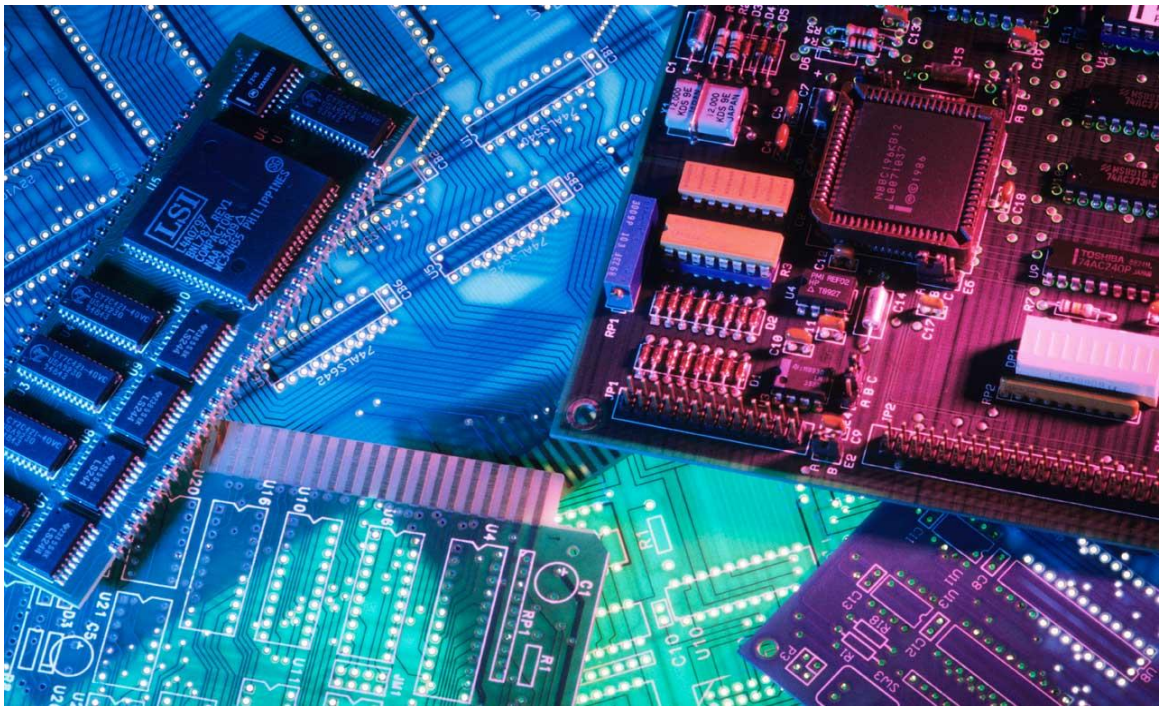
MICROPROCESSORS AND INTERFACING

III B.TECH I SEMESTER



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

LENDI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute, Approved by AICTE & Permanently Affiliated to JNTU-GV, Vizianagaram)

(Accredited By NAAC with A Grade and Accredited by NBA Tier-1)

Jonnada (Village), Denkada (Mandal), Vizianagaram District – 535 005

Phone No. 08922-241111, 241112

E-Mail: lendi_2008@yahoo.com website: www.lendi.org

UNIT – IV
Memory interfacing

Static RAM interfacing, EEPROM interfacing, 8255 programmable peripheral interface, interrupts and interrupt service routines, interrupt cycle of 8086, 8259A programmable interrupt controller.

INDEX

UNIT – IV
MEMORY INTERFACING

8086 Memory Banks

8086 has a 20 bit address bus and hence it can address 2^{20} or 1,048,576 addresses. In each location a byte is stored. So when a word is stored in the memory, it is stored in two consecutive memory locations. Strictly speaking, both memory read and memory write operations require more than one memory cycle. If we want 8086 to complete memory read and memory write operations to be completed with one machine cycle, the memory is to be organized in the form of two banks. Each bank will have 524,288 bytes each.

One memory bank contains all the even addressed locations like 00000, 00002 and 00004. The data lines of this bank are connected to the lower eight data lines, D_0 through D_7 of the 8086. The other memory bank has all the odd addressed locations like 00001, 00003 and 00005. The data lines of this bank are connected to the upper eight data lines, D_8 through D_{15} of the 8086. Address line A_0 is used as part of the enabling for memory in the lower bank. Address lines A_1 through A_{19} are used to select the desired memory device in the bank to address the desired byte in the service. These address lines from A_1 through A_{19} are also used to access a particular location in the upper bank. An additional signal called Bus High Enable (BHE – Active Low) is used to enable the upper memory bank. An external latch, strobed by ALE, grabs the BHE (Active Low) signal that holds it stable for the rest of the machine cycle. The following table shows the required logic levels on the BHE (Active Low) and A_0 signals for various types of memory accesses.

Address	Data type	BHE (active low)	A0	Bus cycles	Data lines used
0000	BYTE	1	0	ONE	D_0-D7
0000	WORD	0	0	ONE	D0-D15
0001	BYTE	0	1	ONE	D7-D15
0001	WORD	0 1	1 0	FIRST SECOND	D7-D15 D0-D7

MEMORY INTERFACING TO 8086

We have four common types of memory:

- Read only memory (ROM)
- Flash memory (EEPROM)
- Static Random access memory (SRAM)
- Dynamic Random access memory (DRAM).

The common pin connections to all memory devices are: The address input, data output or input/outputs, selection input and control input used to select a read or write operation.

Address connections: All memory devices have address inputs that select a memory location within the memory device. Address inputs are labeled from A_0 to A_n.

Data connections: All memory devices have a set of data outputs or input/outputs. Today many of them have bidirectional common I/O pins.

Selection connections: Each memory device has an input that selects or enables the memory device. This kind of input is most often called a **chip select** (CS), **chip enable** (CE) or simply **select** (S) input.

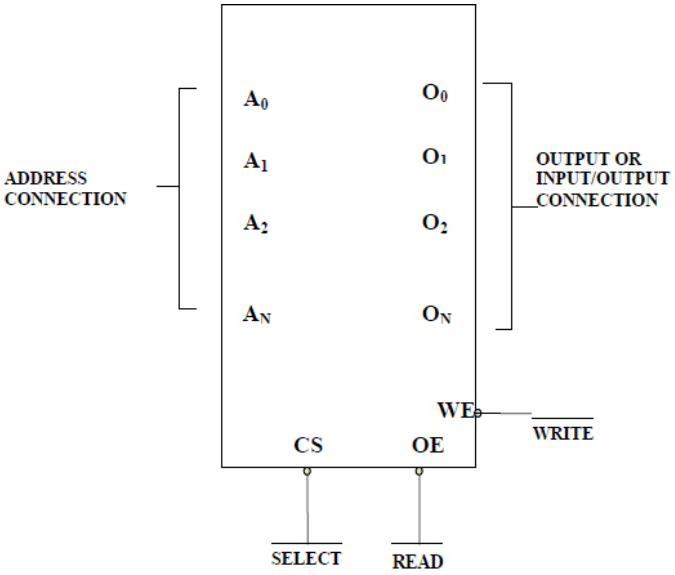


Figure: Memory component illustrating the address, data and, control connections

RAM memory generally has at least one $\overline{CS}$ or $\overline{S}$ input and ROM at least one $\overline{CE}$.

- If the $\overline{CE}$, $\overline{CS}$, $\overline{S}$ input is active the memory device perform the read or write.
- If it is inactive the memory device cannot perform read or write operation.
- If more than one $\overline{CS}$ connection is present, it is almost be active to perform read or write data.

Control connections: A ROM usually has only one control input, while a RAM often has one or two control inputs.

- The control input most often found on the ROM is the **output enable** ($\overline{OE}$) or **gate** ($\overline{G}$), this allows data to flow out of the output data pins of the ROM.
- If $\overline{OE}$ and the selected input are both active, then the output is enable, if $\overline{OE}$ is inactive, the output is disabled at its high-impedance state.
- The $\overline{OE}$ connection enables and disables a set of three-state buffer located within the memory device and must be active to read data.

A **RAM memory device** has either one or two control inputs. If there is one control input it is often called R/$\overline{W}$. This pin selects a read operation or a write operation only if the device is selected by the selection input ($\overline{CS}$).

- If the RAM has two control inputs, they are usually labeled $\overline{WE}$ or $\overline{W}$ and $\overline{OE}$ or $\overline{G}$.
- ($\overline{WE}$) **write enable** must be active to perform a memory write operation and $\overline{OE}$ must be active to perform a memory read operation.
- When these two controls $\overline{WE}$ and $\overline{OE}$ are present, they must never be active at the same time.

The **ROM** (Read Only Memory) permanently stores programs and data and data was always present, even when power is disconnected. It is also called as non-volatile memory. **EPROM** (Erasable Programmable Read Only Memory) is also erasable if exposed to high intensity ultraviolet light for about 20 minutes or less, depending upon the type of EPROM.

- We have PROM (Programmable Read Only Memory)
- RMM (Read Mostly Memory) is also called the flash memory.
- The flash memory is also called as an EEPROM (Electrically Erasable Programmable ROM), EAROM (Electrically Alterable ROM), or a NOVROM (Non-volatile ROM).
- These memory devices are electrically erasable in the system, but require more time to erase than a normal RAM.

INTERFACING WITH MEMORIES

The figure below shows a general block diagram of an 8086 memory array. In this, the 16-bit word memory is partitioned into high and low 8-bit banks on the upper halves of the data bus selected by BHE, and A_0.

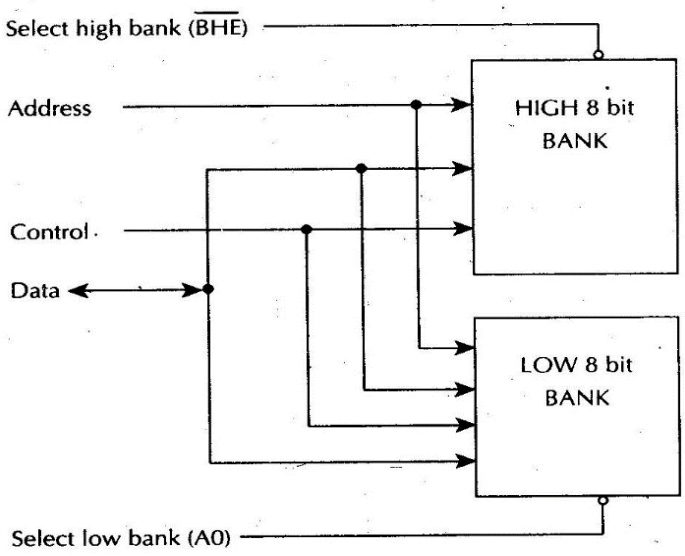


Figure: Block diagram of 8086 memory array

STATIC RAM INTERFACING

Since static RAMs are read/write memories, both A_0 and BHE' must be included in the chip select/enable decoding of the devices and write timing must be considered in the compatibility analysis.

For each static RAM, the memory data lines must be connected to either the upper half AD_{15}-AD_0 or lower half AD_7-AD_0 of the 8086 data lines.

For static RAMs without output enable pins, read and write lines must be used as enables for chip select generation to avoid bus contention. If read and write lines are not used to activate the chip selects, static RAMs with common input/output data pins such as 2114 will face extreme bus contentions between chip selects and write active. The 8086 A_0 and BHE pins must be used to enable the chip the chip selects. A possible way of generating chip selects for high and low static RAM banks is given in the below figure**.** Note that Intel 8205 decoder has three enables E_1, E_2, and E_3, three inputs A_0 and A_2, and eight outputs O_0-O_7.

