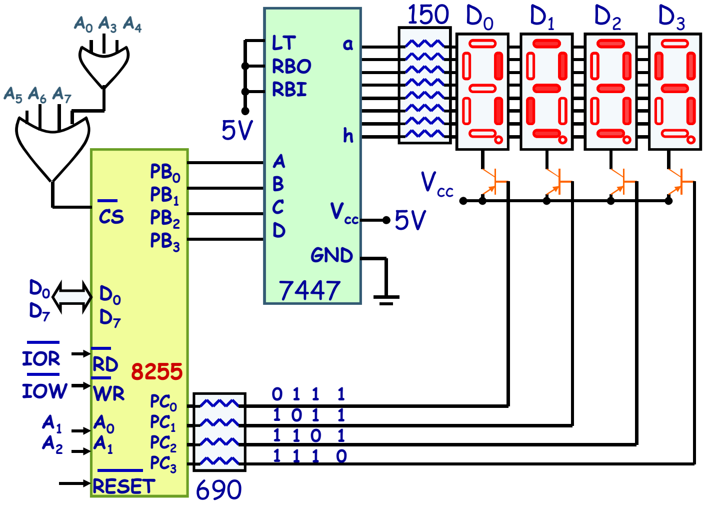


Drawbacks:

- When using the scheme to drive multiple digits.
  - Power consumption – say, 8 devices displaying digit 8 at the same time. Seven segments are lit in one device.
  - Current drawn = 7 segments x 8 devices x 20 mA = 1.12 A
- Each seven segment display device, requires a separate 7447 decoder and each decoder draws a current of 13 mA.
- Current required by decoder and LED displays is huge.

Solution:

- **Software-multiplexed or Scanned display (uses only one 7447 decoder to drive multiple devices)**



Input a data from a keyboard

Input data is a valid one digit Hex number

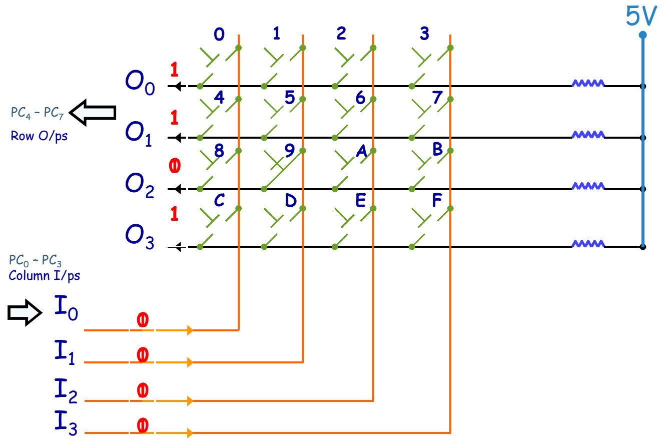
Use a lookup table to convert it into seven segment code

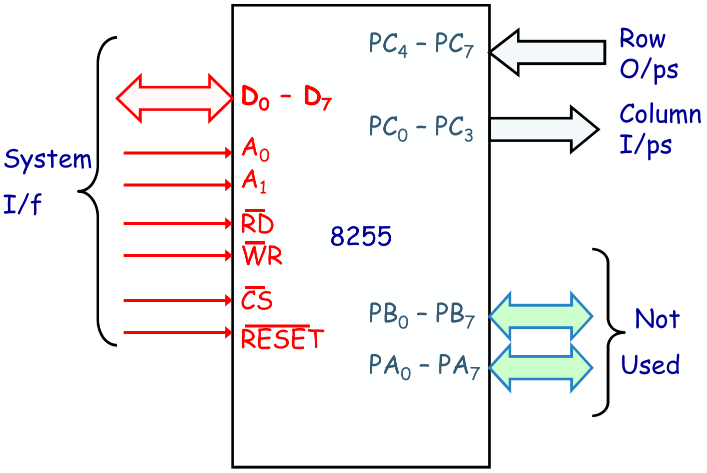
Output data on seven-segment display

I/O INTERFACING

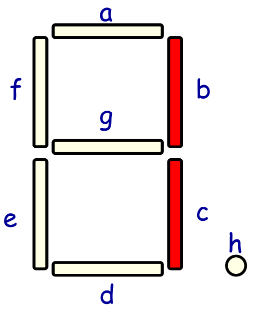
Example: Keypad Interface

E.g. Mode 0 – Keypad I/f



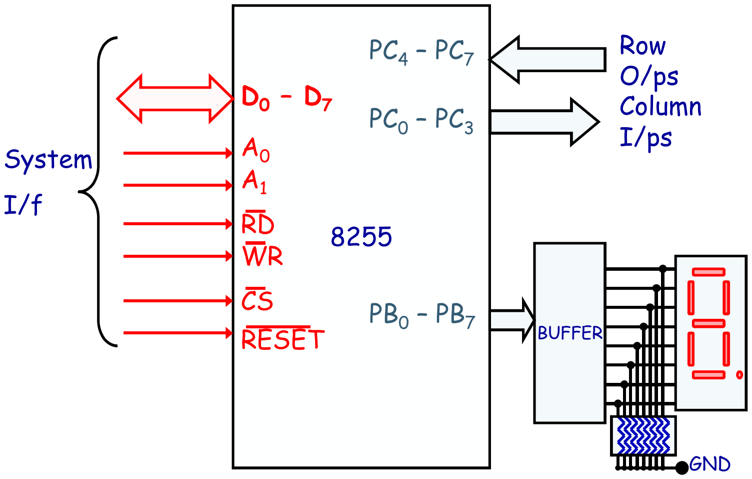


KEY	O_3	O_2	O_1	O_0	I_3	I_2	I_1	I_0	HEX
0	1	1	1	0	1	1	1	0	EE
1	1	1	1	0	1	1	0	1	ED
2	1	1	1	0	1	0	1	1	EB
3	1	1	1	0	0	1	1	1	E7
4	1	1	0	1	1	1	1	0	DE
5	1	1	0	1	1	1	0	1	DD
6	1	1	0	1	1	0	1	1	DB
7	1	1	0	1	0	1	1	1	D7
8	1	0	1	1	1	1	1	0	BE
9	1	0	1	1	1	1	0	1	BD
A	1	0	1	1	1	0	1	1	BB
B	1	0	1	1	0	1	1	1	B7
C	0	1	1	1	1	1	1	0	7E
D	0	1	1	1	1	1	0	1	7D
E	0	1	1	1	1	0	1	1	7B
F	0	1	1	1	0	1	1	1	77



COMMON CATHODE DISPLAY CODES

Display	h	g	f	e	d	c	b	a	HEX
0	1	1	1	0	1	1	1	0	3F
1	1	1	1	0	1	1	0	1	06
2	1	1	1	0	1	0	1	1	5B
3	1	1	1	0	0	1	1	1	4F
4	1	1	0	1	1	1	1	0	66
5	1	1	0	1	1	1	0	1	6D
6	1	1	0	1	1	0	1	1	7D
7	1	1	0	1	0	1	1	1	07
8	1	0	1	1	1	1	1	0	7F
9	1	0	1	1	1	1	0	1	67
A	1	0	1	1	1	0	1	1	77
B	1	0	1	1	0	1	1	1	7C
C	0	1	1	1	1	1	1	0	39
D	0	1	1	1	1	1	0	1	5E
E	0	1	1	1	1	0	1	1	79
F	0	1	1	1	0	1	1	1	71



```
.Model Tiny

.DATA
TABLE_D     DB          3FH,06H,5BH,4FH,66H,6DH,7DH,

            DB          07H,7FH,67H,77H,7CH,39H,5EH,

            DB          79H,71H


TABLE_K     DB          EEH,, EDH, EBH, E7H, DEH, DDH,

            DB          DBH, D7H, BEH, BDH, BBH, B7H

            DB          7EH,  7DH, 7BH, 77H,
```

```
.CODE
.STARTUP
            MOV         AL,10011000B    } Initialize
            OUT         06H,AL          } 8255
X0:         MOV         AL,00H          }
            OUT         04H,AL          }
X1:         IN          AL, 04H         } Check for
            AND         AL,F0H          } key release
            CMP         AL,F0H          }
            JNZ         X1              }
            CALL        DELAY_20MS    ⟶Debounce
```

```
            MOV         AL,00H          }
            OUT         04H ,AL         }
X2:         IN          AL, 04H         } Check for
            AND         AL,F0H          }
            CMP         AL,F0H          } Key press
            JZ          X2              }
            CALL        DELAY_20MS
            MOV         AL,00H          }
            OUT         04H ,AL         }
            IN          AL, 04H         } Check for
            AND         AL,F0H          }
            CMP         AL,F0H          } Key press
            JZ          X2              }
```

```
            MOV         AL, 0EH         }
            MOV         BL,AL           }
            OUT         04H,AL          } Check for
            IN          AL,04H          }
            AND         AL,F0H          } Key press
            CMP         AL,F0H          }
            JNZ         X3              } Column1
            MOV         AL, 0DH         }
            MOV         BL,AL           }
            OUT         04H ,AL         } Check for
            IN          AL,04H          }
            AND         AL,F0H          } Key press
            CMP         AL,F0H          }
            JNZ         X3              } Column2
```

```
MOV     AL, 0BH
MOV     BL,AL
OUT     04H,AL
IN      AL,04H
AND     AL,F0H
CMP     AL,F0H
JNZ     X3
MOV     AL, 07H
MOV     BL,AL
OUT     04H,AL
IN      AL,04H
AND     AL,F0H
CMP     AL,F0H
JZ      X2
```

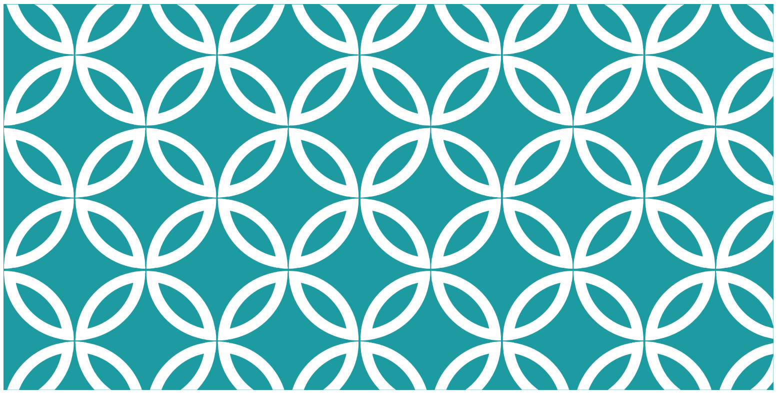
Check for Key press Column3

Check for Key press Column4

```
X3:     OR      AL,BL
        MOV     CX,0FH
        MOV     DI,00H
X4:     CMP     AL,TABLE_K[DI]
        JZ      X5
        INC     DI
        LOOP    X4
X5:     MOV     AX,DI
        LEA     BX, TABLE_D
        XLAT
        OUT     02H,AL
        JMP     X0
.EXIT
END
```

