Lecture-40

INPUT/OUTPUT Techniques

In all process content applications, μp would like to communicate with different I/O devices for data transfers, i.e., transfer of data between circuitry external to the microprocessor and the processor itself. This transfer of data is in addition to transfers between the microprocessor and memory and is referred to as input/output or I/O.

In addition to the memory interfaced to the microprocessor many input & output peripheral devices may have to be interfaced with the system to obtain a micro-computer depending upon particular control applications. Some of the input devices interfaced are push-bottom switches, toggle switches, keyboard, TTY, CRT, analog to digital converters (ADC) etc. Some of the output devices interfaced with the system are LEDs, seven segment displays, TTY, CRT, printers, digital to analog converters (DAC) etc.

The techniques used to transfer data between processor and I/O devices are called I/O techniques. These I/O techniques are mainly classified in two types - CPU initiated and device initiated I/O data transfer.

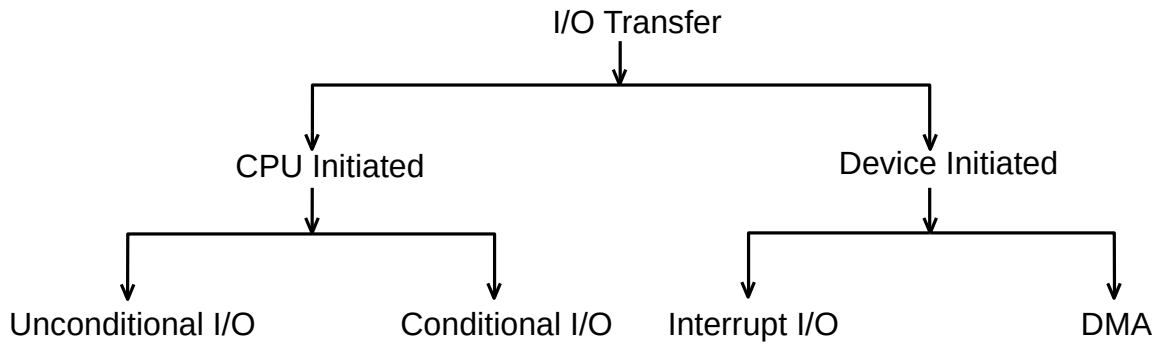

Fig.7.1 I/O Transfer Techniques

In the case of **CPU initiated I/O transfer**, it is the μp which through a software initiates the action of data transfer. In the case of device initiated it is the external device which is interested in communication with CPU initiates the data transfer operation.

The CPU initiated I/O transfer is also known as program controlled I/O, because the transfer of data is completely under the control of the microprocessor program. An I/O operation takes place only when an I/O transfer instruction is encountered in the execution of the program. This I/O transfer may be unconditional or conditional I/O transfer. In the case of **CPU initiated unconditional I/O transfer** CPU initiates the data transfer and assumes that the external device is always ready to input a data into the μp at the time of data transfer initiated by the μp and, therefore, microprocessor affects the transfer of data through corresponding instructions. Toggle switch is an example of such input device the data from which is read as and when processor initiates data transfer operations. It is always ready for data transfer. In the case of an output data transfer the processor assumes that the output device is already to accept data and sends the necessary data through the corresponding instructions.

There are I/O devices such as ADC, printer etc., which are not always ready for data transfer. These devices take some time to get ready for data transfer and, therefore, **CPU initiated polled I/O transfer** is used for data transfer. In such cases, again it is the CPU that initiates the data transfer operation but it does not assume that the peripheral device is ready for data transfer. It checks the readiness of the I/O device before the data transfer occurs. This involves testing one or more status flags or bits associated with the

I/O device before the data transfer. This is known as hand shake control signals. The flow chart for CPU initiated polled I/O transfer is shown in fig.7.2.

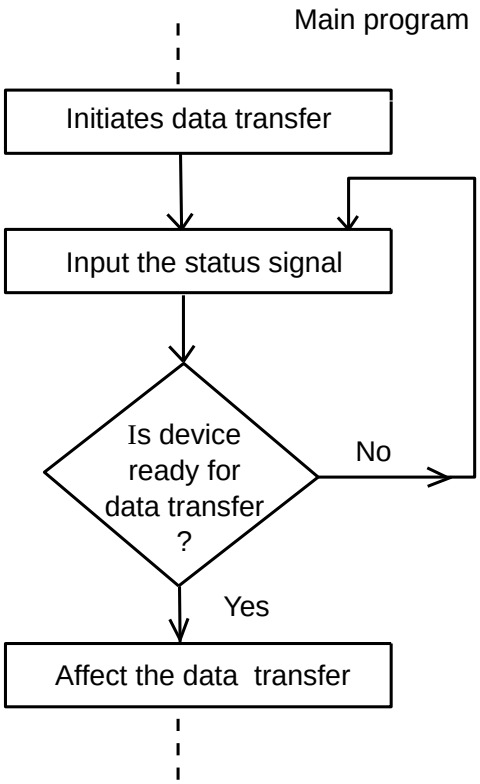

Fig.7.2 Conditional I/O Transfer

CPU initiated polled I/O transfer can be preferred if and only if the μp has no other useful work to do in the process control under consideration and can wait till the device is ready for data transfer. On the other hand, in some applications the wait away time is large enough like 1sec or 1min and the process control application under consideration demands some useful work to be done by the microprocessor. In such cases, CPU initiated polled I/O transfer shall be dispensed with and device initiated I/O transfer should be preferred.