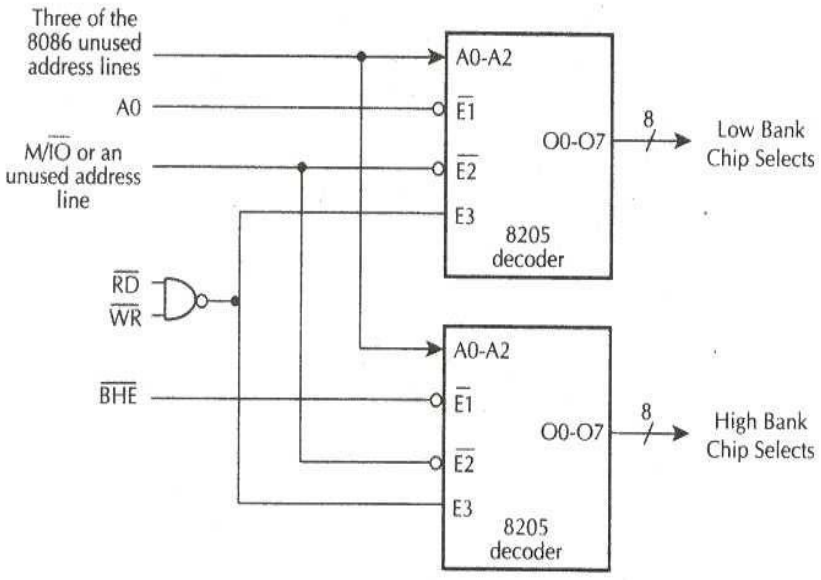


Figure: Generating chip selects for SRAM

Procedure of SRAM Inter facing:

Arrange the available memory chips so as to obtain 16-bit data bus width. The **upper 8-bit** bank is called **odd address Bank** and **Lower** is called **even address Bank**. Connect available memory address lines of memory chips with those of the up and also connect the memory WR and RD inputs to the corresponding processor control signal. Connect the 16-bit data bus of the memory bank to up. The remaining addressing lines, BHE and A0 are used for decoding chip select signals for odd and even memory banks. The CS of memory is derived from the output of the decoding circuit.

EEPROM INTERFACING

ROMs and EPROMs are the simplest memory chips to interface to the 8086. Since ROMs and EPROMs are read-only devices, A_0 and BHE' are not required to be part of the chip enable/select decoding. The 8086 address lines must be connected to the ROM/EPROM chip chips starting with A_1 and higher to all the address lines of the ROM/EPROM chips. The 8086 unused address lines can be used as chip enable/select decoding.

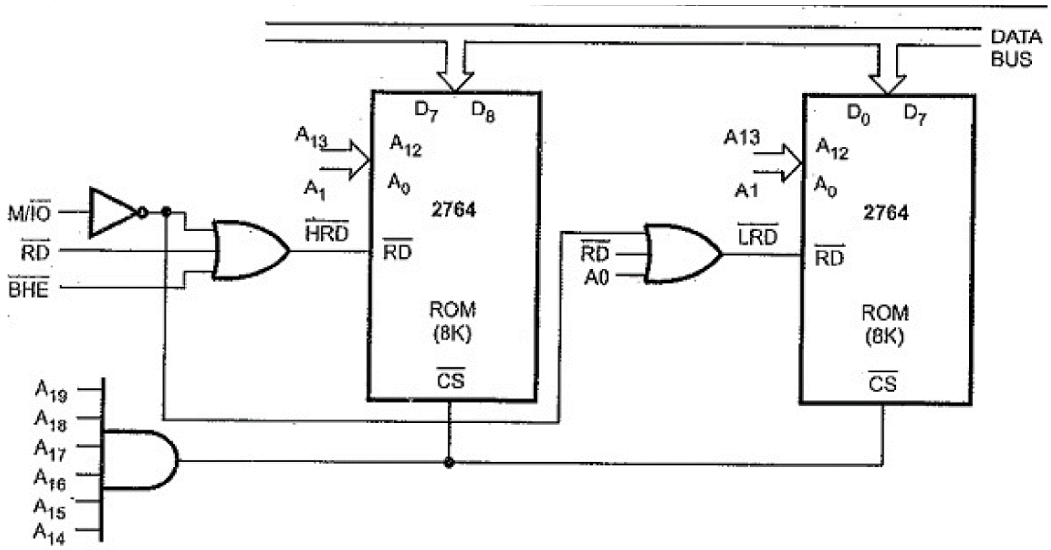


Figure: Generating chip selects for EEPROM

To interface the ROMs/RAMs directly to the 8086-multiplexed bus, they must have output enable signals. The below figure shows the 8086 interfaced to two 2764s (8K X 8). Byte accesses are obtained by reading the full 16-bit word onto the bus with the 8086 discarding the unwanted byte and accepting the desired byte.

The semiconductor memories are organized as two-dimensional arrays of memory locations. For example: 4K*8 or 4K byte memory contains 8-bit data and only one of the 4096 locations can be selected at a time. For addressing 4K bytes of memory, 12 address lines are required. For N memory locations, n address lines are required where n is

$$n = \log_2 N$$

For 4096 Locations, $n = \log_2 4096 = 12$

Note: If out of N memory Locations, only P memory locations are to be interfaced, then the least significant 'P' address lines out of available 'n' lines can be directly connected from the microprocessor to the memory chip while the remaining (n-p) higher order address lines may be used for address decoding (as I/P s to the chip selection logic). The output of the decoding circuit is connected with pin of the memory chip CS.

Example 1: Interface two 4K*8 EPROMs and two 4K*8 RAM chips with 8086.

Solution: After Reset, the CS and IP are initialized to form address FFFF0H. Hence, this address must lie in the EPROM. The address of RAM may be selected anywhere in the 1 MB address space of 8086, but we will select the RAM.

= Two 4K byte Location = 8K bytes of EPROM =1024 *8 = 8192. Locations =2^{13}.

Therefore 13 address Lines (A0-A12) are needed. A13-A19 are used for decoding to generate the chip select. The two 4K*8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. Depending upon the status of A0, BHE lower byte or higher byte or whole word is accessed from even/odd address.

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						EPROM						8K × 8								

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
						RAM						8K × 8								
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure: Address mapping

- FE000h-FFFFFh=ROM
- FC000h-FDFFFh-RAM

BHE'	**A0**	**Indication**
0	0	Whole word
0	1	Upper byte from or to odd address
1	0	Lower byte from or to even address
1	1	None

The memory system in this example contains in total four 4K x 8 memory chips. The two 4K x 8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. If A0 is 0, i.e. the address is even and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address. If A0 is 1, i.e. the address is odd and is in RAM, the BHE goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address.

If the selected addresses are in ROM, the respective ROM chips are selected. If at a time A0 and BHE both are 0, both the RAM and ROM chips are selected, i.e. the data transfer is of 16 bits.

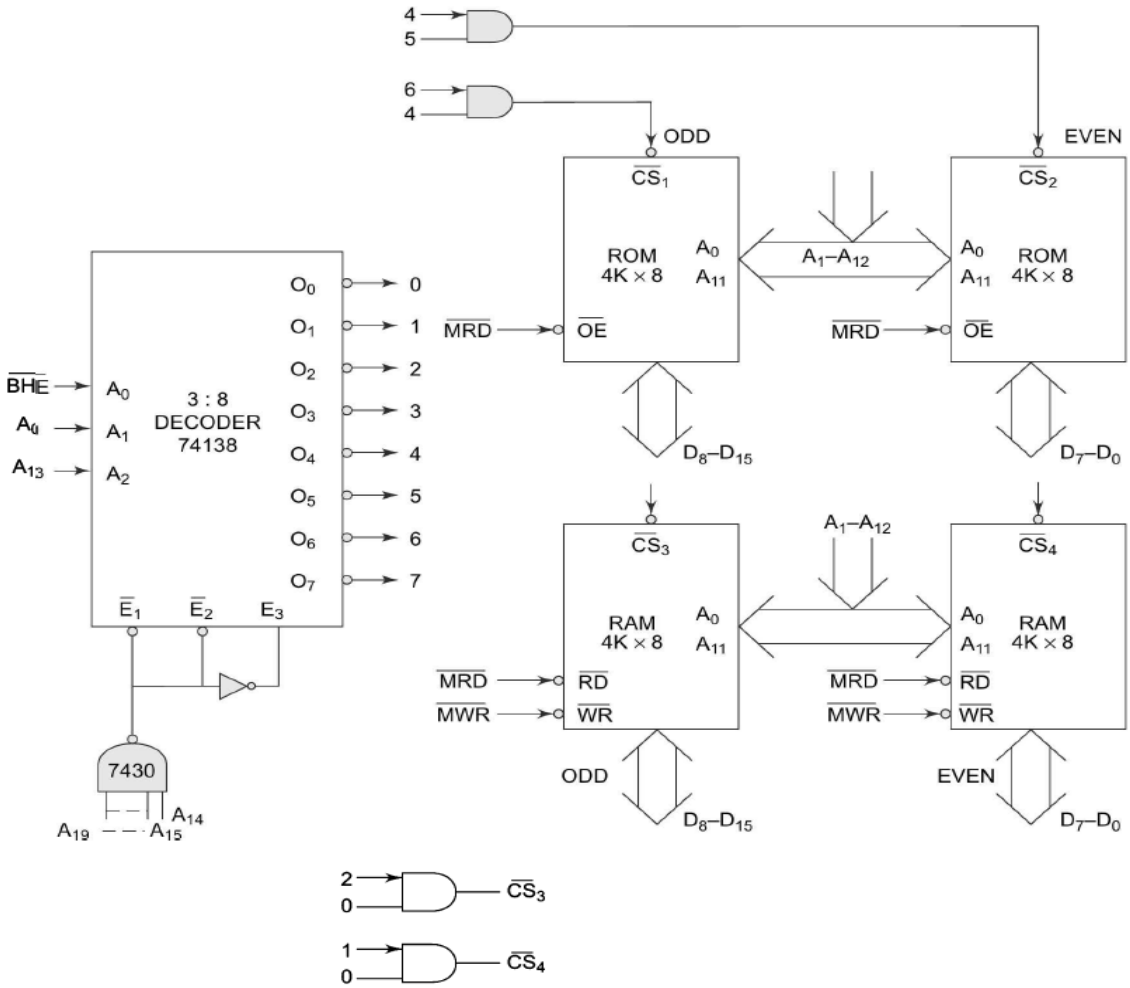


Figure: Interfacing diagram of ERPROM and RAM

Note:
A13 A0 BHE' = 1 1 0 - odd address
A13 A0 BHE' = 1 0 1 - even address
ROM Interfacing, A13 is always 1,
RAM Interfacing, A13 is always 0

The two 4K x 8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. If **A0 is 0**, i.e. the **address is even** and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address. If **A0 is 1**, i.e. the **address is odd** and is in RAM, the BHE goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address. If the selected addresses are in ROM, the respective ROM chips are selected. If at a time A0 and BHE both are 0, both the RAM or ROM chips are selected. Selection of Chips is shown in the table.

Decoder I/P → *Address/ $\overline{BHE}$ →*	A_2 A_{13}	A_1 A_0	A_0 $\overline{BHE}$	*Selection/* *Comment*
Word transfer on $D_0 - D_{15}$	0	0	0	Even and odd addresses in RAM
Byte transfer on $D_7 - D_0$	0	0	1	Only even address in RAM
Byte transfer on $D_8 - D_{15}$	0	1	0	Only odd address in RAM
Word transfer on $D_0 - D_{15}$	1	0	0	Even and odd addresses in ROM
Byte transfer on $D_0 - D_7$	1	0	1	Only even address in ROM
Byte transfer on $D_8 - D_{15}$	1	1	0	Only odd address in ROM

Example 2: It is required to interface two chips of 16K x 8 ROM and two chips of 32K x 8 RAM with 8086. Select the EPROM address suitably. The RAM address must start at 00000h. Show the implementation of this memory system.

Solution: The last address in the map of 8086 is FFFFFh. After resetting, the processor starts from FFFF0h. Hence this address must lie in the address range of EPROM.