Decode key

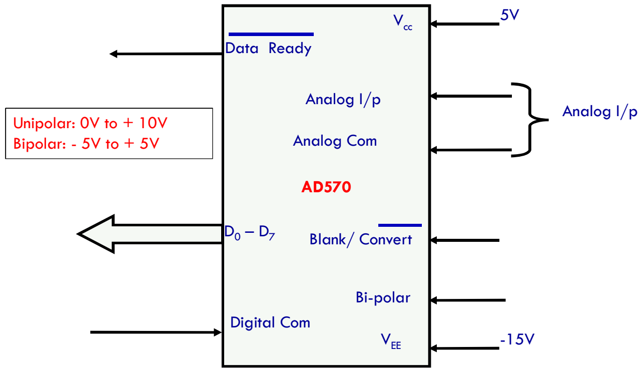
Display

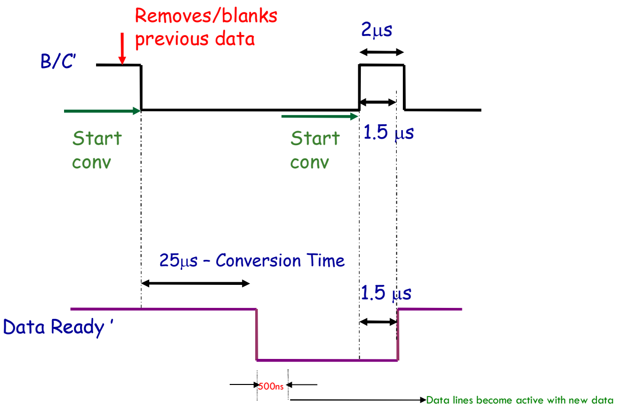


I/O Interfacing

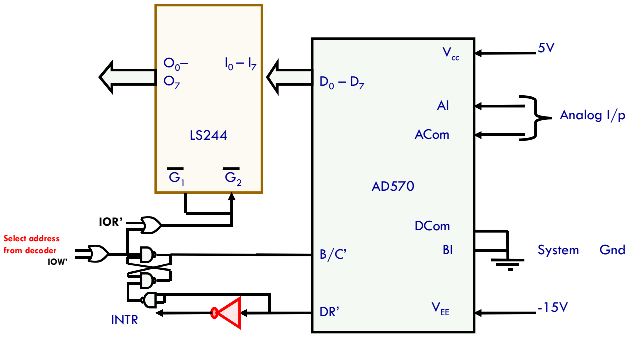
ADC

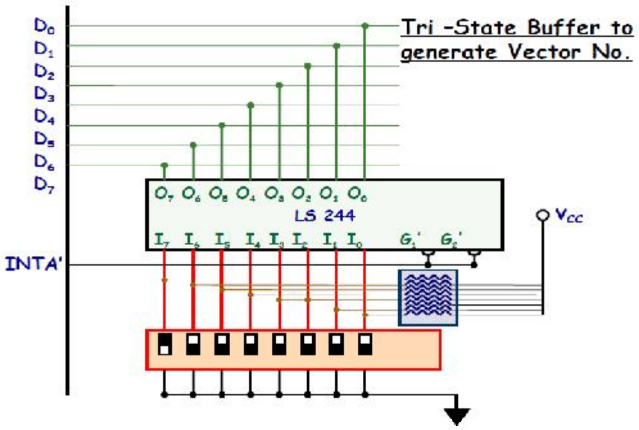
Interfacing A/D Converter



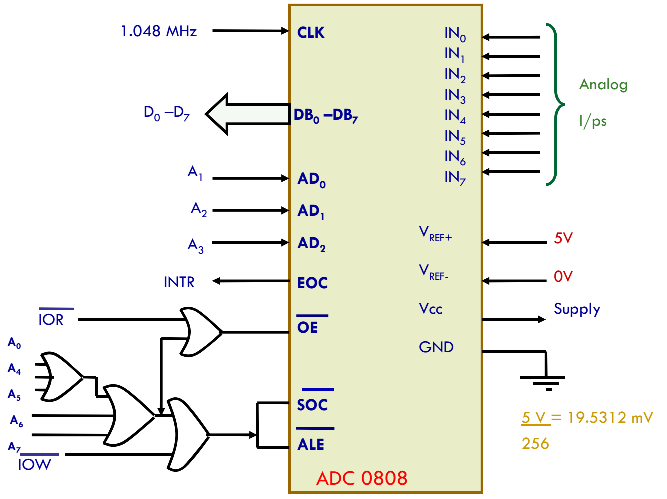


$$\frac{10\ V}{256} = 39.0625\ mV$$

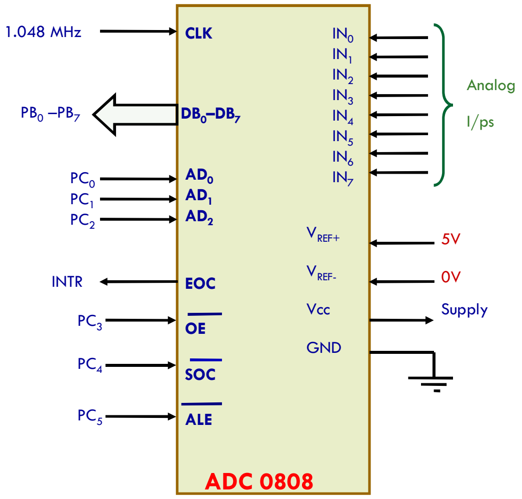


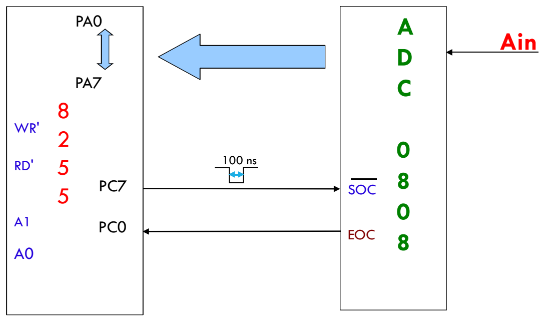


Tri -State Buffer to generate Vector No.



$\frac{5\ V}{256} = 19.5312$ mV





Base Address of 8255 =80H

I/O Control Word: 10010001 = 91H
BSR Control Word: 00001110 = 0EH ;for resetting PC7
00001111 = 0FH ;for setting PC7

```
MOV AL,91H
OUT 86H,AL
MOV AL,0EH
OUT 86H,AL
CALL DELAY
MOV AL,0FH
OUT 86H,AL
BACK: IN AL,84H
      AND AL,01H   ; TO CHECK IF EOC = 1
       JZ BACK
       IN AL,80H
```

Programmabl
e Interval
Timer
8253/8254

Counting/ Generation of Timing Signals

Software – delay routines

Adv – Flexibility

Disadv – Less precision

Hardware – 555/ RC

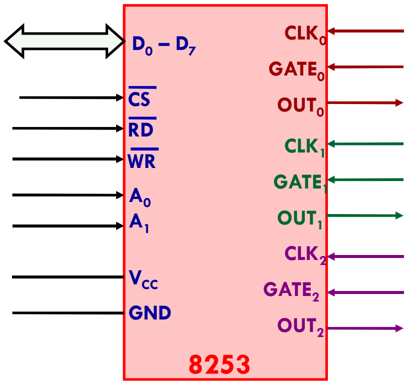
Adv – Precision

Disadv – Not Flexible

S/w Controlled hardware Timer- 8253/8254

FEATURES OF 8253/8254

- Three 16 – bit counters
- Max frequency
  - 2.6MHz 8253
  - 8MHz 8254
  - 10MHz 8254-2
- Down Counters
- Count value to be loaded in counter can be a Binary/BCD Number
- Can operate in one of 6 possible Modes



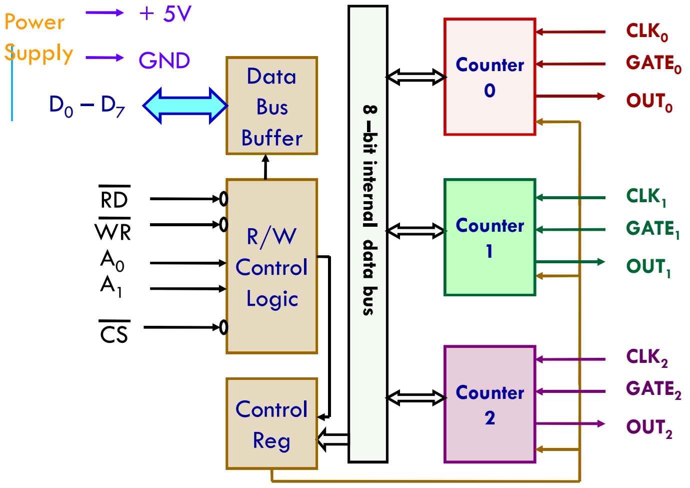
Pin Out of 8253

8253 – TIMING

Counter Output freq = $\frac{\text{Input clk freq}}{\text{Count value}}$

Count – 16-bit

BCD/ Binary



8253 Internal

CS'	A_1	A_0	Selected
0	0	0	Counter 0
0	0	1	Counter 1
0	1	0	Counter 2
0	1	1	Control Register
1	X	X	8253/8254 Not Selected

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD
Selects Counter		Read/Write Control		Timer Mode			0 – binary 0000_h $FFFF_h$ 1 – BCD 0000 9999
00	Counter 0	00	Latch Counter	000	Interrupt on T/C		
01	Counter 1	01	R/W LSB	001	h/w re-Triggerable one shot		
10	Counter 2	10	R/W MSB	010	rate generator		
11	Read Back Command	11	R/W LSB followed by MSB	011	Square wave generator		
				1x0	s/w triggered strobe		
				1x1	h/w triggered strobe		

Before you can use……

1. Initialize the mode of every counter planned to be used
2. This is done by sending individual command words for every counter
3. These CWs are sent at A_1A_0 =11
4. Send counts to the counters
5. This is done at counter addresses
6. Enable gates for counting to start

Ex:

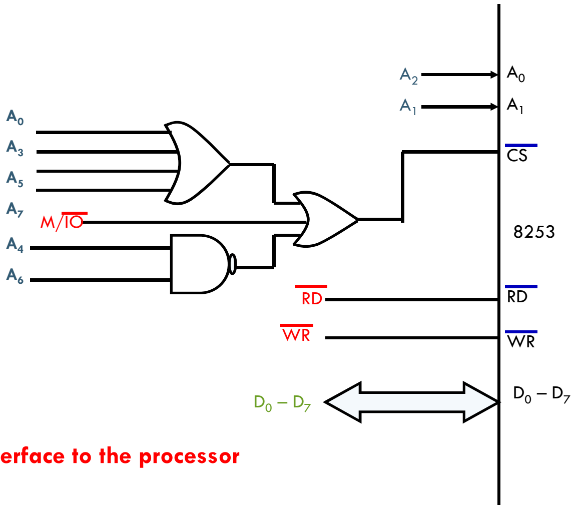
8253 interfaced starting at 50_H

C_0 used in mode 1, MSB+LSB, binary

To be loaded with 4000_H

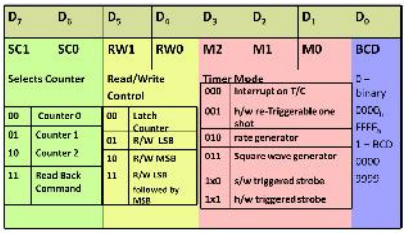
C_1 used in mode 0, LSB only, BCD

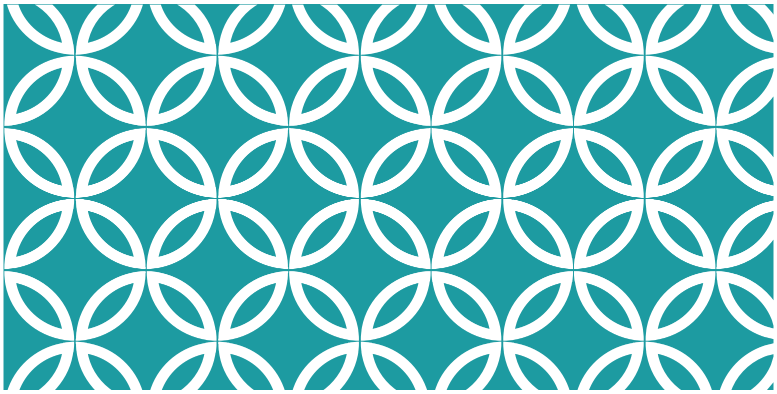
To be loaded with 99



Interface to the processor

```
cnt0    equ     50h
cnt1    equ     52h
creg    equ     56h
        mov     al,00110010b
        out     56h,al
        mov     al,01010001b
        out     56h,al
        mov     al,0
        out     50h,al
        mov     al,40h
        out     50h,al
        mov     al,99h
        out     52h,al
```