Device initiated I/O transfer is one which the device initiates the transfer of data. There are two main types of device initiated I/O data transfer – interrupt driven and DMA.

In the case of **device initiated interrupt I/O transfer** (also known as interrupt program controlled I/O), an external device indicates directly to the μp its readiness to transfer data by making the corresponding interrupt control signal active. Most μp interrupt inputs can be disabled under program controlled I/O. When a microprocessor program is interrupted, μp recognizes the fact that the external device has requested for the data transfer & control is transferred to an interrupt service subroutine. This subroutine performs the data transfer and then returns the control to the program at the point it was interrupted, and processing continues. Thus, with an interrupt program controlled I/o operation, the data transfer is requested by an external device and then implemented by an interrupt service subroutine.

DMA stands for direct memory access. It is also referred to as hardware controlled I/O. It is the only type of I/O transfer in which the μp is not involved in the transfer of data between the memory and the external device. In this case there is direct data transfer between an I/O device and memory; data is not transferred from an I/O device to one of the μp registers and then to memory or vice-versa. In DMA, μp goes to the HOLD state. The address bus, data bus and the data communication control signal like $\overline{WR}$, $\overline{RD}$, IO/$\overline{M}$ are all tri-stated and the external device which initiates the DMA through making HOLD control signal active, takes the command of the address bus, data bus and the control signals for data transfer applications between the

memory & device. DMA is used primarily to transfer a number of words or block of data at high speed.

As discussed earlier, input port and output port are basically external registers. The IO/$\overline{M}$ control signal of 8085A determines whether the address generated during a data transfer process refers to memory (IO/$\overline{M}$ =0) or to the separate I/O address space (IO/$\overline{M}$ =1). I/O address space is further divided by control strobes into an input address space ($\overline{RD}$ = 0) and an output address space ($\overline{WR}$ = 0).

Fig. shows how the address space of 8085A is partitioned by these control signals. When I/O ports are assigned separate address space, distinct from the address space of the external registers that comprise main memory, they are referred to as isolated I/O or standard I/O.

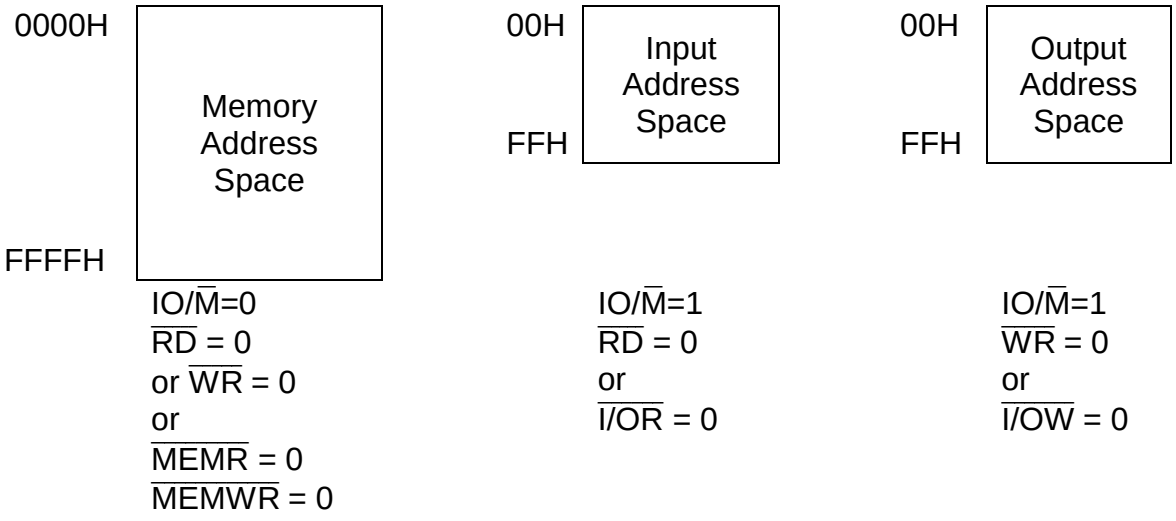

Fig.7.3 Isolate I/O Space or Standard I/O

Only the IN and OUT instructions provide data transfer for isolated I/O. IN PORT and OUT PORT instructions require three machine cycles for execution. The first is, of course, an OPCODE FETCH machine cycle. The second is MEMORY READ machine cycle during

which the 8-bit port address is transferred from memory to the processor and placed in both the (W) & (Z) temporary registers (as discussed earlier) and the third is either an I/O READ on I/O WRITE machine cycle during which the actual data transfer from or to the I/O device takes place.

During the I/O READ and I/O WRITE machine cycles, the 8-bit port address, duplicated in (W) & (Z) registers, is outputted from the processor 8085A on address lines A_{15}-A_8 and address data bus lines AD_7-AD_0. The read ($\overline{RD}$) and write ($\overline{WR}$) control strobes from the 8085A specify the exact time at which an input port's tri-state buffer is enabled to drive the data bus or the exact time at which an output port the data placed on the data bus by the processor respectively.