Addresses	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								32KB			EPROM									
F8000H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0FFFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								64KB RAM												
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

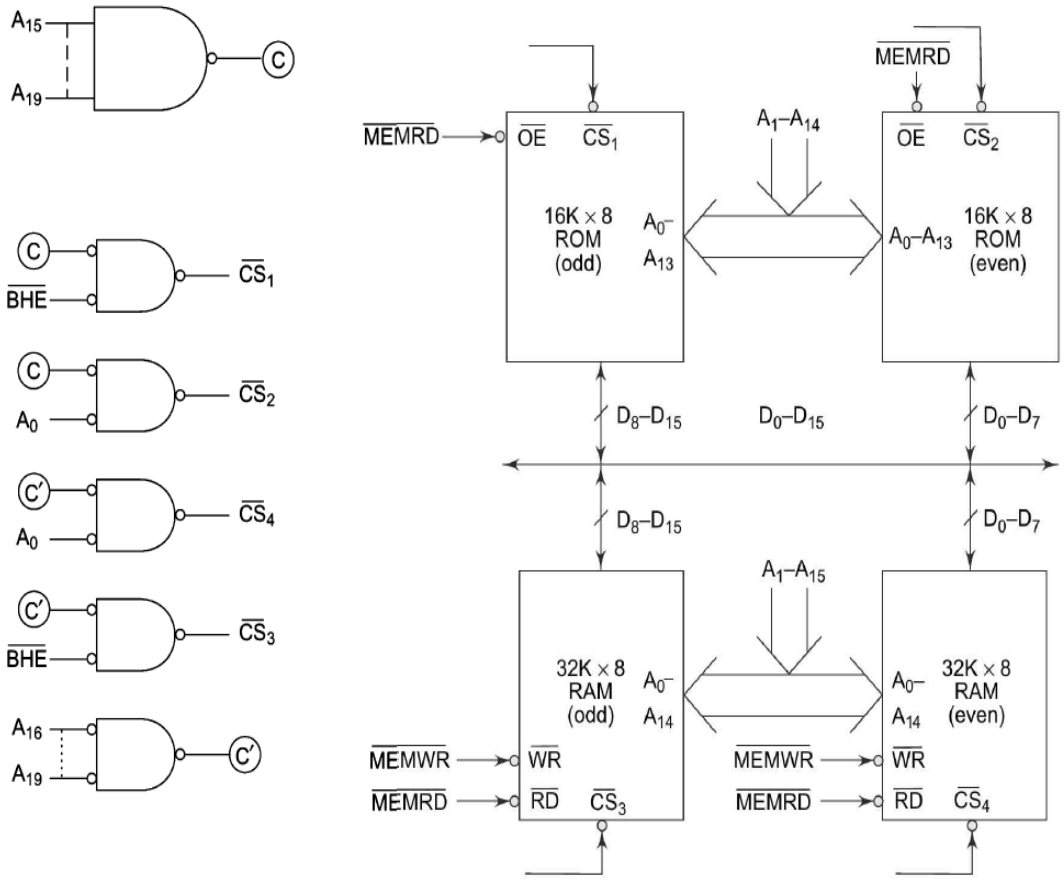


Figure: Interfacing diagram

For 32 KB of ROM, number of address lines required -15 (A_0-A_{14}) and for 64 KB of RAM, number of address lines required -16 (A_0-A_{15}). It is better not to use a decoder to implement the above map because it is not continuous. There is some unused address space between the last RAM address (0FFFFh) and the first EPROM address (F8000h).

For **32 KB of EPROM**, number of address lines required -15 **(A0-A14),** hence A15-A19 are used for chip selection. For **64 KB of RAM**, number of address lines required -16 **(A0-A15),** hence A16-A19 are used for chip selection.

Interfacing I/O ports

I/O ports are the devices through which the microprocessor communicates with other devices or external data sources/destinations. Input activity, as one may expect, is the activity that enables the microprocessor to read data from external devices, for example keyboards, joysticks, mouse etc. These devices are known as input devices as they feed data into a microprocessor system. Output activity transfers data from the microprocessor to the external devices, for example CRT display, 7-segmentdisplays, printers etc. The devices which accept the data from a microprocessor system are called output devices.

Thus for a microprocessor the input activity is similar to read operation, while the output activity is similar to write operation i.e. an input device can only be read and an output device can only be written. Hence IORD operation is for reading data from an input device and IOWR operation is for writing data to an output device After executing an OUT operation, the data appears on the data bus and simultaneously a device select signal is generated from the address and control signals.

INTEL 8255: PROGRAMMABLE PERIPHERAL INTERFACE (or)
PROGRAMMABLE INPUT- OUTPUT PORT

The parallel input- output port chip 8255 is also called **as programmable peripheral input-output port**. The 8255A is a general purpose programmable I/O device designed for use with Intel 8-bit, 16- bit and higher capability microprocessors. It consists of **three 8-bit bidirectional I/O ports** (**24 Input/output lines**) in **two groups of 12 lines** each that can be configured to meet different system I/O needs.

The three ports are PORT A, PORT B and PORT C. The three ports are divided in two groups **Group A** (PORT A and upper PORT C), **Group B** (PORT B and lower PORT C). Port A contains one 8-bit output latch/buffer and one 8-bit input buffer. Port B is same as PORT A. However, PORT C can be split into two parts PORT C lower (PC_0-PC_3) and PORT C upper (PC_7-PC_4). **Both the port Cs are assigned the same address.** Thus one can use three 8 bit I/O ports or two 8 bit I/O ports and two 4 bit I/O ports from 8255. All of these **ports can function independently either as input or as output ports**. This can be achieved by programming the bits of an internal register of 8255 called as **control word register CWR**.

ARCHITECTURE OF 8255:

The 8 bit data bus buffer is controlled by the read/write control logic. The read/write control logic manages all the internal and external transfers of both data and control words. $\overline{RD}$, $\overline{WR}$, A_1, A_0 and RESET are the input signals, provided by the microprocessor to the READ/WRITE control logic of 8255.

The 8-bit, 3-state bidirectional buffer is used to interface the 8255 internal data bus with the external system data bus. This buffer receives or transmits data upon the execution of input or output instructions by the microprocessor. The control words and status information is also transferred through the buffer.

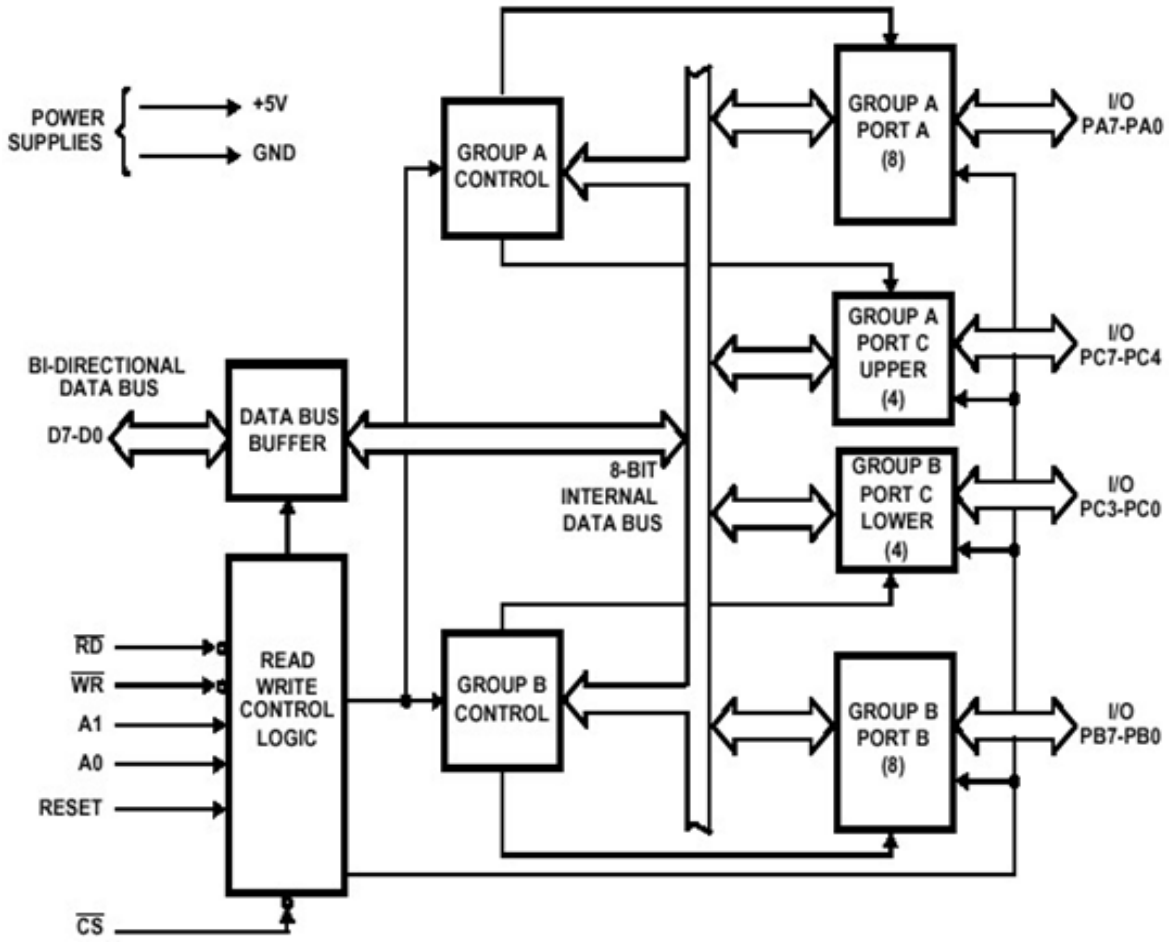


Figure: Internal architecture of 8255

Group A and Group B controls: These block receive control from the CPU and issues commands to their respective ports.

Group A – Port A and Port C Upper (PC7 –PC4)

Group B – Port B and Port C Lower (PC3 – PC0)

Control word register can only be written into no read operation of the CW register is allowed.

Port A: This has an 8 bit latched/buffered output and 8 bit input latch. It can be programmed in 3 modes – mode 0, mode 1, and mode 2.

Port B: This has an 8 bit latched / buffered O/P and 8 bit input latch. It can be programmed in mode 0, mode1.

Port C: This has an 8 bit latched input buffer and 8 bit output latched/buffer. This port can be divided into two 4 bit ports and can be used as control signals for port A and port B. it can be programmed in mode 0.

PIN DESCRIPTION OF 8255:

The signal description of 8255 are as follows:

PA7-PA0: These are eight port A lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.

PC7-PC4: Upper nibble of port C lines. They may act as either output latches or input buffers lines. This port also can be used for generation of handshake lines in mode 1 or mode 2.

PC3-PC0: These are the lower port C lines, other details are the same as PC7-PC4 lines.

PB0-PB7: These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.

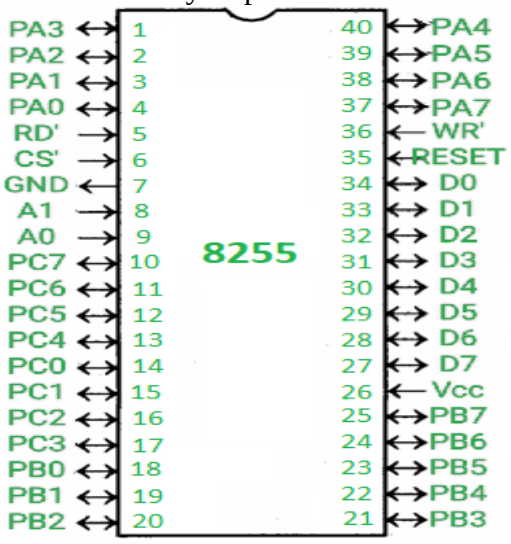


Figure: 8255 pin configuration

D_7-D_0	Data Bus (Bi-Directional)
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}	+5 Volts
GND	0 Volts

RD: This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.