D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD
Selects Counter		Read/Write Control		Timer Mode			0 – binary 0000_h $FFFF_h$ 1 – BCD 0000 9999
00	Counter 0	00	Latch Counter	000	Interrupt on T/C		
01	Counter 1	01	R/W LSB	001	h/w re-Triggerable one shot		
10	Counter 2	10	R/W MSB	010	rate generator		
11	Read Back Command	11	R/W LSB followed by MSB	011	Square wave generator		
				1x0	s/w triggered strobe		
				1x1	h/w triggered strobe		



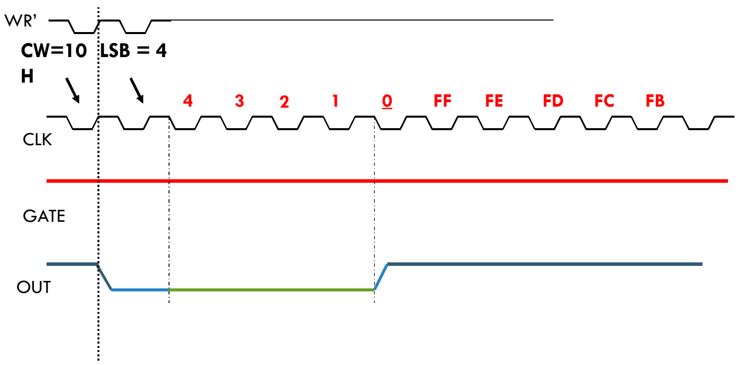
8253 Timer Modes

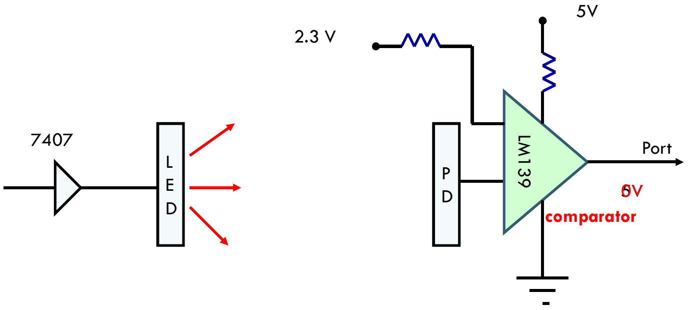
Mode 0

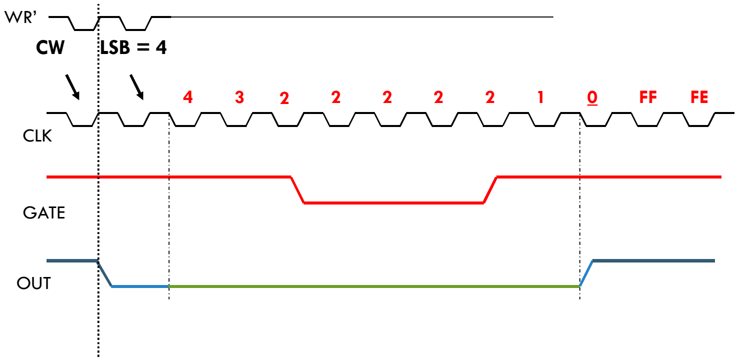
MODE 0

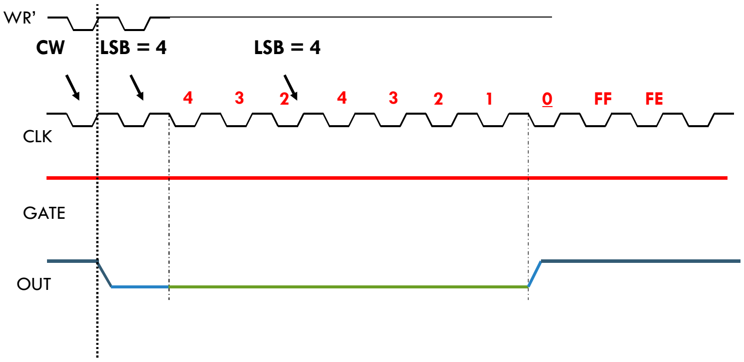
Event Counter

Interrupt on reaching Terminal Count









Mode 0

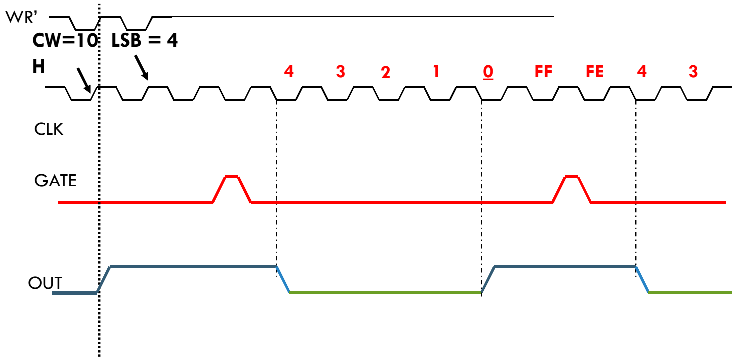
- **Interrupt on terminal count (event counter)**
- Out pin goes low when mode word or new count is written
- Now if clock is applied and gate=1, countdown begins
- Countdown stops if gate=0: resumes if gate=1
- If count written is N then OUT becomes high after N+1 clocks
- OUT remains high till a new count is written
- Countdown continues as FF_H, FE_H if gate =1
- Application – object counting

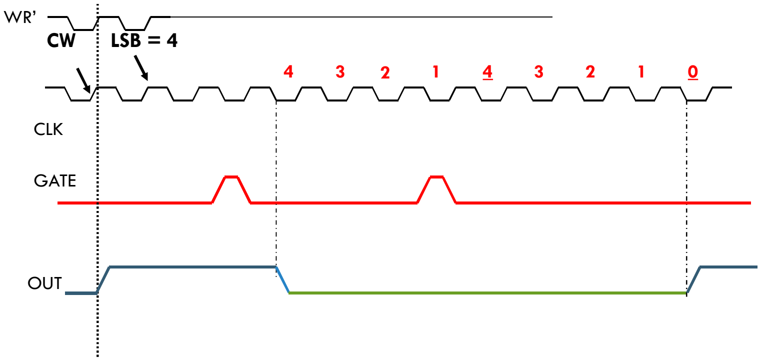
MODE 1

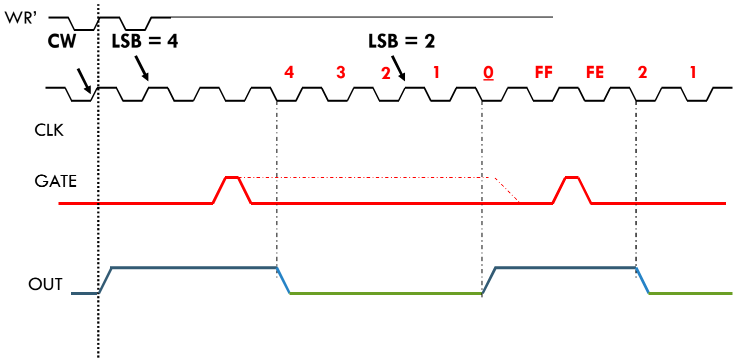
H/w Re-Triggerable One-Shot

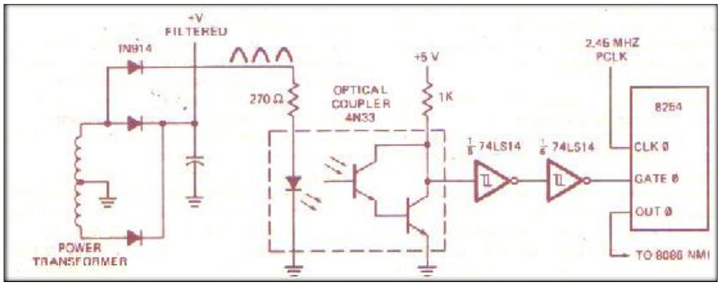
(Programmable One- Shot)

8253 Timer Modes
Mode 1





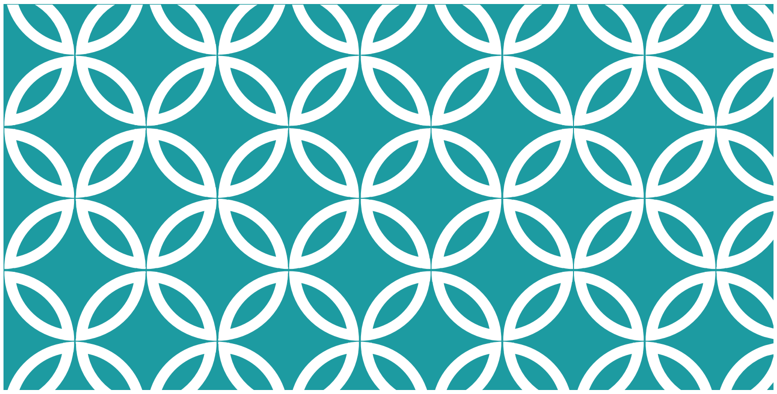




AC Power Failure Detection

Mode 1

- **Programmable one-shot- also h/w retriggerable one-shot**
- Two step process
  - Load count register
  - Send 0-to-1 pulse on GATE to trigger it
- When triggered ⇒ o/p goes low after one clock cycle & stays low for N clock cycles ⇒ goes high
- If gate is made low it does not stop counter
- A +ve transition at gate reloads the counter & countdown begins afresh
- A new count is not loaded till gate is triggered
- Application – detect ac power failure