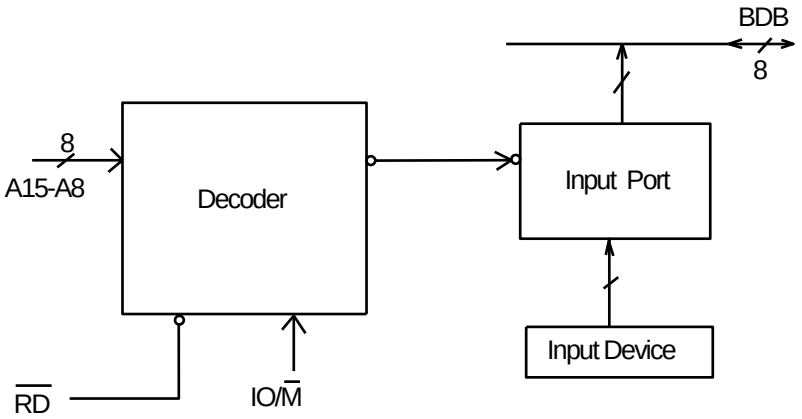

Fig.7.4 Interfacing Input Device using Isolated I/O

For input ports, external decoding logic combines $\overline{RD}$, IO/$\overline{M}$, and the port address and generates a unique input device select pulse for each input port as shown in fig.7.4. The pulse occurs only during the I/O READ machine cycle of an IN PORT instruction that addresses the specific port. The input device select pulse enables the input port's tri-state buffer. If through design or program error, the tri-state buffers of two or more ports or a port and a memory device are

simultaneously enabled both drive the data bus and cause bus contention.

For output operation, external decoding logic combines $\overline{WR}$, IO/$\overline{M}$ and the port address and generates a unique output device select pulse as shown in fig.7.5. The device select occurs only during the I/O WRITE machine cycle of an OUT PORT instruction that addresses the port and checks the output port's register. Typically each output device select pulse checks a single output port; however it is possible that more than one port is selected simultaneously.

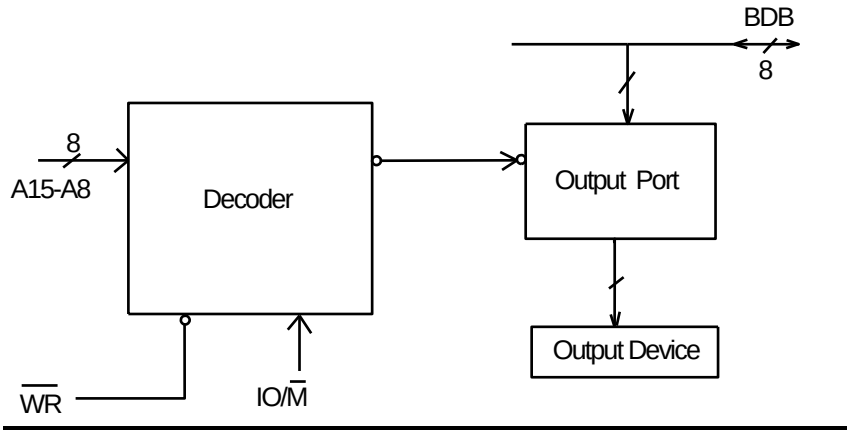

Fig.7.5 Interfacing Output Device using Isolated I/O

Lecture-41

Design of Device Selection Logic

The design of device selection logic varies, depending on how many I/O devices are required in a system. If only a single input port and a single output port are required, address decoding is unnecessary. The IO/$\overline{M}$ control signal is simply combined with $\overline{RD}$ to generate an input read strobe $\overline{I/OR}$ and with $\overline{WR}$ to generate output write strobe $\overline{I/OW}$. These strobes directly control the input buffer and the output latch, respectively. The port address byte of the I/O instruction, in such cases, is don't care but cannot be omitted from the instruction or its object code. IN PORT & OUT PORT are 2-byte instructions, and both bytes must appear in the program wherever these instruction are used. It may be 00H to FFH does not matter.

When more than one input or output port is required in a system, the port address needs to be decoded to generate device select pulses for each input and output port. The simplest form of decoding is the linear selection methods. This requires the smallest amount of logic but can only be used for eight or fewer input devices (input ports) and eight or fewer output devices (output ports). Here, one address bit is associated exclusively with each I/O port and is logically combined with IO/$\overline{M}$ and $\overline{RD}$ or $\overline{WR}$ strobe, delayed only by the propagation time of the device selection logic. For example, in fig. A_4 = 1 address bit selects port 16D. If the $\overline{RD}$ strobe is inverted and NANDed with A_4 and IO/$\overline{M}$, an active low device select pulse $\overline{IDSP\ 10\ H}$ is generated. This pulse is connected to the active low enable input of a tri-state buffer and determines when the buffer drive

the bus. If the tri-state buffer has multiple inputs, the internal logic of the tri-state buffer itself may be sufficient, and no external gates are required.