WR: This is an input line driven by the microprocessor. A low on this line indicates write operation.

CS: This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR signals, otherwise RD and WR signal are neglected.

A1-A0: These are the address input lines and are driven by the microprocessor. These lines A1-A0 with RD, WR and CS from the following operations for 8255.These address lines are used for addressing any one of the four registers, i.e. three ports and a control word register as given in table below:

Table: Basic operations of 8255

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	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
					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
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A0 and A1 pins of 8255 are connected with A1 and A2 respectively.

D0-D7: These are the data bus lines those carry data or control word to/from the microprocessor.

RESET: Logic high on this line clears the control word register of 8255. **All ports are set as input ports by default after reset.**

MODES OF OPERATION OF 8255

These are two basic modes of operation of 8255.

1) I/O mode and
2) Bit Set-Reset mode (BSR).

In I/O mode, the 8255 ports work as **programmable I/O ports**, while in **BSR mode** only **port C** (PC0-PC7) can be **used to set or reset its individual port bits.** Under the **I/O mode** of operation, further there are **three modes of operation of 8255**, so as to support different types of applications, mode 0, mode 1 and mode 2.

BSR MODE:

In this mode any of the 8-bits of port C can be set or reset depending on D0 of the control word. The bit to be set or reset is selected by bit select flags D3, D2 and D1 of the Control Word Register as given in table.

D3	D2	D1	Selected bits of Port C
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

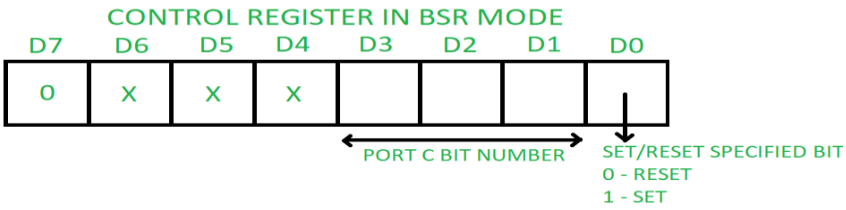
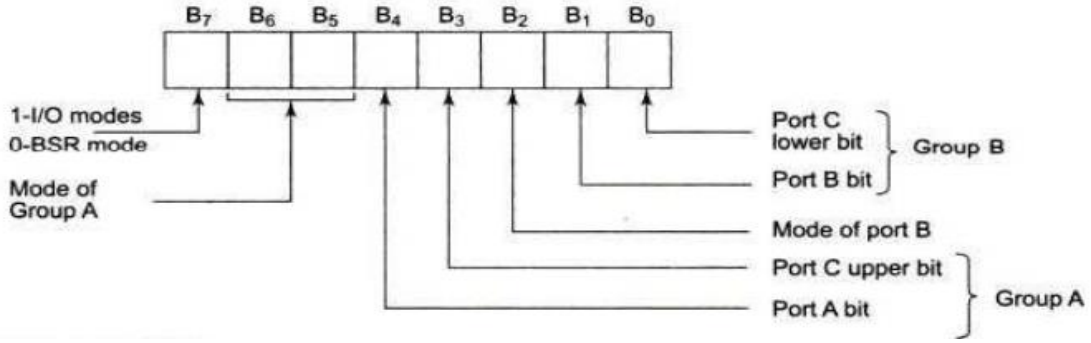


Figure: BSR mode Control Word Register format

I/O MODES:



(B6, B5: 00- Mode 0, 01- Mode 1, 1X- Mode 2)
(B2- 0- Mode 0, 1- Mode 1)

Figure: Control Word Register

PORT	MODE 0	MODE 1	MODE 2	BSR MODE
Port A	Yes	Yes	Yes	No
Port B	Yes	Yes	No	No
Port C	Yes	No [Hand Shaking]	No [Hand Shaking]	Yes

a) Mode 0 (Basic I/O mode):

This mode is also called as basic input/output mode. This mode provides simple input and output capabilities using each of the three ports. Data can be simply read from and written to the input and output ports respectively, after appropriate initialisation.

The **salient features of this mode 0** are as listed below:

1. Two 8-bit ports (port A and port B) and two 4-bit ports (port C upper and lower) are available. The two 4-bit ports can be combinedly used as a third 8-bit port.
2. Any port can be used as an input or output port.
3. Output ports are latched. Input ports are not latched.
4. A maximum of four ports are available so that overall 16 I/O configuration are possible.

All these modes can be selected by programming a register internal to 8255 known as CWR. The control word register has two formats. The first format is valid for I/O modes of operation, i.e. modes 0, mode 1 and mode 2 while the second format is valid for bit set/reset (BSR) mode of operation. These formats are shown in following figure.

b) Mode 1: (Strobed input/output mode):

In this mode the handshaking control the input and output action of the specified port. Port C lines PC0-PC2, provide strobe or handshake lines for port B. This group which includes port B and PC0-PC2 is called as group B for Strobed data input/output. Port C lines PC3-PC5 provide strobe lines for port A. This group including port A and PC3-PC5 from group A. Thus port C is utilized for generating handshake signals.

The **salient features of mode 1** are listed as follows:

1. Two groups – group A and group B are available for strobed data transfer.
2. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
3. The 8-bit data port can be either used as input and output port. The inputs and outputs both are latched.
4. Out of 8-bit port C, PC0-PC2 are used to generate control signals for port B and PC3-PC5 are used to generate control signals for port A. the lines PC6, PC7 may be used as independent data lines.

c) Mode 2 (Strobed bidirectional I/O)

This mode of operation of 8255 is also called as strobed bidirectional I/O. This mode of operation provides 8255 with an additional feature for communicating with a peripheral device on an 8-bit data bus. Handshaking signals are provided to maintain proper data flow and synchronization between the data transmitter and receiver. The interrupt generation and other functions are similar to mode 1.

In this mode, 8255 is a bidirectional 8-bit port with handshake signals. The RD and WR signals decide whether the 8255 is going to operate as an input port or output port.

The **Salient features of Mode 2 of 8255** are listed as follows:

1. The single 8-bit port in group A is available.
2. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
3. Three I/O lines are available at port C (PC2 – PC0)
4. Inputs and outputs are both latched.
5. The 5-bit control port C (PC3-PC7) is used for generating / accepting handshake signals for the 8-bit data transfer on port A.

INTERRUPTS and INTERRUPT SERVICE ROUTINES

The dictionary meaning of the word **interrupt** is to break the sequence of operation. While the CPU is executing a program, an interrupt breaks the normal sequence of execution of instructions, diverts its execution to some other program called **Interrupt Service Routine (ISR).** After executing ISR, the control is transferred back again to the main program which was being executed at the time of interruption.

Whenever a number of devices interrupt a CPU at a time, and if the processor is able to handle them properly, it is said to have **multiple interrupt processing capability.** For example, 8085 has five hardware interrupt pins and is able to handle the interrupts simultaneously under the control of software. In case of 8086, there are two interrupt pins, i.e. NMI and INTR. The **NMI is a non-maskable interrupt** input pin which means that any interrupt request at NMI input cannot be masked or disabled by any means. The **INTR interrupt may be masked** using the Interrupt flag (IF). The **INTR is of 256 types.** The INTR types may be from 00 to FFH (or 0 to 255).

If **more than one type of INTR interrupt occurs at a time**, then an external chip called **programmable interrupt controller** is required to handle them. The same is the case for INTR interrupt input of 8085. **Interrupt service routines (ISRs)** are the program to be executed by interrupting the main program execution of the CPU, after an interrupt request appears. After the execution of ISR, the main program continues its execution further from the point at which it was interrupted.

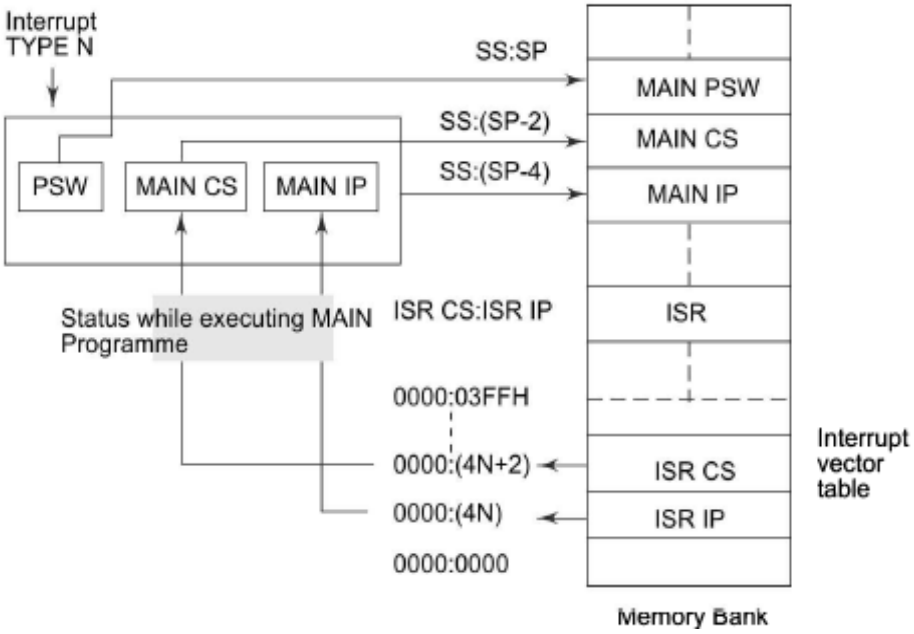
INTERRUPT CYCLE OF 8086

Interrupts are broadly classified into two types. The first out of them is **external interrupt** and the second is **internal interrupt**. In external interrupt, an **external device** or a signal interrupts from the outside or in other words, the interrupt is generated outside the processor, for example, a keyboard interrupt. The internal interrupt is **generated by the processor circuit**, or by the execution of an **interrupt instruction.** The examples of this type are divide by zero interrupt, overflow interrupt, interrupts due to INT instructions, etc.

Suppose an external device interrupts the CPU at the interrupt pin, either NMI or INTR of the 8086, while the CPU is executing an instruction of a program. The CPU first completes the execution of the current instruction. The instruction pointer (IP) is incremented to point to the next instruction. **The CPU then acknowledges the requesting device on its INTA' pin immediately if it is a NMI, TRAP or divide by zero interrupt**. **If it is an INT request (INTR), the CPU checks the IF flag.** If the IF flag is set, the interrupt request is acknowledged using the INTA' pin. If the IF is not set, the interrupt requests are ignored. Note that the responses to the NMI, TRAP and divide by zero interrupt requests are independent of the IF flag.

After an interrupt is acknowledged, the CPU computes the vector address from the type of interrupt that may be passed to the interrupt structure of the CPU internally (in case of software interrupts, NMI, TRAP and divide by zero interrupts) or externally, i.e. from an interrupt controller in case of external interrupts. (**The contents of IP and CS are next pushed to the stack. The contents of IP and CS now point to the address of the next instruction of the main program from which the execution is to be continued after executing the ISR. The PSW is also pushed to the stack.**). **The interrupt flag (IF) is cleared. The Trap flag (TF) is also cleared**, after every response to the single step interrupt. The control is transferred to the interrupt service routine for serving the interrupting device. The new address of ISR is found out from the interrupt vector table (IVT). The execution of ISR starts.

If further interrupts are to be responded to during the time the first interrupt is being serviced, the IF should again be set to 1 by the ISR of the first interrupt. If the interrupt flag is not set, the subsequent interrupt signals will not be acknowledged by the processor, till the current one is completed. The programmable interrupt controller is used for managing such multiple interrupts based on their priorities. At the end of ISR the last instruction should be IRET. When the CPU executes IRET, the contents of flags, IP and CS which were saved at the start by the CALL instruction are now retrieved to the respective registers. The execution continues onwards from this address received by IP and CS.