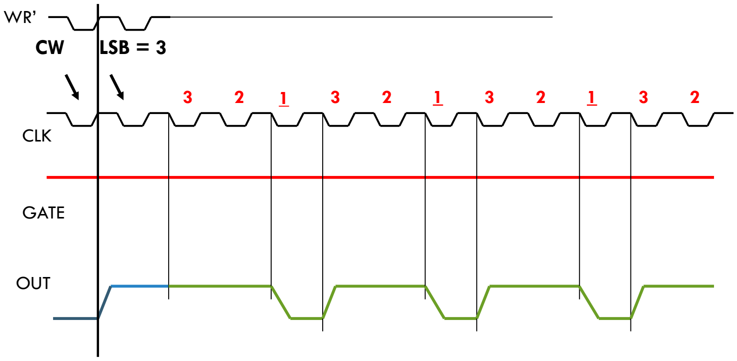
8253 Timer Modes

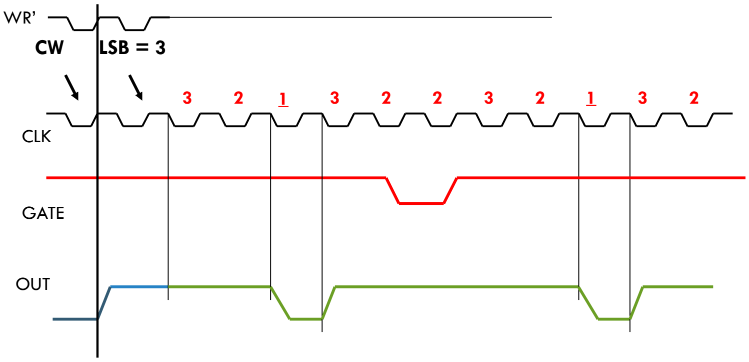
Mode 2

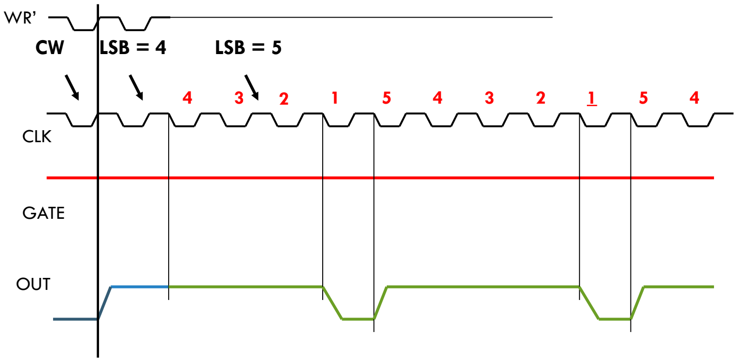
MODE 2

Rate Generator

(Divide-by-N Counter)







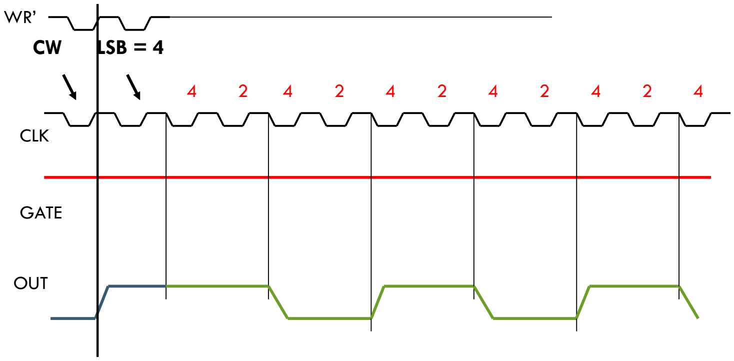
❑Mode 2

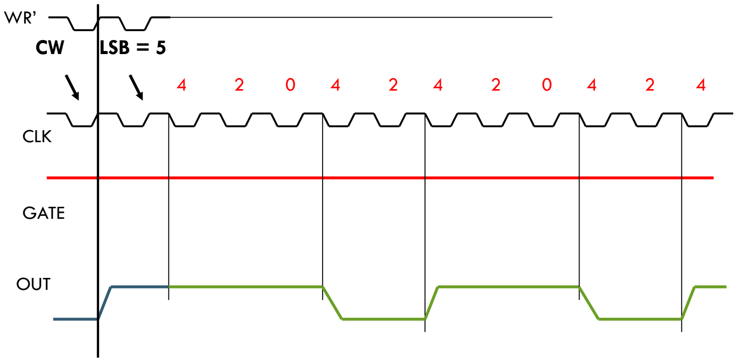
- ❑**Rate generator or divide-by-N counter**
- ❑Countdown starts one clock cycle after the gate is made high (or one cycle after the count is written if gate is already high)
- ❑On reaching a count of one, the OUT goes low for one cycle. If the counter is loaded with a number N, then OUT pin will go low for one clock cycle every N input clock pulses.
- ❑Now count is automatically reloaded and whole process repeats
- ❑If a new count is written then it is loaded only after previous countdown finishes
- ❑If gate is made low during countdown then counting stops and OUT is made immediately high
- ❑Application : frequency generation, real time clock

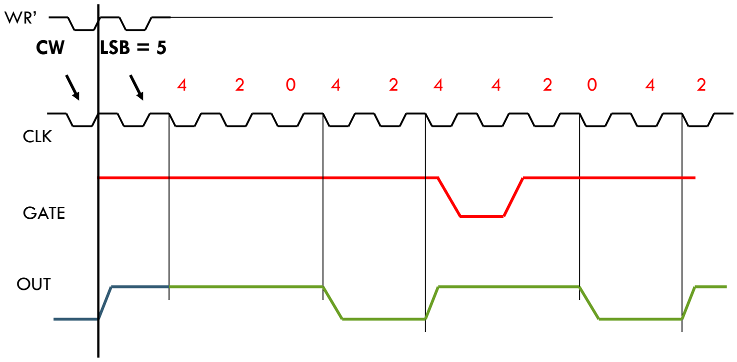
8253 Timer Modes
Mode 3

MODE 3

Square Wave Generator







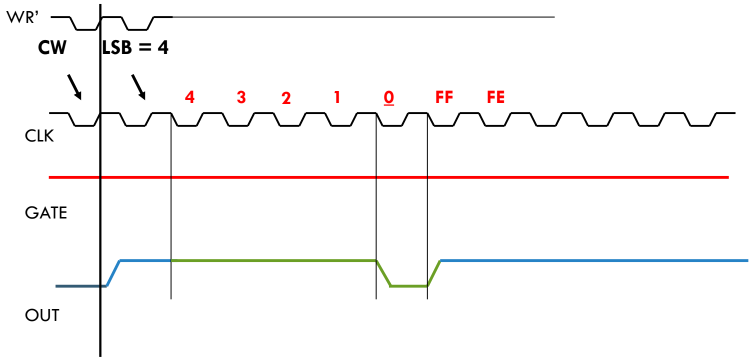
❑**Mode 3**

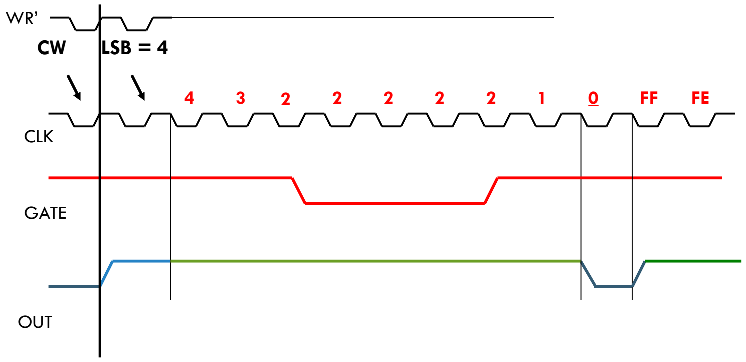
- ❑**Square wave generator**
- ❑If GATE = 1, OUT is a square wave (50% duty, or slightly off if N is odd)
- ❑If N is odd then will be high for (N+1)/2 and low for (N-1)/2
- ❑Each clock pulse decrements the counter by 2
- ❑Count is automatically reloaded on 2
- ❑If gate is made low during countdown then counting stops and when gate is made high again, counting continues.
- ❑Application: clock input generation for other devices, audio tone generator

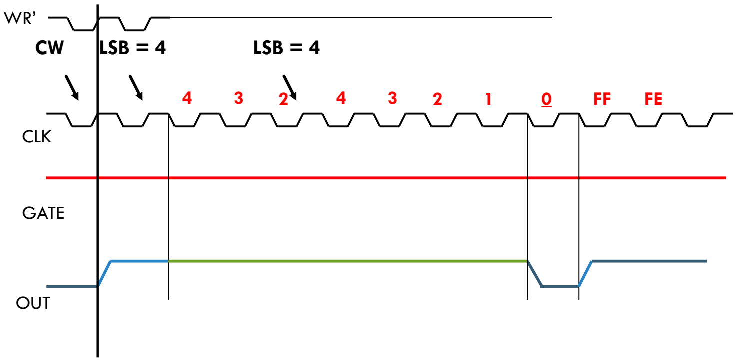
8253 Timer Modes
Mode 4

MODE 4

Software Triggered Strobe







❑Mode 4

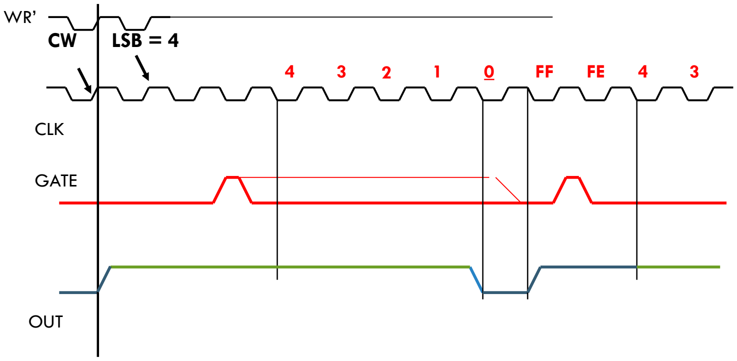
- ❑**Software triggered strobe**
- ❑If GATE =1, OUT goes low N+1 cycles after the count is written.
- ❑OUT is low for one clock cycle
- ❑Count must be reloaded to repeat the strobe
- ❑not automatically reloaded
- ❑If GATE→low, the OUT → high and count stops; count resumes (from where it stopped) when GATE→high
- ❑Application : I/O strobe

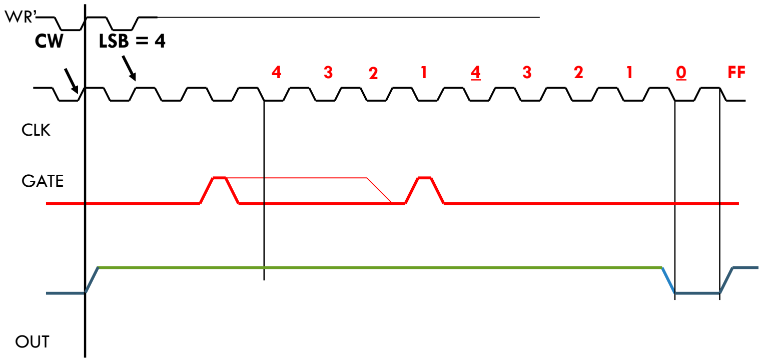
8253 Timer Modes

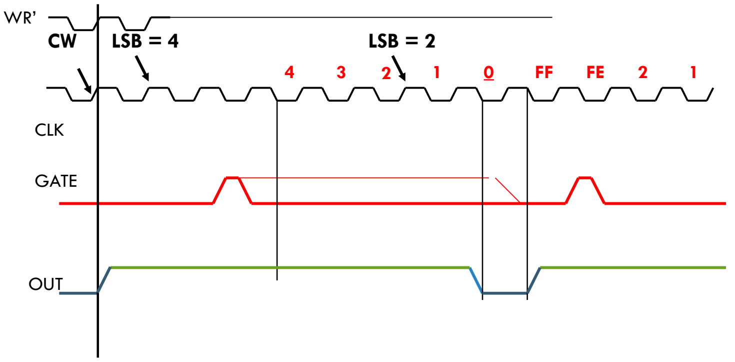
Mode 5 and read back

MODE 5

Hardware Triggered Strobe







❑Mode 5

- ❑**Hardware triggered strobe**
- ❑Like mode 4, but triggering is done with GATE instead
- ❑Count begins when 0-1 pulse hits GATE
- ❑OUT goes low N+1 cycles after gate is triggered.
- ❑OUT is low for one clock cycle
- ❑Gate must be triggered again to repeat the strobe
- ❑not automatically reloaded
- ❑If GATE→low, it does not stop countdown
- ❑ A trigger on gate in between countdown will reload the count and keep OUT high
- ❑Application: I/O handshake

Reading Count value of counter (s)