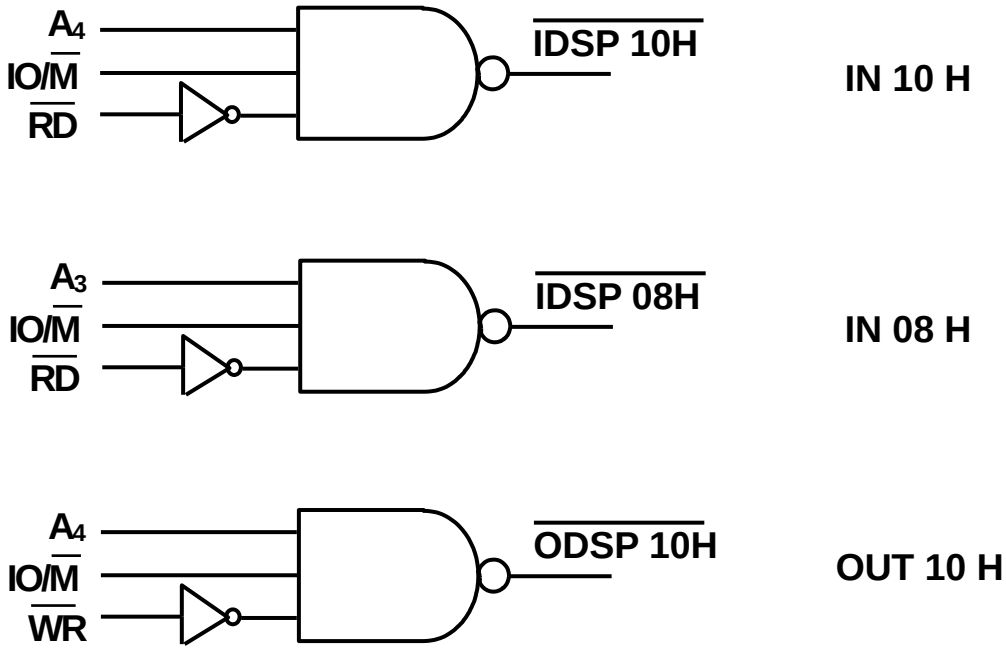

Fig.7.6 Generation of Device Selection Signal

A disadvantage of linear selection is the possibility that a programming error will damage the hardware. If the port address is not decoded properly, two or more input ports can drive the data bus simultaneously. Suppose, for example, a particular design is using linear selection method. The design contains an input port 4 selected by address bit A_2 and an input port 8 selected by address bit A_3. These input devices can be referred using IN 04 and IN 08 respectively. If a programming error results in the use of an IN 0C instruction instead of an IN 08, execution causes input ports 4 & 8 to be selected simultaneously and damaging the tri-state buffers. When linear selection is used extra care must be taken to ensure that two ports are not selected simultaneously.

Since there are only eight unique address bit in the port address, only eight input and eight output ports can be selected with linear selection method. To select from a large number of I/O devices requires decoding port addresses. With exhaustive decoding the maximum of device select pulses that can be generated is 512 - 256 associated with IN instructions, 256 with OUT instructions.

To generate more than one device select pulse, decoders are used to decode the address bits. If an application requires four or less a input and four or less output device select pulses, a 74139 dual 1 of 4 (or 2 to 4)decoder can be used. The active low enable of one of the decoder is connected to the $\overline{I/OR}$ strobe and the select inputs to address bits A_0 and A_1, providing four input device select pulses. All four outputs remain high until the $\overline{I/OR}$ strobe occurs. The active low strobe enables the decoder and an active low device select pulses occurs at the output selected by A_0 and A_1. The unselected outputs remain high. The active low enable of the other 1 of 4 decoder is connected to $\overline{I/OW}$ strobe and select inputs to address bits A_1 and A_0, thus providing four output device select pulses.