Or

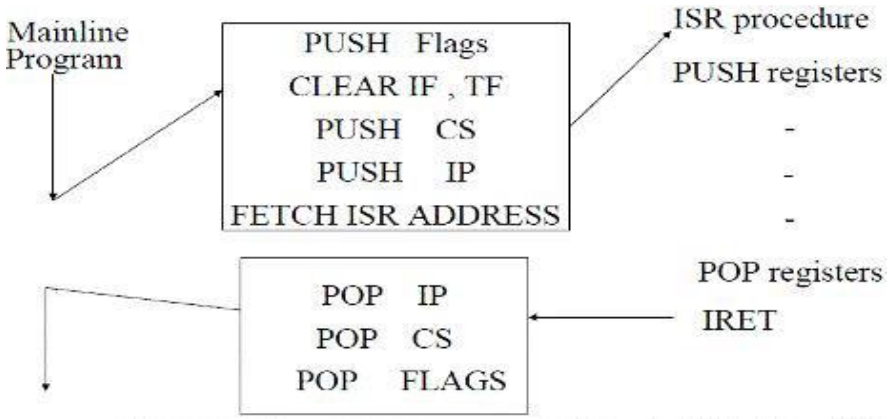


Figure: Interrupt response sequence

Here, we discuss how the 8086 finds out the address of an ISR. Every external and internal interrupt is assigned with a type (N) that is either implicit (in case of NMI, TRAP and divide by zero) or specified in the instruction INT N (in case of internal interrupts). In case of external interrupts, the type is passed to the processor by an external hardware like programmable interrupt controller. In the zeroth segment of physical address space, i.e. CS = 0000, Intel has reserved 1024 locations for storing the interrupt vector table. The 8086 supports a total of 256 types of interrupts, i.e. from 00H to FFH. **Each interrupt requires 4 bytes**, two bytes each for IP and CS of its ISR. Thus a total of 1024 bytes are required for 256 interrupt types, hence the interrupt vector table starts at location 0000:0000 and ends at 0000: 03FFH.

The interrupt vector table contains the IP and CS of all the interrupt types stored sequentially from address 0000:0000 to 0000:03FFH. The interrupt type N is multiplied by 4 and the hexadecimal multiplication is obtained gives the offset address in the zeroth code segment at which the IP and CS addresses of the interrupt service routine are stored. The execution automatically starts from the new CS: IP.

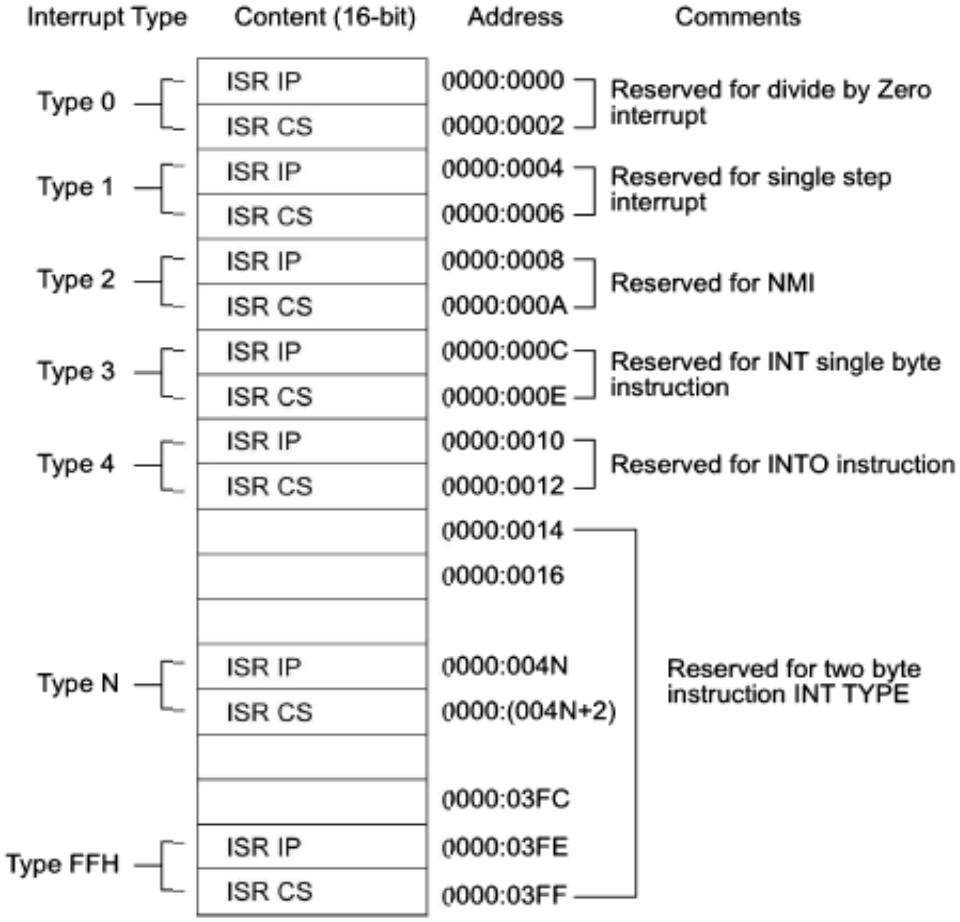


Figure: Structure of Interrupt Vector Table of 8086

8086 INTERRUPT TYPES:

Divide-by-Zero Interrupt-Type 0:

The 8086 will automatically do a type 0 interrupt if the **result of a DIV operation or an IDIV operation is too large to fit in the destination register.** For a type 0 interrupt, the 8086 pushes the flag register on the stack, resets IF and TF and pushes the return addresses on the stack.

Single step Interrupt-Type 1:

The use of single step feature found in some monitor programs and debugger programs. When you tell a system to single step, it will execute one instruction and stop. If they are correct we can tell a system to single step, it will execute one instruction and stop. We can then examine the contents of registers and memory locations. In other words, when in **single step mode a system** will stop after it executes each instruction and wait for further direction from you. The 8086 trap flag and type 1 interrupt response make it quite easy to implement a single step feature direction.

Non-maskable Interrupt-Type 2:

The 8086 will automatically do a type 2 interrupt response when it **receives a low to high transition on its NMI pin**. When it does a type 2 interrupt, the 8086 will push the flags on the

stack, reset TF and IF, and push the CS value and the IP value for the next instruction on the stack. It will then get the CS value for the start of the type 2 interrupt service procedure from address 0000AH and the IP value for the start of the procedure from address 00008H.

Breakpoint Interrupt-Type 3:

The type 3 interrupt is produced by execution of the INT3 instruction. The main use of the type 3 interrupt is **to implement a breakpoint function** in a system. When we insert a breakpoint, the system executes the instructions up to the breakpoint and then goes to the breakpoint procedure. Unlike the single step which stops execution after each instruction, the breakpoint feature executes all the instructions up to the inserted breakpoint and then stops execution.

Overflow Interrupt-TYPE4:

The 8086 o**verflow flag will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented in the destination register or memory location.** For example, if you add the 8 bit signed number 01101100 and the 8 bit signed number 010111101, the result will be 10111101. This would be the correct result if we were adding unsigned binary numbers, but it is not the correct signed result.

Software Interrupts-Type 0 through 255:

The 8086 INT instruction can be used to cause the 8086 to do any one of the 256 possible interrupt types. The desired interrupt type is specified as part of the instruction. The instruction INT32, for example will cause the 8086 to do a type 32 interrupt response. The 8086 will push the flag register on the stack, reset TF and IF, and push the CS and IP values of the next instruction on the stack.

Priority of 8086 Interrupts:

If two or more interrupts occur at the same time then the highest priority interrupt will be serviced first, and then the next highest priority interrupt will be serviced. As an example suppose that the INTR input is enabled, the 8086 receives an INTR signal during the execution of a divide instruction, and the divide operation produces a divide by zero interrupt. Since the internal interrupts-such as divide error, INT, and INTO have higher priority than INTR the 8086 will do a divide error interrupt response first.

8259A PROGRAMMABLE INTERRUPT CONTROLLER

The processor 8085 had five hardware interrupt pins. Out of these five interrupt pins, four pins were allotted fixed vector address but the pin INTR was not allowed any vector address, rather an external device was supposed to hand over the type of interrupt, i.e. (Type 0 to 7 for RST0 to RST7), to the microprocessor. The microprocessor then gets this type and derives the interrupt vector address from that. The microprocessor then gets this type and derives the interrupt vector address from that. Consider an application, where a number of I/O devices connected with a CPU desire to transfer data using interrupt driven data transfer mode. In these types of applications, more number of interrupt pins are required than available in a typical microprocessor. Moreover, in these multiple interrupt systems, the processor will have to take care of the priorities for the interrupts, simultaneously occurring at the interrupt request pins.

To overcome all these difficulties, we require a programmable interrupt controller which is able to handle a number of interrupts at a time. This controller takes care of a number of simultaneously appearing interrupt requests along with their types and priorities. This relieves the processor from all these tasks. The programmable interrupt controller 8259A from Intel is one such device. Its predecessor 8259 was designed to operate only with 8-bit processors like 8085. A

modified version, 8259A was later introduced that is compatible with 8-bit as well as 16 bit processors.

ARCHITECTURE OF 8259A

The architectural block diagram of 8259A is shown below.

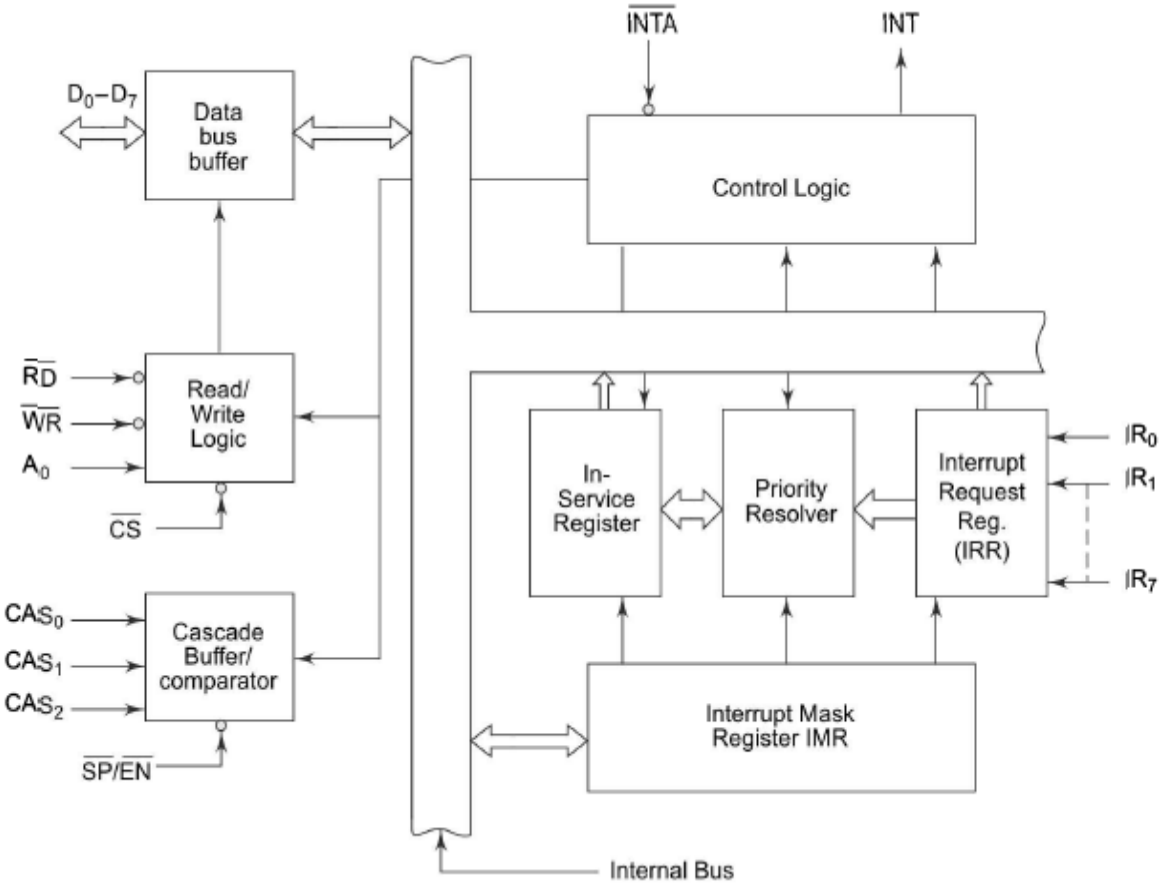


Figure: Functional block diagram of 8259A

Interrupt Request Register (IRR)

The interrupts at the IRQ input lines are handled by the Interrupt Request Register (IRR) internally. The IRR is used to store all the interrupt requests in it in order to serve them one by one on the priority basis.