❑**8254 counters have latches on their outputs. Normally enabled during counting, so that latch outputs follow counter outputs.**

❑**When reading the current count value, we read data on outputs of these latches.**

❑**For reading the correct count….Counting must be stopped.**

❑**Can be done by removing clock, gate etc…….but not preferred as requires extra hardware..**

❑**So….**

❑**Latch the count before reading…..by**

❑**Sending counter latch command word at CR (A_1 A_0 =11) and then read.**

Read back command word (8254 only)

Select counter bits

1	1	Count	Status	C2	C1	C0	0

Latch count of selected counters

…status of …ed counters

Status Register (8254 only)

OUT	NULL	RW1	RW0	M2	M1	M0	BCD

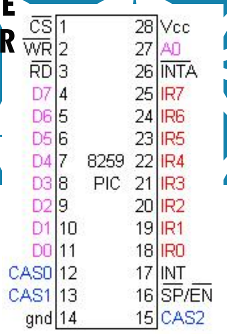
Counter mode

…for BCD …unter

Level of OUT pin

Read/write operation

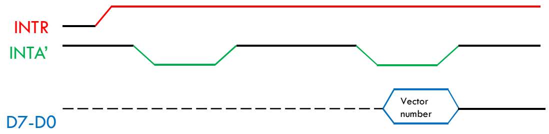
PROGRAMMABLE INTERRUPT CONTROLLER

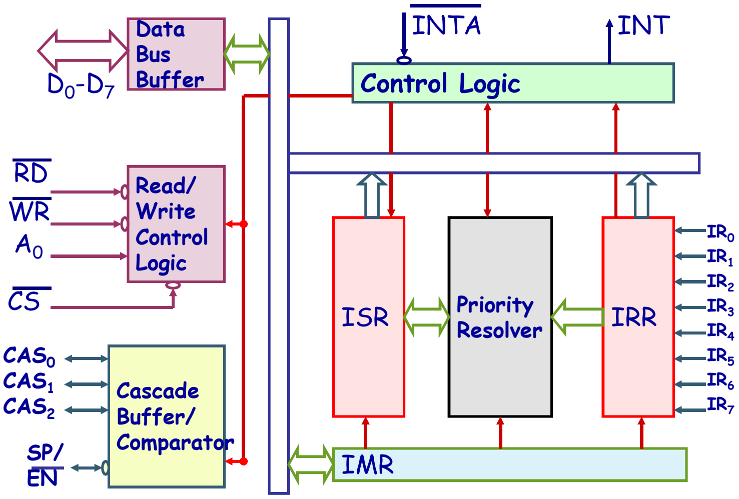


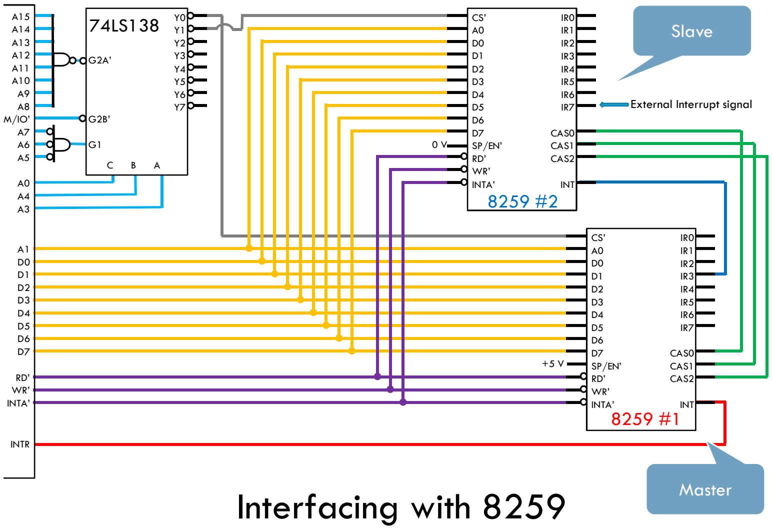
8259

80X86 - INTERRUPTS

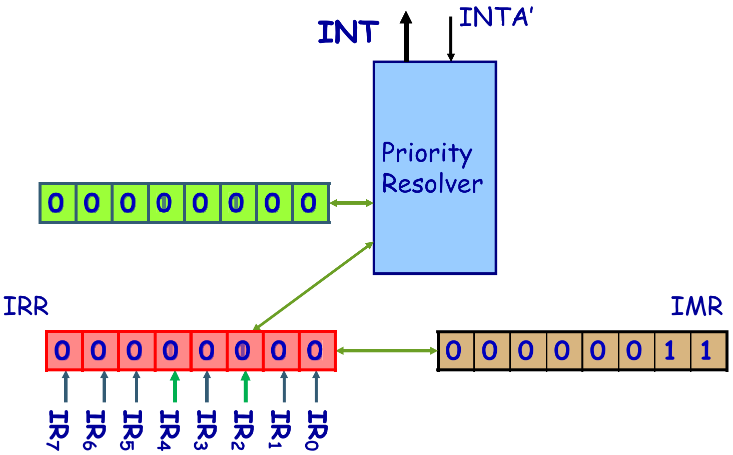
- ❑In response to INTR 80X86 expects a **vector number**
- ❑External hardware device required
- ❑It enters into INTA' machine cycle



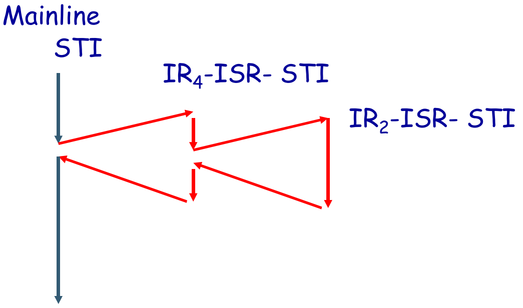


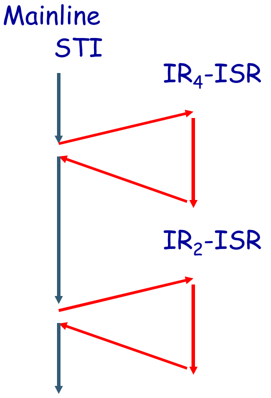


Interfacing with 8259



nxt





bck

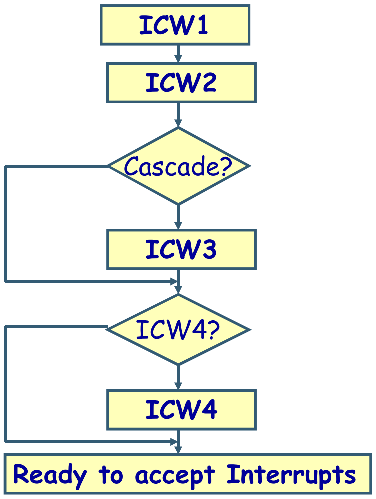
INITIALIZING 8259

Base Address $FF00_h$ – 2 addresses are $FF00_H$, $FF02_H$

4 ICWs

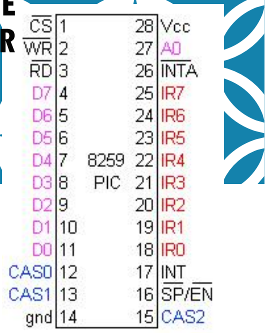
3 OCWs

Order of writing important for ICWs



PROGRAMMABLE INTERRUPT CONTROLLER

8259—Command Words



ICW$_x$	A$_0$	D$_7$				D			
1	0	A$_7$	A$_6$	A$_5$	1	LTIM	ADI	SNGL	IC4
2	1	A$_{15}$ T$_7$	A$_{14}$ T$_6$	A$_{13}$ T$_5$	A				
4	1	0	0	0	SFNM	BUF	M/S	AEOI	μPM

X for 8086

1=level triggered
0=edge triggered

Interval
of 4
of 8
8086

de

1=Special Fully Nested Mode

Master/

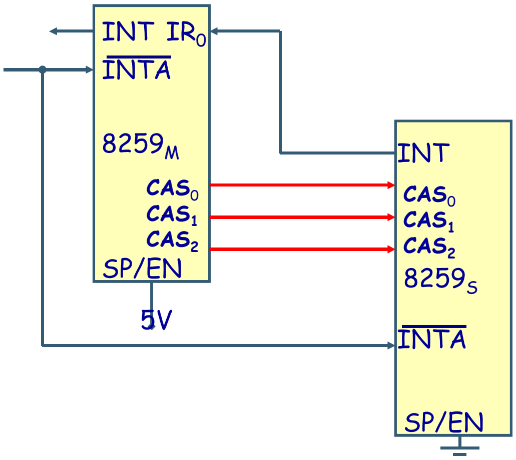
1=

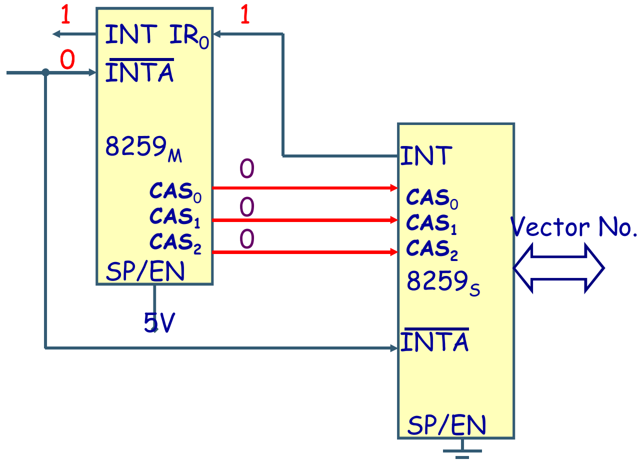
1 for 8086

BUF	M/S	
0	X	Non buffered Mode
1	0	Buffered Mode/Slave
1	1	Buffered Mode/Master

ICW$_x$	A$_0$	D$_7$	D$_6$	D$_5$	D$_4$	D$_3$	D$_2$	D$_1$	D$_0$
3	1	S$_7$	S$_6$	S$_5$	S$_4$	S$_3$	S$_2$	S$_1$	S$_0$
3S	1	0	0	0	0	0	ID$_2$	ID$_1$	ID$_0$

ICW3 – used for cascade





OCW$_x$	A$_0$	D$_7$	D$_6$	D$_5$	D$_4$	D$_3$	D$_2$	D$_1$	D$_0$
1	1	M$_7$	M$_6$	M$_5$	M$_4$	M$_3$	M$_2$	M$_1$	M$_0$
2	0	R	SL	EOI	0	0	L$_2$	L$_1$	L$_0$

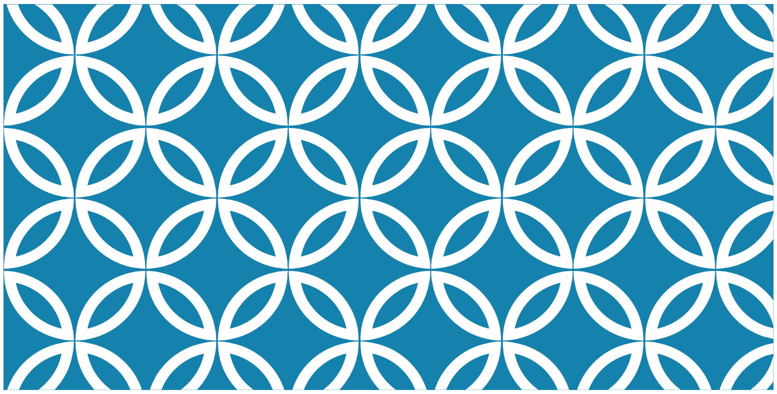
R	SL	EOI	Purpose
0	0	1	Non-specific EOI
0	1	1	Specific EOI
1	0	1	Rotate on non-specific EOI
1	0	0	Rotate on AEOI
1	1	1	Rotate on specific EOI
1	1	0	Set priority