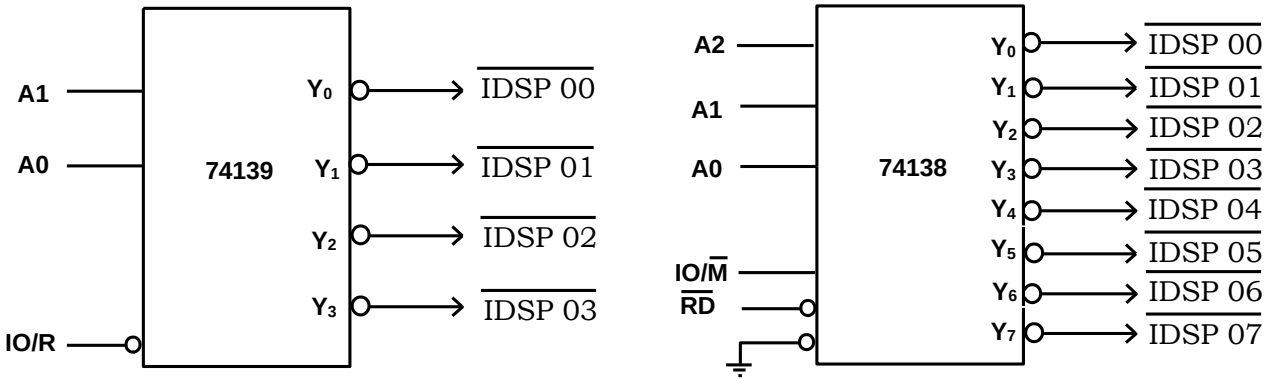

Fig.7.7 Use of Decoder Chips to Generate Device Select Pulse

Because only the least significant two address bits are connected to the decoder; the most significant six port address bits in the second byte of the instructions are don't cars. The effect of these don't cares is the generation of same device select pulses by a number of IN and OUT instruction with different addresses called 'FOLD BACK' addresses. Therefore, if the unused bits are considered zeros, the address is called primary address of the device and rest all possible addresses are 'FOLD BACK' addresses.

Decoders with both active low and active high enable inputs provide even greater flexibility. The 74LS138 is a 3 to 8 decoder with two active low and one active high enable inputs. When the required input combination is not applied to the enables, all outputs of the decoder are HIGH, regardless of the select or address input values. When 74LS138 is used to generate device select pulses, the $\overline{RD}$ or $\overline{WR}$ strobe is connected to one of the active low enables, the second input may be grounded and the IO/$\overline{M}$ signal is connected to the active high enable input. This arrangement requires no additional logic to generate an $\overline{I/OR}$ or $\overline{I/OW}$ device select strobes because generation is accomplished by the internal enable circuit of the chip.

Memory Mapped I/O :

In this case the I/O devices are not given separate addresses other than memory i.e. 0000H to FFFFH but part of this memory space is reserved for I/O devices. The advantage of memory mapped I/O is any instruction that references memory can also transfer data between an I/O device and the processor as long as the I/O port is

assigned to the memory address space rather than to the I/O address space. The registers associated with the I/O port are simply treated as memory location registers.

Consider an example in which address bit A_{15} designates whether instructions reference memory or an I/O device. If A_{15}= 0, a memory register is addressed; If A_{15}= 1, than a memory mapped I/O device is addressed. This assignment devotes the first 32k bytes of memory address space to memory devices and second 32k bytes of memory address space to memory mapped I/O devices. External logic generates device select pulses for memory mapped I/O only when IO/$\overline{M}$ = 0, the appropriate address is on the address bus and a $\overline{WR}$ or $\overline{RD}$ strobe occurs.

Input and output transfer using memory mapped I/O are not limited to the accumulator. For example, some of 8085A instructions that can be used for input from memory mapped I/O ports:

1. MOV r, M: Move the content of input port whose address is available in (H,L) register pair to any internal register (r).
2. LDA addr: Load the accumulator with the content of the input port whose address is available as a second and third byte of the instruction.

Few more instructions to input data from I/O devices include ANA M and ADD M. These instructions provide input data transfer and logical or arithmetic computation in a single instruction.

Another instruction is LHLD addr which inputs data from two ports and store the contents in (H) and (L) registers using a single instruction. The addresses of the input ports are addr as specified in the instruction as 2nd and 3rd bytes and next higher address addr+1.

Similarly, some of the instructions that output the data from memory mapped ports are

1. MOV M,r
2. STA addr
3. MVI M, data
4. SHLD addr

LHLD addr and SHLD addr instructions carry out 16-bit I/O transfers with single instructions which reduce program execution time considerably.

Therefore, memory mapped I/O can be used to transfer data from/to any general purpose registers, multiple data transfer and actions in one instruction. The price paid for this added capability is a reduction in directly addressable main memory and the necessity of decoding a 16- bit address rather than an 8-bit address.

When a microprocessor puts out an address and generates a control strobe for a memory read, it has no way of determining whether the device that responds with data is a memory device or an I/O device nor does it care. If only requires that the devices that respond does so within the allowable access time or uses the READY line to request a sufficient number of WAIT states. The same is true when microprocessor executes a write to memory. It supplies an address, data, and a write strobe and continues its operations. External logic determines whether memory, I/O or anything at all receives the data transferred.

Lecture-42

Interfacing I/O Devices

Example-1:

Interface 8-toggle switches with the microprocessor using isolated I/O address A0H. The required connection is shown in fig.7.8.