In-Service Register (ISR):

This ISR is used to store all the interrupt requests those are being served, i.e. ISR keeps a track of the requests being served.

Priority Resolver:

This logic block determines the priorities of the interrupt requests appearing simultaneously. The highest priority is selected and stored into the corresponding bit of the ISR during INTA' pulse. The IR0 has the highest priority while IR7 has the lowest priority, normally in the fixed priority mode. The priorities however may be altered y programming the 8259A in the rotating priority mode.

Interrupt Mask Register (IMR):

The IMR stores the bits required to mask the interrupt inputs. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower quality.

Interrupt Control Logic:

This block manages the interrupt and the interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests. This also accepts the interrupt acknowledge (INTA) signal from the CPU that causes the 8259A to release the vector address on to the data bus.

Data Bus Buffer

This tristate, bidirectional 8-bit buffer is used to interface the 8259A to the microprocessor system data bus. Control words, status and vector information pass through the data bus buffer during read or write operation.

Read/Write Control Logic

This circuit accepts and decodes commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the data bus.

Cascade Buffer/Comparator

This function block stores and compares the IDs of all 8259A's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259A is used as a master and are inputs when the 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected, will send its pre-programmed subroutine address (vector address) onto the data bus during the next one or two consecutive INTA' pulses.

PIN DIAGRAM OF 8259A

The pin configuration of 8259A and functional description of each of the pin are given below:

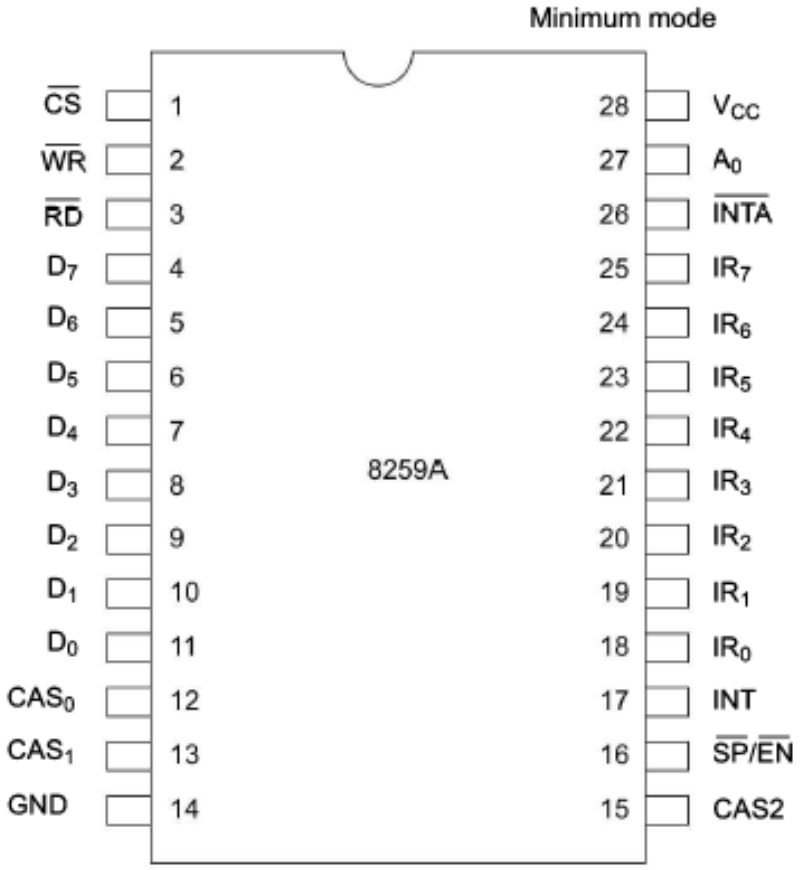


Figure: 8259A pin diagram

Vcc: SUPPLY: 5V Supply.

GND: GROUND

CS': CHIP SELECT: A low on this pin enables RD' and WR' communication between the CPU and the 8259A. INTA' functions are independent of CS'.
WR': WRITE: A low on this pin when CS' is low enables the 8259A to accept command words from the CPU.

RD': READ: A low on this pin when CS' is low enables the 8259A to release status onto the data bus for the CPU.

D7-D0: I/O BIDIRECTIONAL DATA BUS: These pins form a bidirectional data bus that carries 8-bit data either to control word or from status word registers. This also carries interrupt-vector information is transferred via this bus.

CAS0-CAS2: I/O CASCADE LINES: A single 8259A provides eight vectored interrupts. If more interrupts are required, the 8259A is used in the cascade mode in which a master 8259A along with eight slaves 8259A can provide upto 64 vectored interrupt lines. These three lines act as select lines for addressing the slaves 8259A.

SP'/EN': SLAVE PROGRAM/ENABLE BUFFER: This is a dual function pin. When the chip is used in the buffered mode, it can be used as an output to control buffer transceivers (EN). If this is not used in the buffered mode then the pin is used as input to designate whether the chip is used as a master (SP'= 1) or slave (SP'= 0).

INT: INTERRUPT: This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.

IR0-IR7: INTERRUPT REQUESTS: These pins act as inputs to accept interrupt requests to the CPU. In the **edge triggered mode,** an interrupt service is requested by raising an IR pin from a low to high state. If it is held high until it is acknowledged, and just by latching it to high level, if used in **level triggered mode**.

INTA': INTERRUPT ACKNOWLEDGE: This pin is used to enable in 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.

A0: ADDRESS LINE: This pin acts in conjunction with the CS', WR', and RD' pins. It is used by the 8259A to decode various command words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086, 8088).

The device 8259A can be interfaced with any CPU using either polling or interrupt. In **polling**, the CPU keeps on checking each peripheral device in sequence to ascertain if it requires any service from the CPU. If any such service is noticed, the CPU serves the request and then goes to the next device in sequence. After all the peripheral devices are scanned as above the CPU again starts from the first device. The type of system operation **results in the reduction of processing speed** because most of the CPU time is consumed in polling the peripheral devices.

In the **interrupt driven method,** the CPU performs the main processing task till it is interrupted by a service requesting peripheral device. The net **processing speed of these type of systems is high** because the CPU serves the peripheral only if it receives the interrupt request. If

more than one interrupt requests are received at a time, all the requesting peripherals are served one by one on priority basis. This method of interfacing may require additional hardware if the number of peripherals to be interfaced is more than the interrupt pins available with the CPU.

INTERRUPT SEQUENCE IN AN 8086 SYSTEM:

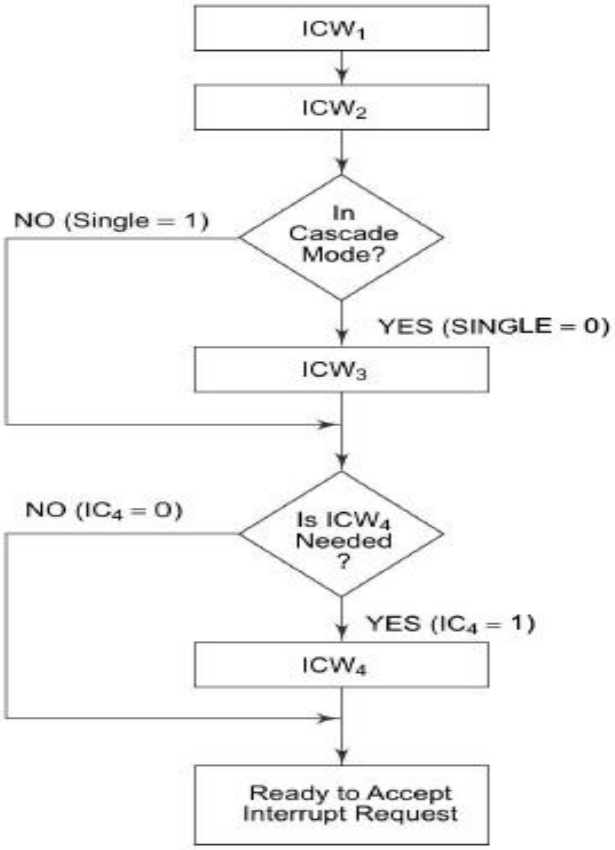


Figure: Initialization sequence of 8259A

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

The events occur as follows in an 8086-8259A system is described as follows:

1. One or more of the Interrupt request lines (IR7-0) are raised high, setting the corresponding IRR bit(s).

2. The 8259A evaluates these requests, resolves priority, and sends an INT signal to the CPU, if appropriate.

3. The CPU acknowledges the INT and responds with an INTA' pulse.

4. Up on receiving an INTA' from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the data bus during this cycle.

5. The 8086 will initiate a second INTA' pulse. During this pulse, the 8259A releases an 8-bit pointer onto the data bus where it is read by the CPU.

6. This completes the interrupt cycle. In the AEOI mode, the ISR bit is reset at the end of the second INTA' pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

(**Note:** If no interrupt request is present at step 4 (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectoring bytes and the CAS lines will look like an interrupt level 7 was requested.)

COMMAND WORDS OF 8259A

The command words of 8529A are classified in two groups, i.e. Initialization Command Words (ICWs) and Operation Command Words (OCWs).

INITIALIZATION COMMAND WORDS (ICWS)

Before it starts functioning, the 8259A must be initialized by writing two to four command words into the respective command word registers. These are called Initialization Command Words (ICWs). Whenever a command is issued with **A0 = 0 and D4 = 1**, this is interpreted as **Initialization Command Word 1 (ICW1).** It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval and whether ICW4 is required or not, etc. If **A0 = 1,** the command word is recognized as **ICW2.** ICW2 stores the details regarding interrupt vector addresses.

Once ICW1 is loaded, the following initialization procedure is carried out internally.

a. The edge sense circuit is reset, i.e. by default 8259A interrupts are edge sensitive.

b. The Interrupt Mask Register is cleared.

c. IR7 input is assigned the lowest priority.

d. The slave mode address is set to 7.

e. Special Mask Mode is cleared and Status Read is set to IRR.

f. If IC4 = 0, then all functions of ICW4 are set to zero. Master/Slave in ICW4 is only used in the buffered mode.

Initialization Command Words:

In an 8085 based system A15-A8 of the interrupt vector address are the respective bits of ICW2. In 8086 based system, five most significant bits of the interrupt type byte are inserted in place of T7-T3 respectively and the remaining three bits (A8, A9 and A10) are inserted internally as 000 (as they are pointing to IR0). Other seven interrupt levels vector addresses are internally generated automatically by 8259 using IR0 vector. Address interval is always four in an 8086 based system.

ICW1 and ICW2 are compulsory command words in initialization sequence of 8259A. While ICW3 and ICW4 are optional. The ICW3 is read only when there are more than one 8259A's in the system, i.e. cascading is used (SNGL=0). The SNGL bit in ICW1 indicates whether the 8259A is in the cascade mode or not. The ICW3 loads an 8-bit slave register.