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IS_7	IS_6	IS_5	IS_4	IS_3	IS_2	IS_1	IS_0
0	0	0	1	0	0	0	0
7	6	5	4	3	2	1	0
ISR4 SERVICED -EOI/AEOI							
0	0	0	0	0	0	0	0
2	1	0	7	6	5	4	3

OCW_x	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
3	0	0	ESMM	SMM	0	1	P	RR	RIS

Purpose	P	RR	RIS
Read IRR on next RD	0	1	0
Read ISR on next RD	0	1	1
Poll	1	x	x

Purpose	ESMM	SMM
No Action	0	x
Reset SMM	1	0
Set SMM	1	1



16650 PCI

UART

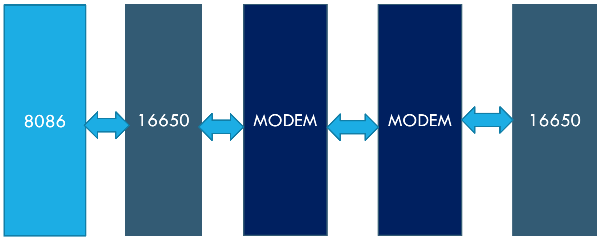
COMMUNICATION

- ❑ Parallel
- ❑ Serial
  - ❑ Synchronous
  - ❑ Asynchronous

SERIAL COMMUNICATION

- ❑ Simplex
- ❑ Half Duplex
- ❑ Full Duplex

SERIAL COMMUNICATION

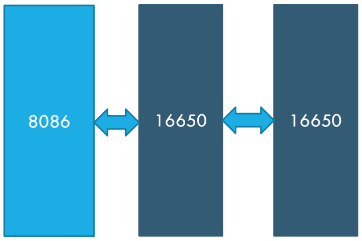


8086 ⇔ 16650 ⇔ MODEM ⇔ MODEM ⇔ 16650

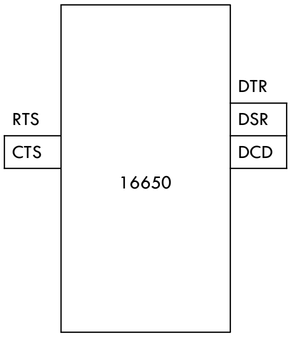
16650-MODEM

- DSR’
- DTR’
- CTS’
- RTS’
- RI’
- DCD’

SERIAL COMMUNICATION- NULL MODEM



NULL MODEM



SERIAL DATA LINES

- SIN
- SOUT

INTERFACE TO 8086

- A_0, A_1, A_2
- CS_0 CS_1 CS_2’
- ADS’
- RD,RD’
- WR, WR’
- RXRDY’
- TXRDY’
- MR
- INTR
- DDIS
- D_0- D_7

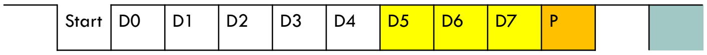
ADDRESSES

A_2	A_1	A_0	Function
0	0	0	RXB/TXB
0	0	1	Int Enable
0	1	0	Int Indentification/ FIFO Control
0	1	1	Line Control
1	0	0	Modem Control
1	0	1	Line Status
1	1	0	Modem Status
1	1	1	Scratch

CLOCK

- ❑ X1,X2
- ❑ BAUDOUT’
- ❑ RCLK
- ❑ OUT1’, OUT2’

FORMAT OF SERIAL DATA



Start	D0	D1	D2	D3	D4	D5	D6	D7	P	

INITIALIZING 16650

- ❑ Program LCR
- ❑ Program BRG

LCR

DL	SB	ST	P	PE	S	L1	L0

BDR – 18.432MHZ

BR	Divisor Value
110	10,473
300	3840
1200	920
2400	480
4800	240
9600	120
19,200	60
38,400	20
57,600	20
115,200	10

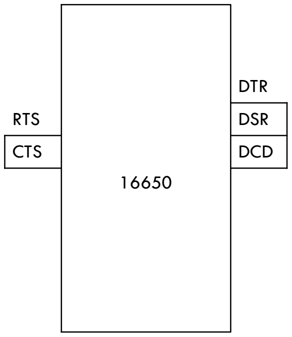
16650 PCI

UART

16650-MODEM

- DSR’
- DTR’
- CTS’
- RTS’
- RI’
- DCD’

NULL MODEM



SERIAL DATA LINES

- SIN
- SOUT

INTERFACE TO 8086

- A_0, A_1, A_2
- CS_0 CS_1 CS_2’
- ADS’
- RD,RD’
- WR, WR’
- RXRDY’
- TXRDY’
- MR
- INTR
- DDIS
- D_0- D_7

ADDRESSES

A_2	A_1	A_0	Function
0	0	0	RXB/TXB/BRL
0	0	1	Int Enable/BRH
0	1	0	Int Indentification/ FIFO Control
0	1	1	Line Control
1	0	0	Modem Control
1	0	1	Line Status
1	1	0	Modem Status
1	1	1	Scratch

CLOCK

- X1,X2
- BAUDOUT’
- RCLK
- OUT1’, OUT2’

FORMAT OF SERIAL DATA

	Start	D0	D1	D2	D3	D4	D5	D6	D7	P		

INITIALIZING 16650

- ❑ Program LCR
- ❑ Program BRG

LCR

DL	SB	ST	P	PE	S	L1	L0

BDR – 18.432MHZ

BR	Divisor Value
110	10,473
300	3840
1200	920
2400	480
4800	240
9600	120
19,200	60
38,400	20
57,600	20
115,200	10

FIFO CONTROL REGISTER

RT1	RT0	0	0	DMA	XMIT RST	REVC RST	EN

SAMPLE INITIALIZATION

9600 baud

8 data

Odd parity

1 stop

SA – $F0_H$

SAMPLE INITIALIZATION

```
LINE    EQU     0F6H
LSB     EQU     0F0H
MSB     EQU     0F2H
FIFO    EQU     0F4H
INIT:
        MOV     AL,10001011B
        OUT     LINE,AL
        MOV     AL,120
        OUT     LSB,AL
        MOV     AL,0
        OUT     MSB,AL
```

SAMPLE INITIALIZATION

```
;contd
        MOV     AL,00001011B
        OUT     LINE,AL
        MOV     AL,00000111B
        OUT     FIFO,AL
```

LINE STATUS REGISTER

ER	TE	TH	BI	FE	PE	OE	DR

SEND DATA

```
LSTAT   EQU     0FAH
DATA    EQU     0F0H
SEND:
X1:     IN      AL,LSTAT
        AND     AL,20H
        JZ      X1
        MOV     AL,DAT
        OUT     DATA,AL
```

RECEIVE DATA

```
LSTAT   EQU     0FAH
DATA    EQU     0F0H
READ:
X1:     IN      AL,LSTAT
        MOV     AH,AL
        AND     AL,01H
        JZ      X1
        MOV     AL,AH
        AND     AL,80H
        JZ      X2
        MOV     AL,0
        JMP     X3
X2:     IN      AL,DATA
X3:     MOV     DAT,AL
```

INTERRUPT CONTROL REGISTER

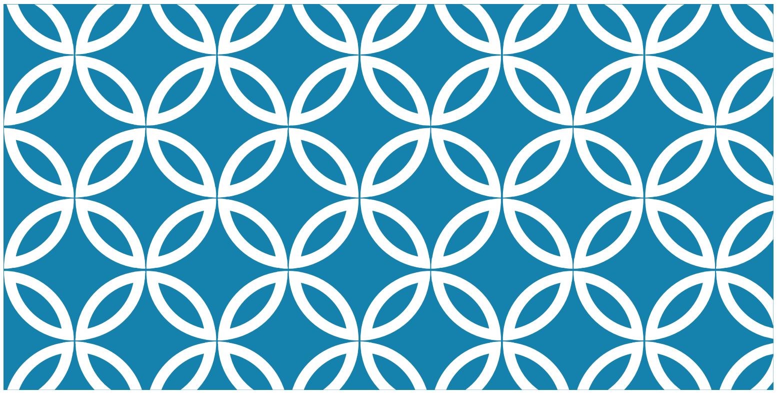
0	0	0	0	EM	EL	ET	ER

INTERRUPT ID REGISTER

				ID	ID	ID	PN

INTERRUPT ID

BITS	Priority	Type
0001	-	No Interrupt
0110	1	Receiver Error
0100	2	Receiver Data
1100	2	Character timeout – Nothing has been read from RX FIFO – for at least 4 characters time
0010	3	Transmitter Empty
0000	4	Modem Status



I/O INTERFACING

DMAC - 8237

DIRECT MEMORY ACCESS

Interrupt driven and programmed I/O require active CPU intervention

- Transfer rate is limited
- CPU is tied up

DMA is the answer

- Direct Memory Access is a method of transferring data between peripherals (I/O) and memory without using the CPU
- Maximum frequency of operation 15 MHz

DMA PROCESS

- Data is transferred between memory and disk directly without involving the processor
  - Interrupt still tells CPU when such a transfer has started and finished
- For DMA transfers
  - CPU tells the device controller the operation to perform and addresses involved.
  - Device controller then carries out the operation without bothering the CPU, using DMA
  - Bus controller arbitrates for the system bus
  - Device controller informs the CPU when complete via an interrupt

DMA MODES

DMA can operate in one of two modes

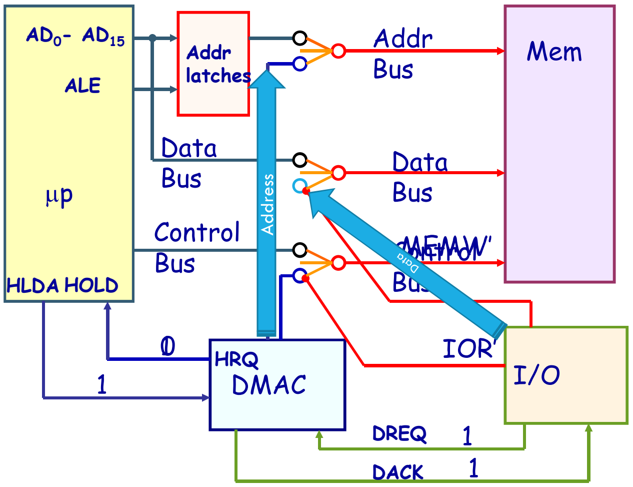
Bus master Mode

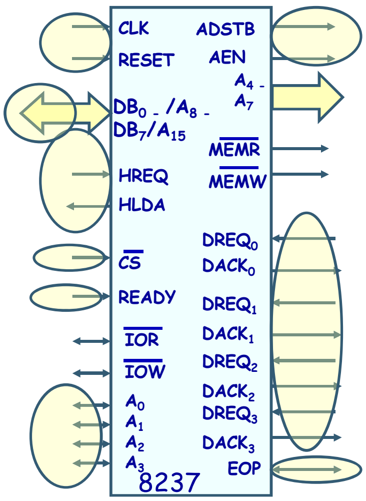
Bus Slave Mode

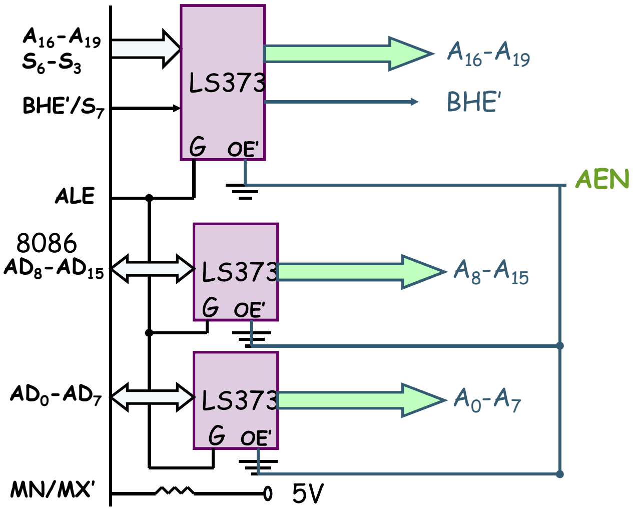
DMA can do data transfers mainly using two modes