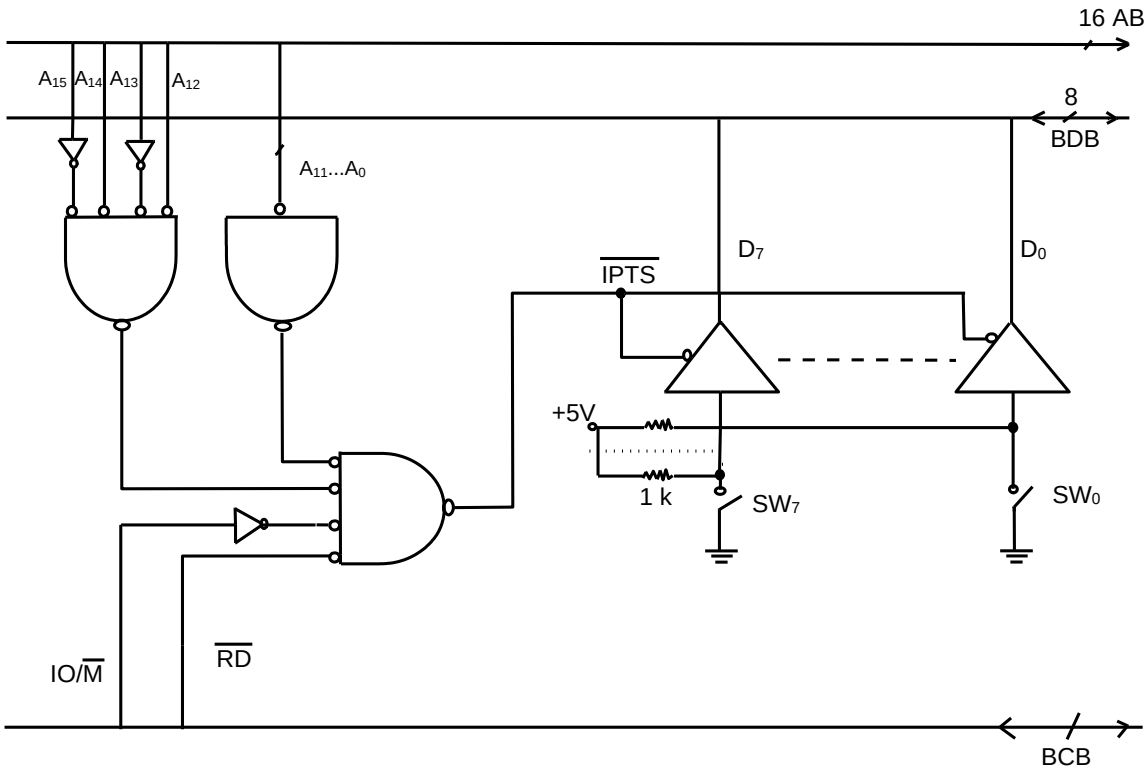

Fig.7.8 Interfacing of 8-Toggle Switches with All Address Lines Decoded

The open and closed positions of the switches are first converted to TTL logic using pull-up resistors. Whenever a switch is open, the output of circuit is logic '1' and whenever switch is closed, the output is logic '0'. These 8-toggle switches are connected to data bus through tri-state buffers having a common enable input $\overline{IPTS}$.

Whenever microprocessor intends 8-bits to be inputted from this input device, it just executes the instruction IN A0H. The 8

address lines are decoded along with IO/$\overline{M}$ and $\overline{RD}$. It is obvious from the logic circuit that whenever address A0H is put on address bus, IO/$\overline{M}$ is high, the circuit generates a LOW device select pulse $\overline{IPTS}$ when $\overline{RD}$ is active (i.e., during T_2 and T_3 states of I/O read machine cycle). This type of inputting data from the input device is known as CPU initiated unconditional I/O transfer.

Example-2:

This decoding circuit of example-1 consists of logic gates because we have used a single port address A0H to identify this input device. There is no reason why we have allocated only one address to an input device connected to input port. If many port addresses are vacant, we can simplify the decoding circuitry greatly by allocating more than one port address to the same device provided the user knows that these addresses are allocated to the same device and shall not be used anywhere to refer to any other input device.

Let us suppose in this example we have allocated all the 16 port address A0H to AFH to the same input device. The extra address A1H to AFH allocated to the same device are known as FOLD BACK ADDRESS once A0H to AFH is allocated to this device, there address should never be allocated to any other device. This is user responsibility.

By using address like this we have made the lower order 4-bits redundant. Thus the decoding circuitry shall be as shown in fig.7.9.