The functions of the ICW3 register are:

a. In the master mode (either when SP'=1, or in buffered mode when M/S = 1 in ICW4) the 8 bit slave register will be set bit-wise to '1' for each slave in the system. The requesting slave will then release the second byte of a CALL sequence.

b. In the slave mode (either when SP'= 0, or if BUF = 1 and M/S = 0 in ICW4) bits D2 to D0 identify the slave, i.e. 000 to 111 for slave1 to slave8. The slave compares its cascade input with these bits and, if they are equal, the second byte of the CALL sequence is released by it on the data bus.

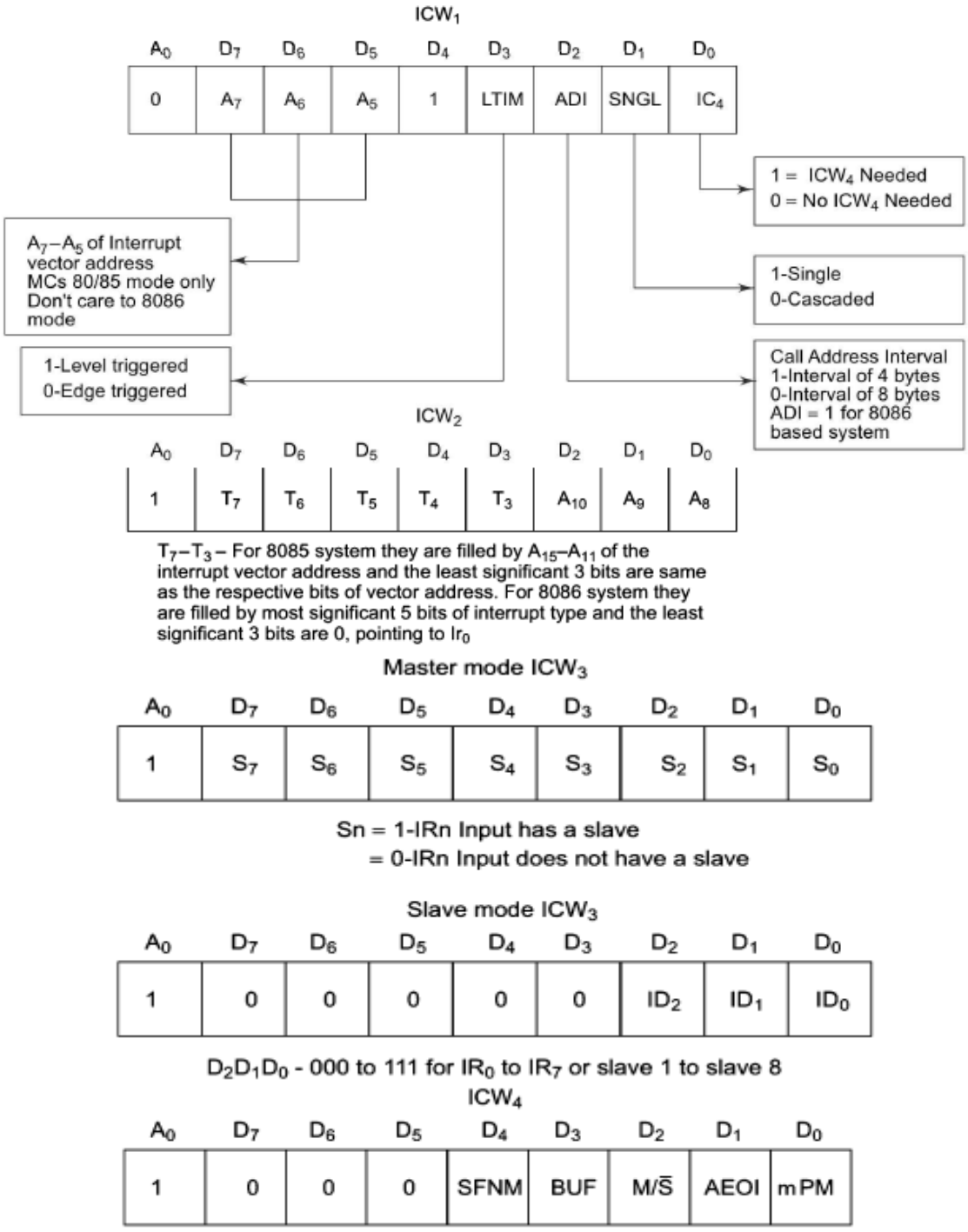


Figure: Initialization command words (ICWs)

ICW4: The use of this command word depends on the IC4 bit of ICW1. If IC4=1, ICW4 is used, otherwise it is neglected. The bit functions of ICW4 are described as follows:

SFNM: Special Fully Nested Mode is selected, if SFNM =1.

BUF: If BUF =1, the buffered mode is programmed. In buffered mode, SP/EN becomes an enable output and the master/ slave determination is by M/S.

M/S: If buffered mode is selected: M/S =1 means the 8259A is programmed to be a master,
M/S =0 means the 8259A is programmed to be a slave.
If BUF= 0, M/S has no function.

AEOI: If AEOI =1, the automatic end of interrupt mode is programmed.

mPM: Microprocessor mode: mPM = 0 sets the 8259A for MCS-8085 system operation,
mPM = 1 sets the 8259A for 8086 system operation.

OPERATION COMMAND WORDS:

Once 8259A is initialized using ICWs, it is ready for its normal functions, i.e. for accepting the interrupts but 8259A has its own ways of handling the received interrupts called **modes of operation**. These modes of operation can be selected by programing, i.e. writing the three internal registers called **operation command word registers**. The data written into them (bit pattern) is called **operation command words.**

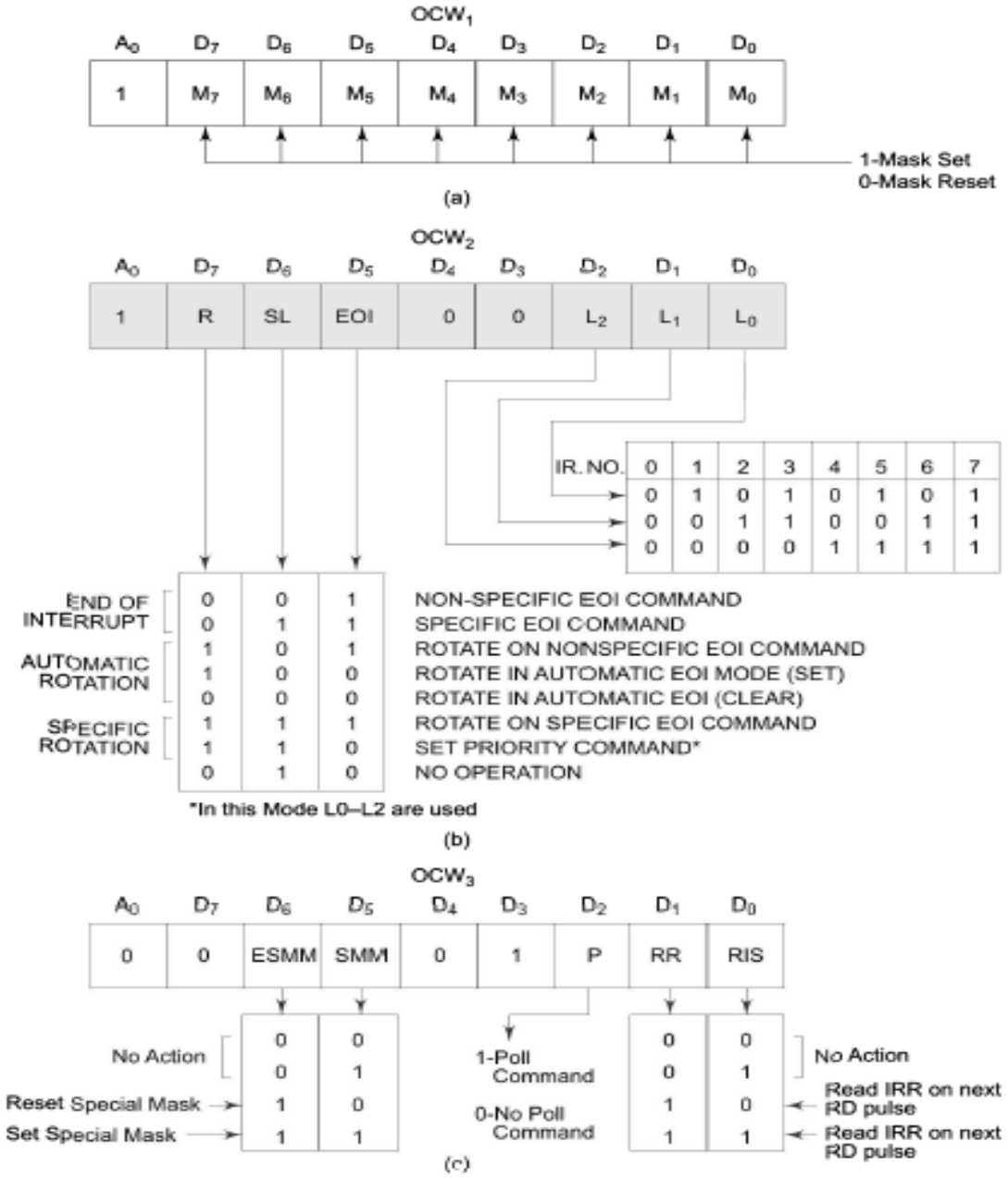


Figure: Operation command words (OCWs)

In the three operation command words OCW1, OCW2 and OCW3, every bit corresponds to some operational feature of the mode selected, except for a few bits those are either 1 or 0.

Operation Control Word 1 (OCW1)

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). M7-M0 represent the eight mask bits. M = 1 indicates the channel is masked (inhibited), M = 0 indicates the channel is enabled.

Operation Control Word 2 (OCW2)

R, SL, EOI -These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

L2, L1, L0-These bits determine the interrupt level acted upon when the SL bit is active.

Operation Control Word 3 (OCW3)

ESMM - Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes don't care.

SMM - Special Mask Mode. If ESMM = 1 and SMM = 1 the 8259A will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 8259A will revert to normal mask mode. When ESMM = 0, SMM has no effect.

Operating modes of 8259:

The different modes of operation of 8259A can be programmed by setting or resetting the appropriate bits of the ICWs or OCWs. The different modes of operation of 8259A are given below:

Fully Nested Mode

This is **the default mode** of operation of 8259A. **IR0** has the **highest** priority and **IR7** has the **lowest** one. When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command just before returning from the service routine, or the AEOI (Automatic End of Interrupt) bit is set. If the ISR (In service) bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt, that will be acknowledged only if the microprocessor's Interrupt enable flag (IF) is set. The priorities can be changed by programming in the rotating priority modes.

End of Interrupt (EOI)

The In Service Register (ISR) bit can be reset either with AEOI bit in ICW1 or by EOI command, issued before returning from the interrupt service routine. There are **two forms of EOI command**: **Specific and Non-Specific**. When the 8259A is operated in modes which preserve the fully nested structure, it can determine which ISR bit to reset on EOI. When a Non- Specific EOI command is issued the 8259A it will automatically reset the highest ISR bit of those that are set.

When a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt command must be issued to reset a particular ISR bit. An ISR bit that is masked by an IMR bit, will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

Automatic End of Interrupt (AEOI) Mode:

If **AEOI =1 in ICW4**, then the 8259A will operate in AEOI mode continuously until reprogrammed by ICW4. In this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse. This **mode should be used only when a nested multilevel interrupt structure is not required within a single 8259A.**

Automatic Rotation (Equal Priority Devices)

In some applications there are a **number of interrupting devices of equal priority**. In this mode, an Interrupt Request (IR) level receives lowest priority after it is served while the next device to be served gets the highest priority in sequence. Once all the devices are served like this, the first device again receives highest priority.