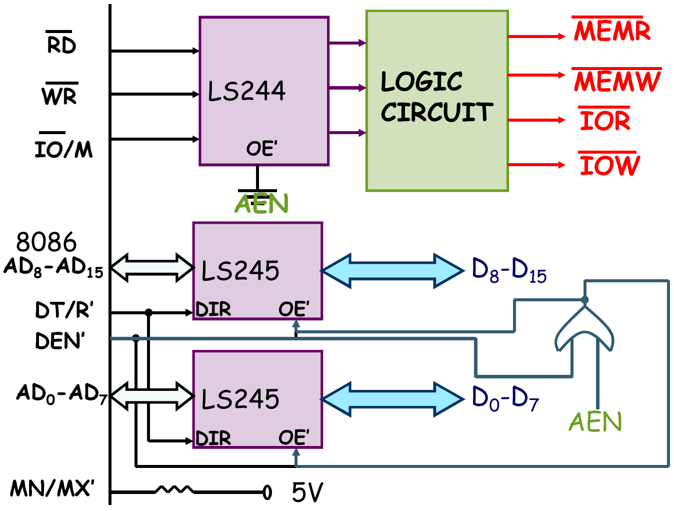
Byte Mode

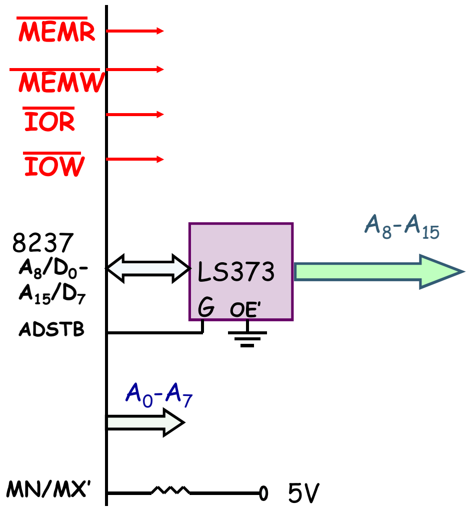
Burst Mode

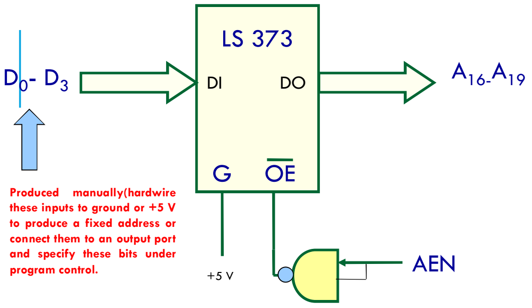




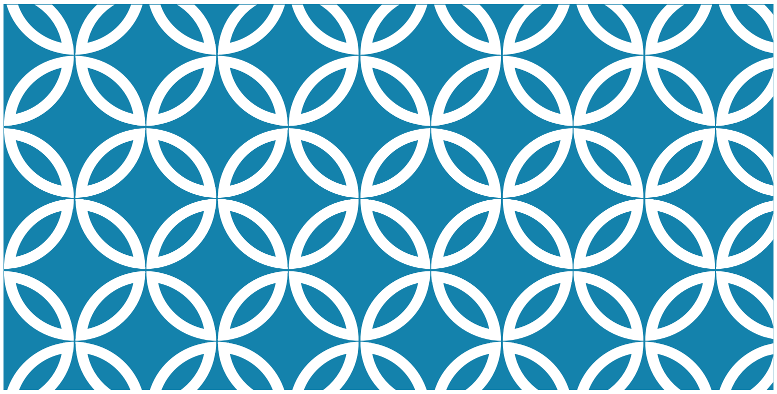








Produced manually(hardwire these inputs to ground or +5 V to produce a fixed address or connect them to an output port and specify these bits under program control.



I/O INTERFACING

DMAC – 8237

- Registers

REGISTERS

Current Address Register - 0000, 0010, 0100, 0110

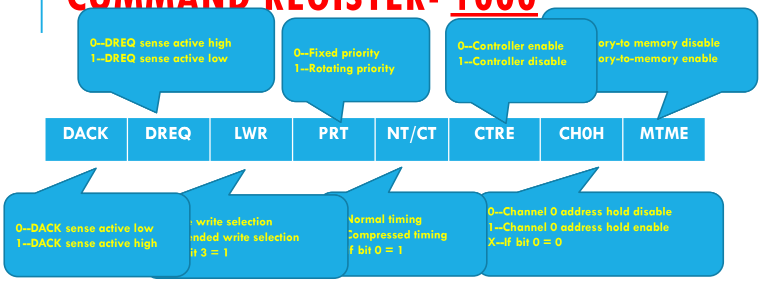
Current Word Count Register -0001, 0011, 0101 ,0111

Base Address Register

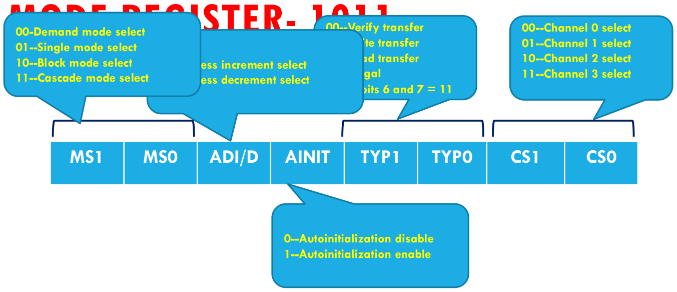
Base Word Count register

DMA channel I/O port addresses

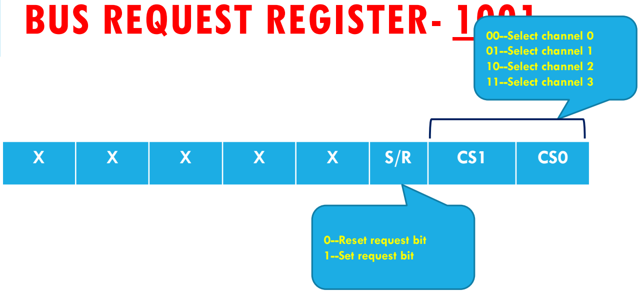
COMMAND REGISTER- 1000



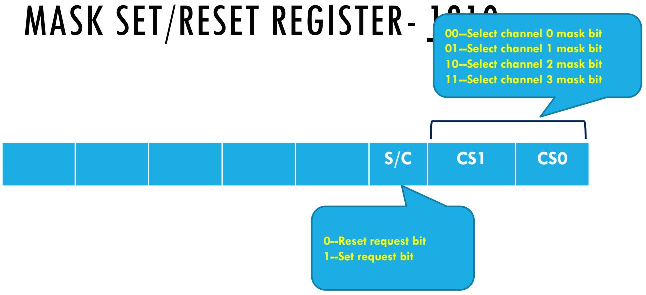
MODE REGISTER- 1011



BUS REQUEST REGISTER- 1001

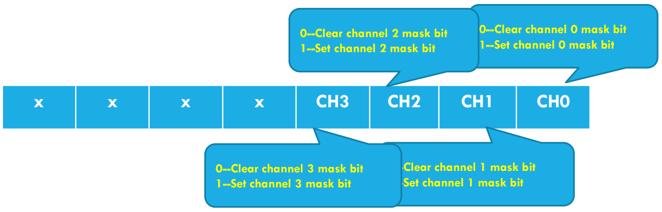


MASK SET/RESET REGISTER- 1010



00--Select channel 0 mask bit
01--Select channel 1 mask bit
10--Select channel 2 mask bit
11--Select channel 3 mask bit
S/C
CS1
CS0
0--Reset request bit
1--Set request bit

MASK REGISTER- 1111



x | x | x | x | CH3 | CH2 | CH1 | CH0

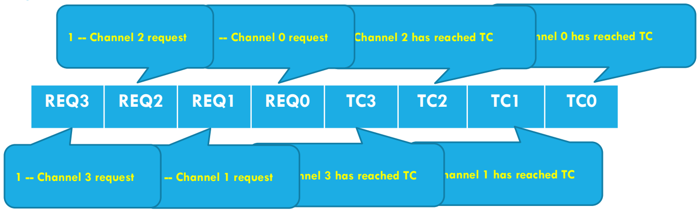
0--Clear channel 2 mask bit
1--Set channel 2 mask bit

0--Clear channel 0 mask bit
1--Set channel 0 mask bit

0--Clear channel 3 mask bit
1--Set channel 3 mask bit

Clear channel 1 mask bit
Set channel 1 mask bit

STATUS REGISTER- 1000



REQ3 | REQ2 | REQ1 | REQ0 | TC3 | TC2 | TC1 | TC0

1 -- Channel 2 request

-- Channel 0 request

Channel 2 has reached TC

nnel 0 has reached TC

1 -- Channel 3 request

-- Channel 1 request

nnel 3 has reached TC

nannel 1 has reached TC

ADDITIONAL ADDRESSES AND S/W COMMANDS

- Three additional software commands are used to control the operation of the 8237. These commands do not have a binary bit pattern.
- A simple output to the correct port number enables the software command. These commands are:

- **1100 – clear first/last FF:**
- **1101 – Master Clear**
- **1110 – Clear Mask Register**

ADDRESSES

Reg	Addr
CH0AR	0000
CH0CR	0001
CHIAR	0010
CH1CR	0011
CH2AR	0100
CH2CR	0101
CH3AR	0110
CH3CR	0111

Reg	Addr
CR/SR	1000
BR	1001
MRSR	1010
MR	1011
CFF	1100
MC/TR	1101
CMR	1110
MASKS	1111

EXAMPLE

Program DMAC for memory to memory transfer

Contents of memory location 01000-013FF to 02000-023FF

Base Address – 70_H

ALP

```
CH0       EQU    70H    ;channel 0 base and CAR address
CH0C      EQU    71H    ;channel 0 CWCR address
CH1       EQU    72H    ;channel 1 base and CAR address
CH1C      EQU    73H    ;channel 1 CWCR address
CR        EQU    78H    ;command register address
REQ       EQU    79H    ;request register address
MR        EQU    7BH    ;Mode register address
CFF       EQU    7CH    ;clear byte pointer F/F address
MASKS     EQU    7FH    ;MSR address
STATUS    EQU    78H    ;status register
```

Calling parameters:

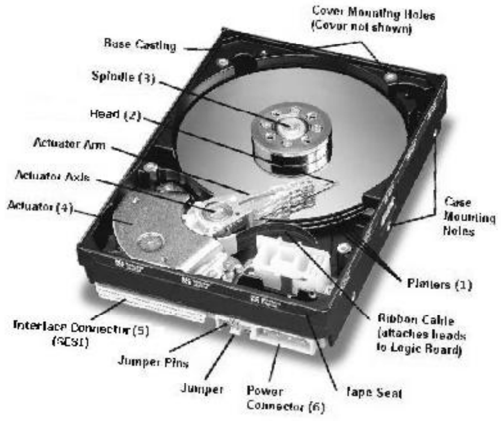
- **DS = segment of source**
- **ES = segment of destination**

```
TRANS PROC NEAR
MOV AX,DS ;program source address
SHL AX,4
ADD AX,SI
OUT CH0,AL
MOV AL,AH
OUT CH0,AL
MOV AX,ES ;program destination address
SHL AX,4
ADD AX,DI
OUT CH1,AL
MOV AL,AH
OUT CH1,AL
MOV AX,CX ;program count
DEC AX
OUT CH1C,AL
MOV AL,AH
OUT CH1C,AL
MOV AL,88H ;program mode
OUT MR,AL
MOV AL,85H
OUT MR,AL
MOV AL,1 ;enable mem to mem transfer
OUT CR,AL
MOV AL,0CH ;unmask channel 0/1
OUT MASKS,AL
MOV AL,4                        ;start DMA
OUT REQ,AL
X1: IN AL,STATUS;checking TC of channel 0
CMP AL,01H
JNZ X1
RET
TRANS ENDP
```

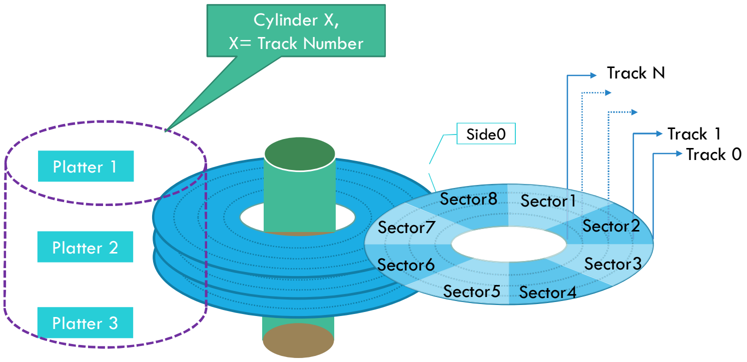
STORAGE DEVICE AND DISK ORGANIZATION

TYPES OF STORAGE DEVICE

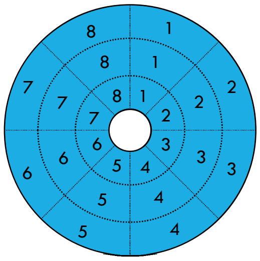
	Primary	Secondary
Cost	Expensive	Less than Primary
Capacity	Limited	Nearly Limited
Access Time	In Billionth of Second	In Millionth of Second
Processing	Directly Accessible	Routed Through Primary Storage



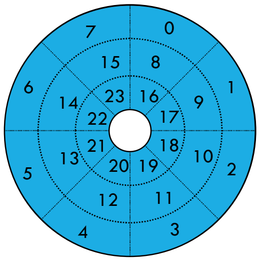
Hard Disk Cross Section



PHYSICAL SECTOR/ LOGICAL SECTOR



Physical Sector

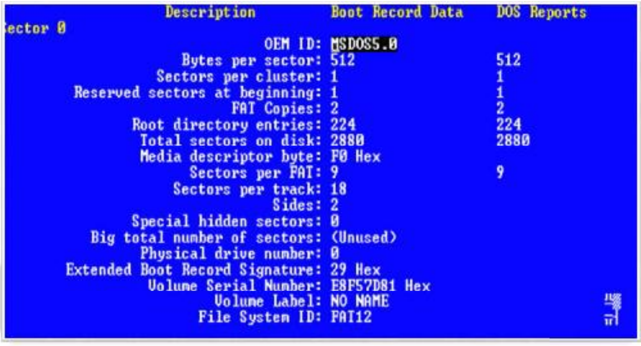


Logical Sector

Boot Sector Table

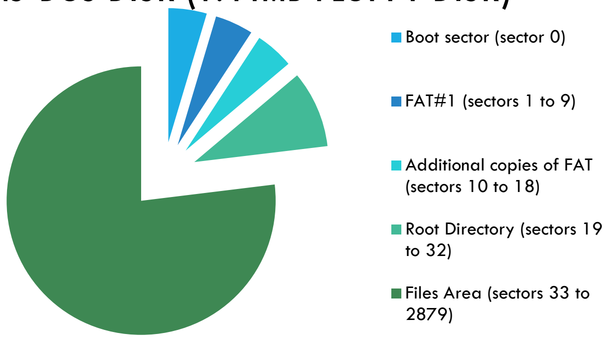
Location	Length (Bytes)	Description
00 – 02H	3	E9 XX XX or EB XX 90
03 – 0AH	8	Disk Formatting Program
0B – 0CH	2	Number of Bytes per Sector
0DH	1	Number of Sectors per Cluster
0E – 0FH	2	Reserved Sector (including boot sector)
10H	1	Number of File Allocation Table (FAT)
11 – 12H	2	Number of Root Directory Entries
13 – 14H	2	Total Sectors on Disk
15H	1	Media Descriptor Byte
16 – 17H	2	Number of Sectors per FAT
18 – 19H	2	Sectors per Track
1A – 1BH	2	Number of Heads
1C – 1FH	4	Number of Hidden Sectors
20 – 23H	4	Total Sector in Logical Volume (Volume Size > 32 MB)
24H	1	Physical Drive Number
25H	1	Reserved
26H	1	Extended Boot Signature Record (29H)
27 – 2AH	4	Volume Serial Number
2B – 35H	11	Volume Label
36 – 3DH	8	Reserved
3EH		Bootstrap

LOGICAL INTERPRETATION: BOOT SECTOR OF A FORMATTED FLOPPY



```
                    Description                   Boot Record Data      DOS Reports
ector 0
                                        OEM ID: MSDOS5.0
                              Bytes per sector: 512                     512
                           Sectors per cluster: 1                       1
                  Reserved sectors at beginning: 1                       1
                                    FAT Copies: 2                       2
                        Root directory entries: 224                     224
                         Total sectors on disk: 2880                    2880
                         Media descriptor byte: F0 Hex
                               Sectors per FAT: 9                       9
                             Sectors per track: 18
                                         Sides: 2
                        Special hidden sectors: 0
                     Big total number of sectors: (Unused)
                         Physical drive number: 0
                Extended Boot Record Signature: 29 Hex
                          Volume Serial Number: E8F57D81 Hex
                                  Volume Label: NO NAME
                                File System ID: FAT12
```

LOGICAL PARTITION: MS-DOS DISK (1.44MB FLOPPY DISK)



ROOT DIRECTORY

DOS keeps track of

- Each file and subdirectory as they are added to the disk by user
- Information about each file/directory is stored

Every time user adds a file, DOS updates directory area

- Name of the file
- Time and date of creation
- Length of file

Files names are limited to eight characters and a three character extension

ROOT DIRECTORY

Location	Length (Bytes)	Description
00 – 07H	8	Filename
08 – 0AH	3	File Extension
0BH	1	File Attribute
0C – 15H	10	Reserved
16 – 17H	2	Time of Creation or Last Updating
18 – 19H	2	Date of Creation or Last Updating
1A – 1BH	2	Starting Cluster Number
1C – 1FH	4	Size of File

ROOT DIRECTORY

Attribute Byte

BIT	7	6	5	4	3	2	1	0
Interpretation	Reserved	Reserved	Archive Bit	Directory	Vol. Label	System File	Hidden File	Read Only

Time Field

BIT	Interpretation
0 – 4	Binary Number 0 to 29 => 0 to 58 second
5 – 10	Binary Number 0 to 59 => 0 to 59 minutes
11 – 15	Binary Number 0 to 23 => 0 to 23 hours

Date Field

BIT	Interpretation
0 – 4	Binary Number 0 to 31 => 0 to 31 days
5 – 8	Binary Number 0 to 12 => 0 to 12 months
9 – 15	Year (base year as 1990)

FILE ALLOCATION TABLE

Purpose – create disk space for files

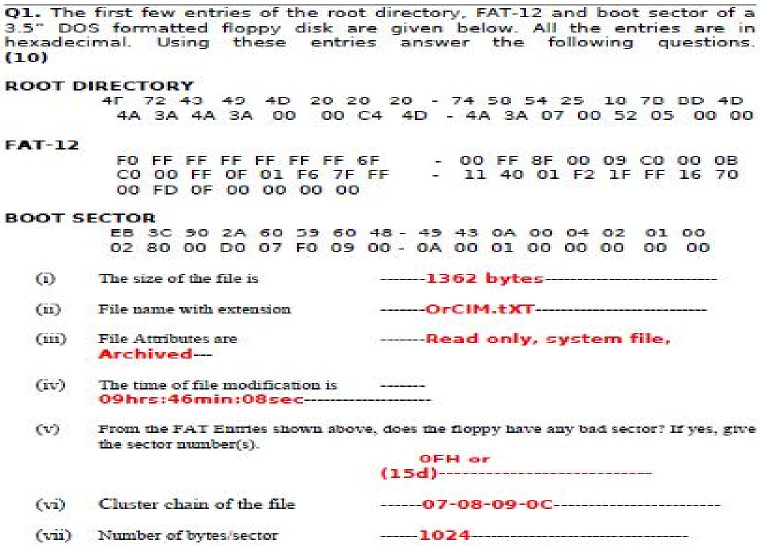
Contains entry for each cluster occupied by the file

Contains information of the group of clusters that are assigned to the file

Fields in FAT gives the cluster number that is allocated to file

FILE ALLOCATION TABLE

12 Bit Content	16 Bit Content	32 Bit Content	Interpretation
000H	0000H	00000000H	Cluster is available
FF0 – FF6H	FFF0 – FFF6H	FFFFFFF0 – FFFFFFF6H	Cluster is reserved
FF7H	FFF7H	FFFFFFF7H	Bad Cluster
FF8 – FFFH	FFF8 – FFFFH	FFFFFFF8 – FFFFFFFFH	Last Cluster in the File
Any other Value	Any other Value	Any other Value	Next Cluster in the File



Q1. The first few entries of the root directory, FAT-12 and boot sector of a 3.5" DOS formatted floppy disk are given below. All the entries are in hexadecimal. Using these entries answer the following questions. **(10)**

ROOT DIRECTORY

```
         4F  72  43  49  4D  20 20  20  - 74 50 54 25  10 7D DD 4D
          4A 3A 4A 3A  00   00 C4  4D  - 4A 3A 07 00 52 05  00 00
```

FAT-12

```
          F0  FF  FF  FF  FF  FF  FF  6F      -  00  FF  8F  00  09  C0  00  0B
          C0  00  FF  0F  01  F6  7F  FF      -   11  40  01  F2  1F  FF  16  70
          00  FD  0F  00  00  00  00
```

BOOT SECTOR

```
         EB  3C  90  2A  60  59  60  48 -  49  43  0A  00  04  02   01  00
         02  80  00  D0  07  F0  09  00 -  0A  00  01  00  00  00   00   00
```

(i) The size of the file is --------**1362 bytes**-----------------------------

(ii) File name with extension --------**OrCIM.tXT**-----------------------------

(iii) File Attributes are --------**Read only, system file, Archived**---

(iv) The time of file modification is -------- **09hrs:46min:08sec**--------------------

(v) From the FAT Entries shown above, does the floppy have any bad sector? If yes, give the sector number(s).
0FH or (15d)-----------------------------

(vi) Cluster chain of the file -------**07-08-09-0C**-------------------------

(vii) Number of bytes/sector ------**1024**-----------------------------------