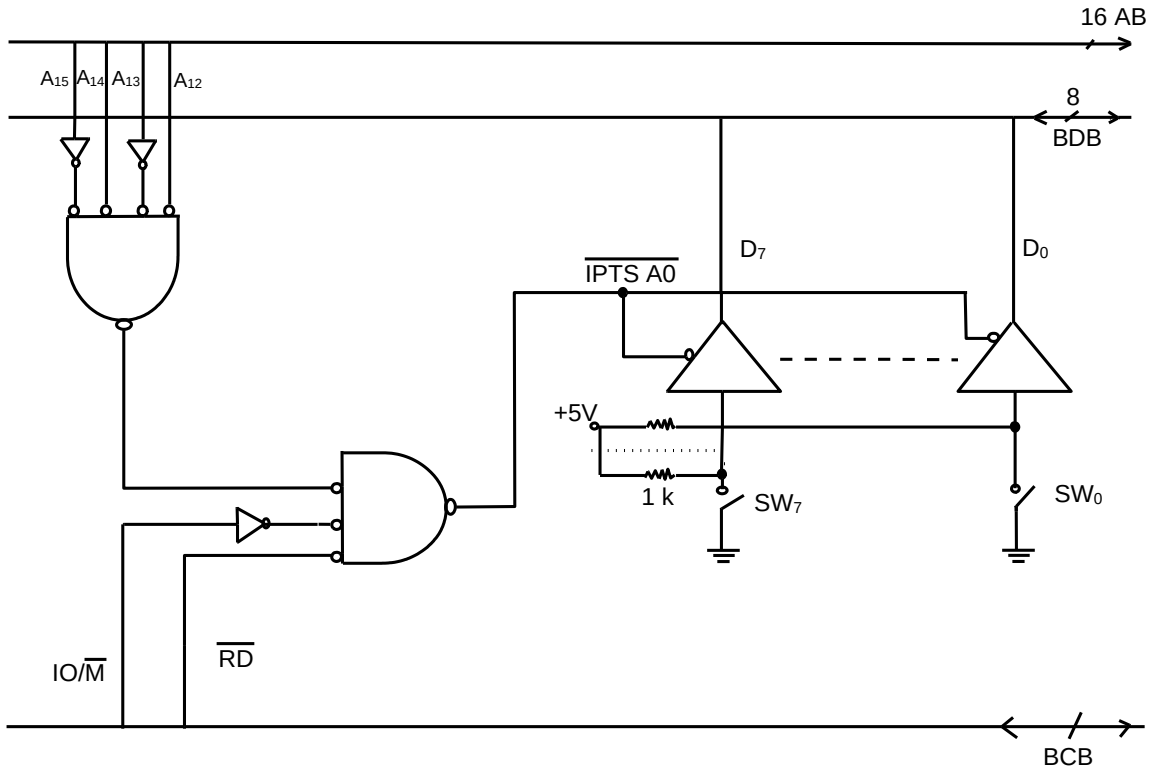

Fig.7.9 Interfacing of 8-Toggle Switches with Redundant Addresses

Microprocessor inputs an 8-bit data whenever required through CPU initiated unconditional I/O structure using IN PORT instruction & any one of the port addresses allocated to this input device. All the addresses assigned to this device will generate input device select pulse $\overline{\text{IPTS}}$.

Example 3:

In example-1 and 2 we have used SSI chips for obtaining the PORT select signal, we can also use MSI chips to simplify the decoding circuitry. Fig-7 gives the decoding circuitry for inputting 8-bit data from 8-toggle switches isolated I/O. PORT address allocated

being A0H to AFH with A0H as primary address and rest as fold back addresses. In the design, we have used 8205 (3 line to 8 line decoder). It could be 74LS138 another 3 line to 8 line decoder.

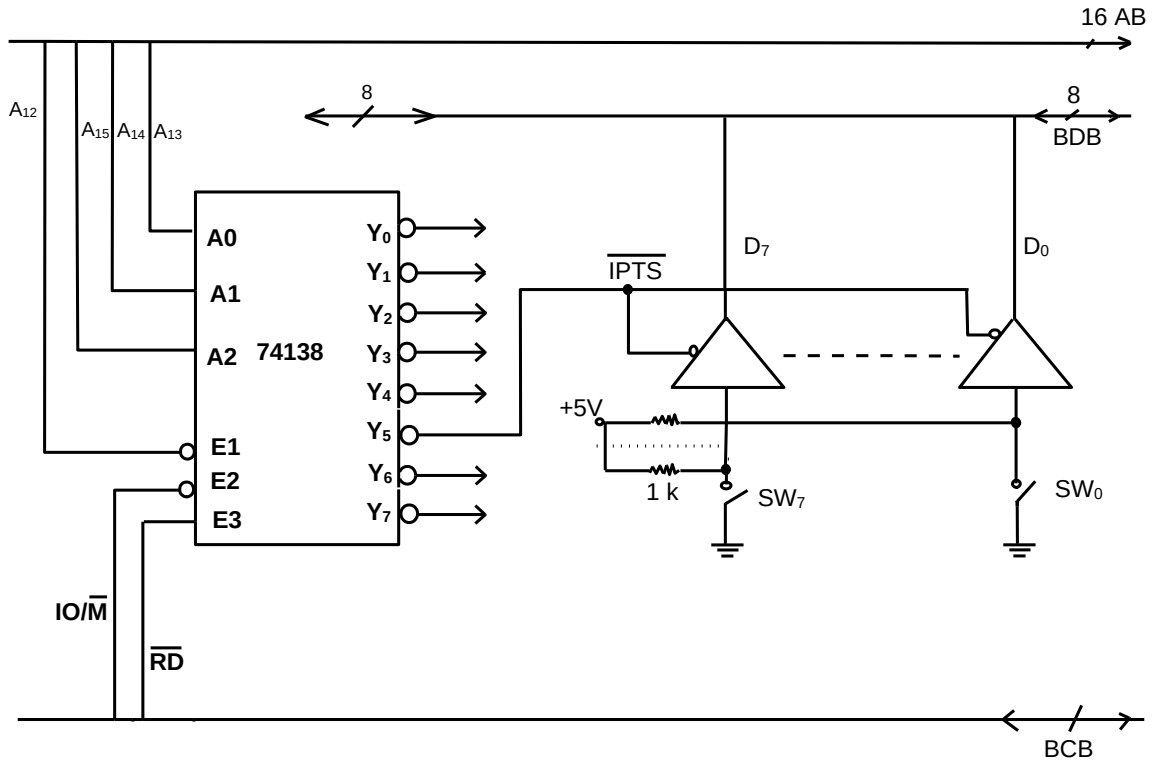

Fig.7.10 Interfacing of 8-Toggle Switches Using 3 Line to 8Line Decoder

The advantage of fig.7.10 can be realized when we have more than one port to select because a maximum of 8 chip select signals, can be generated using the same decoding circuitry of fig.7.10.