Specific Rotation (Specific Priority)

In this mode a **bottom priority level can be selected using L2, L1 and L0 in OCW2 and R= 1, SL= 1, EOI =1.** The selected bottom priority fixes all other priorities. i.e., **if IR5 is programmed as the bottom priority device**, then IR5 will have the least priority and IR4 will have the next higher priority. Thus **IR6 will have the highest priority.** These priorities can be changed during an EOI command by programming the rotate on specific EOI command in OCW2.

Interrupt Masks

Each Interrupt Request input can be masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

Special Mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. In the **special Mask Mode**, when a mask bit is set in OCW1, **it inhibits further interrupts at that level** and **enables interrupts from all other levels (lower as well as higher) that are not masked.**

Edge and Level Triggered Modes

This mode decides whether the interrupt should be edge triggered or level triggered. If bit LTIM of ICW1 = 0, they are edge triggered, otherwise level triggered.

Reading 8259 Status

The **status of the internal registers of 8259A can be read using this mode**. The OCW3 is used to read IRR and ISR while OCW1 is used to read IMR. Reading is possible only in no polled mode.

Poll Command

In the **polled mode of operation, the INT output of 8259A is neglected, though it functions normally**, by not connecting the INT output or by masking INT input of the microprocessor. The Poll mode is entered by setting P =1 in OCW3. The 8259A is polled by using software execution by microprocessor instead of the requests on INT input. The 8259A treats the next RD' pulse to the 8259A as an interrupt acknowledge. An appropriate ISR bit is set, if there is a request. The priority

level is read and a data word is placed on to data bus, after RD' is activated. The data word is shown below:

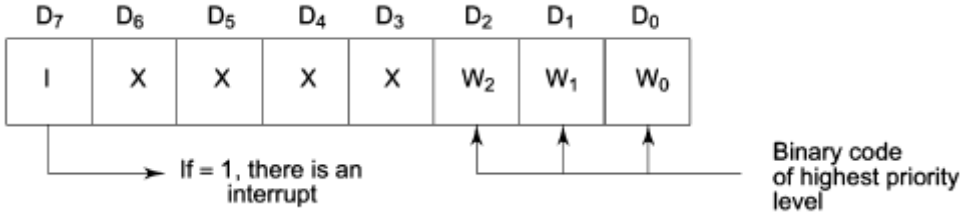


Figure: Data word of 8259

A poll command may give you more than 64 priority levels. This poll mode has nothing to do with the 8086 interrupt structure and the interrupt priorities.

Special Fully Nest Mode

This mode is used in more complicated systems, where cascading is used, and the priority has to be programmed in the master using ICW4. This is somewhat similar to the normal nested mode. In this mode, when an interrupt request from a certain slave is in service, this slave can further sends requests to the master, if the requesting device connected to the slave has higher priority than the one being currently served. In this mode, the master interrupts the CPU only when the interrupting device has a higher or the same priority than the one currently being served. In normal mode, other requests than the one being served are masked out.

When entering the interrupt service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent. This mode is important, since the absence of this mode, the slave would interrupt the master only once and hence the priorities of the slave inputs would have been disturbed.

Buffered Mode

When the 8259A is used in a large system in which bus driving buffers are used on the data buses (e.g. cascading systems), the problem of enabling the buffers arises. The 8259A sends a buffer enable signal on SP'/EN' pin, whenever data is placed on the bus.

Cascade Mode

The 8259A can be easily interconnected in a system containing one master with up to eight slaves to **handle up to 64 priority levels**. The master controls the slaves through the 3 line cascade bus (CAS0- CAS2). The **cascade bus acts like chip selects to the slaves** during the INTA sequence. In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the vector address during second pulse of INTA' sequence.

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first INTA' pulse to the trailing edge of the second INTA' pulse. Each 8259A in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the

corresponding slave. An address decoder is required to activate the Chip Select (CS) input of each 8259A.

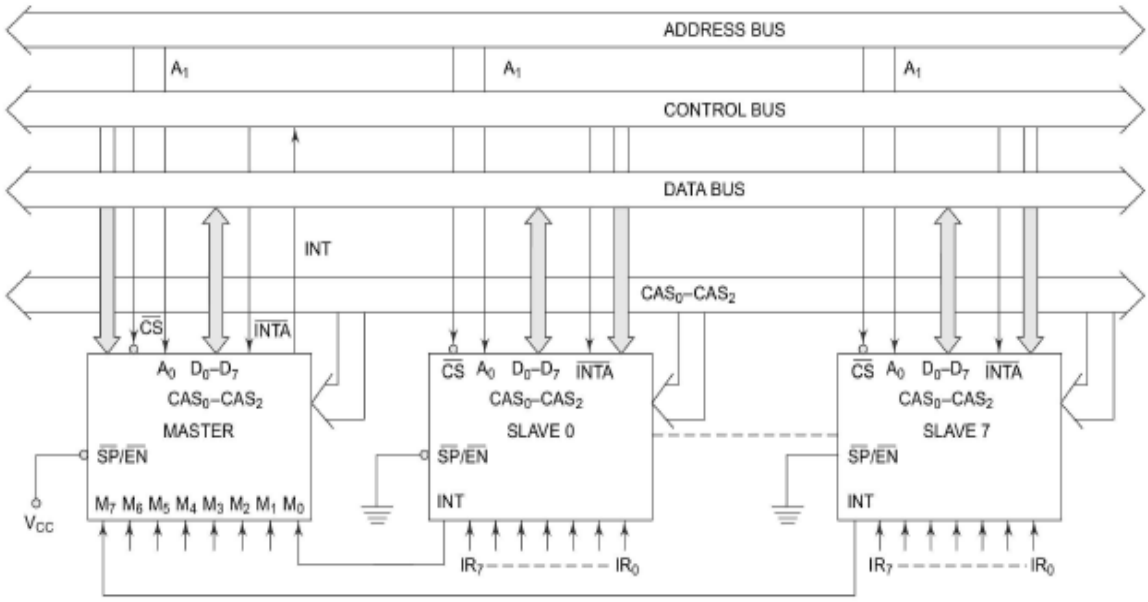


Figure: 8259A in cascade mode

Example: Interfacing Program

Problem: Show 8259A interfacing connections with 8086 at the address 074X. Write an ALP to initialize the 8259A in single level triggered mode. Then set the 8259A to operate with IR6 masked, IR4 as bottom priority level, with special EOI mode. Set special mask mode of 8259A. Read IRR and ISR into registers BH and BL respectively. IR0 of 8259A will have type 80H.

Solution:

Let the starting address is 0000:0200H (80*4 in segment 0000H). The interconnections of 8259A with 8086 is shown below.

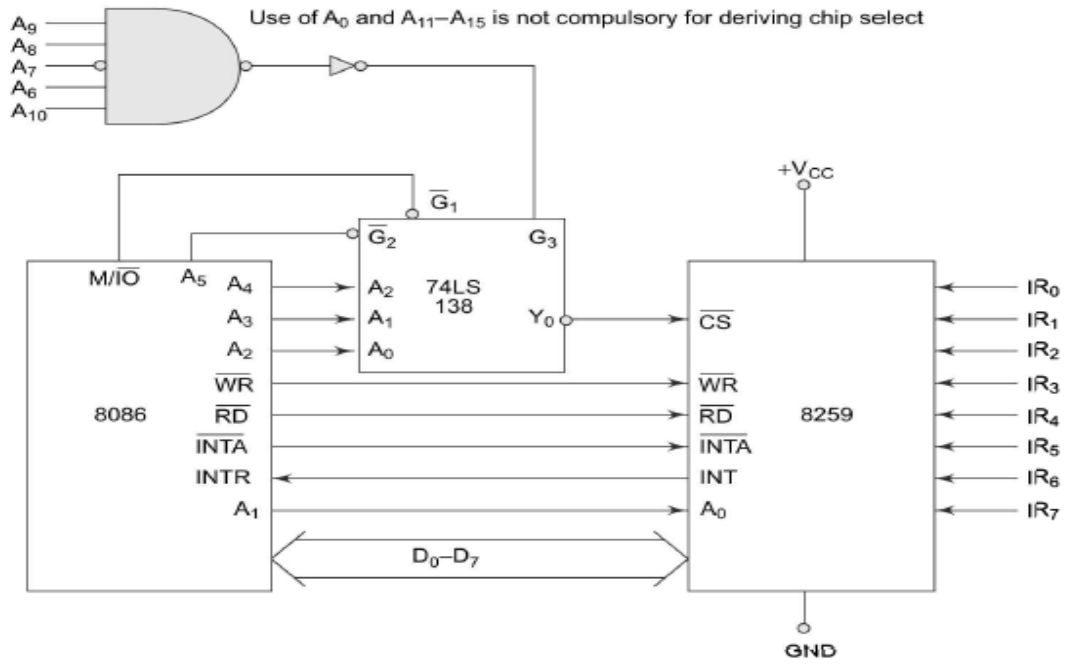


Figure: Interfacing of 8259A with 8086

The 8259 is interfaced with lower byte of the 8086 data bus, hence A0 line of the microprocessor system is abandoned, and A1 of the microprocessor system is connected with A0 of the 8259A. Before going for ALP, all the initialization command words (ICWs) and operation command words (OCWs) must be detected. ICW1 decides single level triggered, address interval of 4 as below:

ICW1:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	1	1	1	1	1	= 1FH

D_0 ICW_4 Needed
D_1 Single 8259A
D_2 Call Address Interval 4
D_3 Level Triggered
D_4 Always set to 1
D_5 D_6 D_7 Don't care for 8086 system
A_0 Always set to 0

ICW2: Vector address = 0000:0010 for IR3

T7	T6	T5	T4	T3	A10	A9	A8	
1	0	0	0	0	0	1	1	=83H

A_8 A_9 A_{10} IR3 selected

There is no slave hence **ICW3** is given below:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	0	ICW_3=00H

Actually ICW3 is not at all needed, because ICW1 in the 8259A is set for single mode.

ICW4:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	1	ICW_4=01H

D0 For 8086 system
D1 Normal EOI
D2 D3 Non buffered mode
D4 For special fully nested mode masking

OCW1: Sets the mask of IR6 as shown below:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	1	0	0	0	0	0	0	OCW_1=40H

IR6 is masked.

OCW2: Sets the modes and rotating priority as shown below:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	0	0	1	0	0	OCW_2=E4H

D0 D2 Bottom priority Level set at IR4
D5 D7 Specific EOI Command with rotating priority

OCW3: Sets the special mask mode and reads ISR and IRR using the following control words.

For reading IRR

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	1	0	1	0	1	0

OCW_3=6AH

D0 D1 Read IRR
D2 No Poll command
D5 D7 Special mask mode

For reading ISR

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	1	0	1	0	1	1

OCW_3=6BH

D0 D1 Read ISR
D2 No Poll command
D5 D7 Special mask mode

Assembly language program

```
ASSUME CS: CODE
CODE SEGMENT
START:  MOV AL,1FH          ; Set the 8259A in single, level
        MOV DX,0740H
        OUT DX,AL           ; triggered mode with call
        MOV DX,0742H        ; address of interval of 4
        MOV AL,83H          ; Select vector address 0010H
        OUT DX,AL           ; for IR3(ICW2)
        MOV AL,01H          ; ICW4 for 8086 system, normal
        OUT DX,AL           ; EOI, non-buffered, SFNM masked
        MOV AL,40H
        OUT DX,AL           ; OCW1 for IR6 masked
        MOV AL,E4H          ; Specific EOI with rotating
        MOV DX,0740H
        OUT DX,AL           ; Priority and bottom level of
        MOV AL,6AH          ; IR4 with OCW2 Write OCW3 reading
        OUT DX,AL           ; IRR and store in BH
        IN AL,DX
        MOV BH,AL
        MOV AL,6BH          ; Write OCW3 to read
        OUT DX,AL           ; ISR and store in BL
        IN AL,DX
        MOV BL,AL
        MOV AH,4CH          ; Return to DOS
        INT 21H             ; highlighted instructions can be replaced with INT 03H
CODE ENDS
END START
```