Example- 4:

In all the above examples, we have interfaced an I/O device using isolated I/O structure. Now let is interface the same device using memory mapped I/O structure. Memory space allocated for this device being page A0H. This means all the memory locations from

A000H to A0FFH has been allocated to this device. The decoding circuitry using 74LS138 decoder and logic gates is shown in fig.7.11.

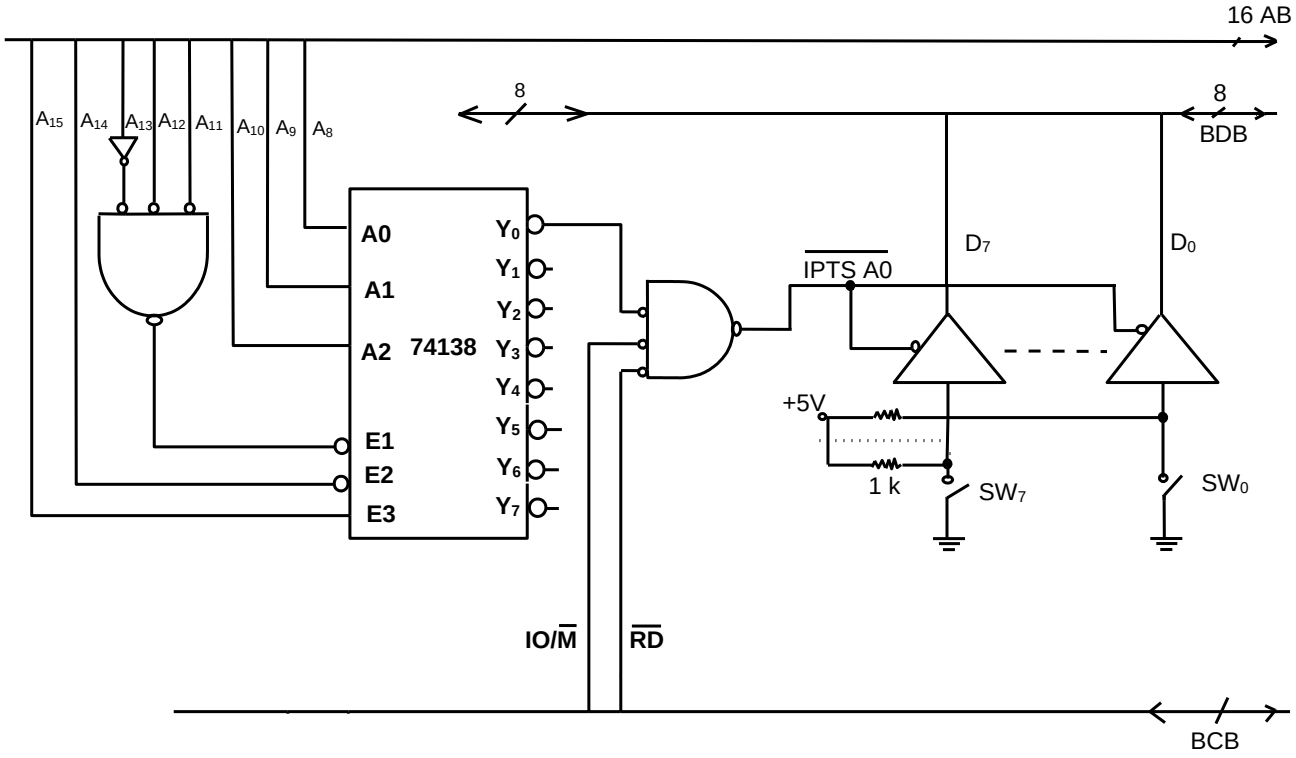

Fig.7.11 Interfacing of 8-Toggle Switches as Memory Mapped I/O

The decoder decodes the address and generates an active LOW pulse at output O_0 which is then combined with I/O$\overline{M}$ and $\overline{RD}$. If the address on address bus is A000H to A0FFH, I/O$\overline{M}$ and $\overline{RD}$ are LOW then the circuit generates input device select pulse $\overline{IPTS}$.

Since any of the address A000H to A0FFH can be used to select the device, therefore, only the higher byte of the address, i.e., page number (A0H) is to be decoded and not the complete 16 –bit address. The lower byte of the address is redundant. Figure shows one way to connect the higher order address bus to different inputs of decoder. There may be several ways to decode the address.

The following instruction sequence can be used to input an 8-bit data from 8-toggle switches, when necessary:

a) We may use instruction using register indirect addressing mode to input 8-bit data:

   LXI H, IPTSA0

   MOV r, M

b) On the other hand, we may use instruction using direct addressing mode to input 8-bit data:

   LDA IPTSA0

Note that whatever may be the memory reference instruction used for inputting 8-bit data from fig 8, the page address A0H should be maintained as higher order 8-bit while the lower order 8-bits are redundant and can be anything from 00H to FFH.