External Memory Interface

Introduction

- Memory chips are organized into a number of locations within the IC
  - Each location can hold 1 bit, 4 bits, 8 bits, or even 16 bits, depending on how it is designed internally
    - The number of locations within a memory IC depends on the address pins
    - The number of bits that each location can hold is always equal to the number of data pins
- To summarize
  - A memory chip contain 2^x location, where x is the number of address pins
  - Each location contains y bits, where y is the number of data pins on the chip
  - The entire chip will contain $2^x \times y$ bits

Introduction

- One of the most important characteristics of a memory chip is the speed at which its data can be accessed
  - To access the data, the address is presented to the address pins, the READ pin is activated, and after a certain amount of time has elapsed, the data shows up at the data pins
  - The shorter this elapsed time, the better, and consequently, the more expensive the memory chip
  - The speed of the memory chip is commonly referred to as its access time

Example

Example

A given memory chip has 12 address pins and 4 data pins. Find: (a) The organization, and (b) the capacity.

Solution:

(a) This memory chip has 4096 locations ($2^{12} = 4096$), and each location can hold 4 bits of data. This gives an organization of 4096×4, often represented as $4K \times 4$.

(b) The capacity is equal to 16K bits since there is a total of 4K locations and each location can hold 4 bits of data.

Example

A 512K memory chip has 8 pins for data. Find:

(a) The organization, and (b) the number of address pins for this memory chip.

Solution:

(a) A memory chip with 8 data pins means that each location within the chip can hold 8 bits of data. To find the number of locations within this memory chip, divide the capacity by the number of data pins. 512K/8 = 64K; therefore, the organization for this memory chip is $64K \times 8$

(b) The chip has 16 address lines since $2^{16} = 64K$

Memory Address Decoding

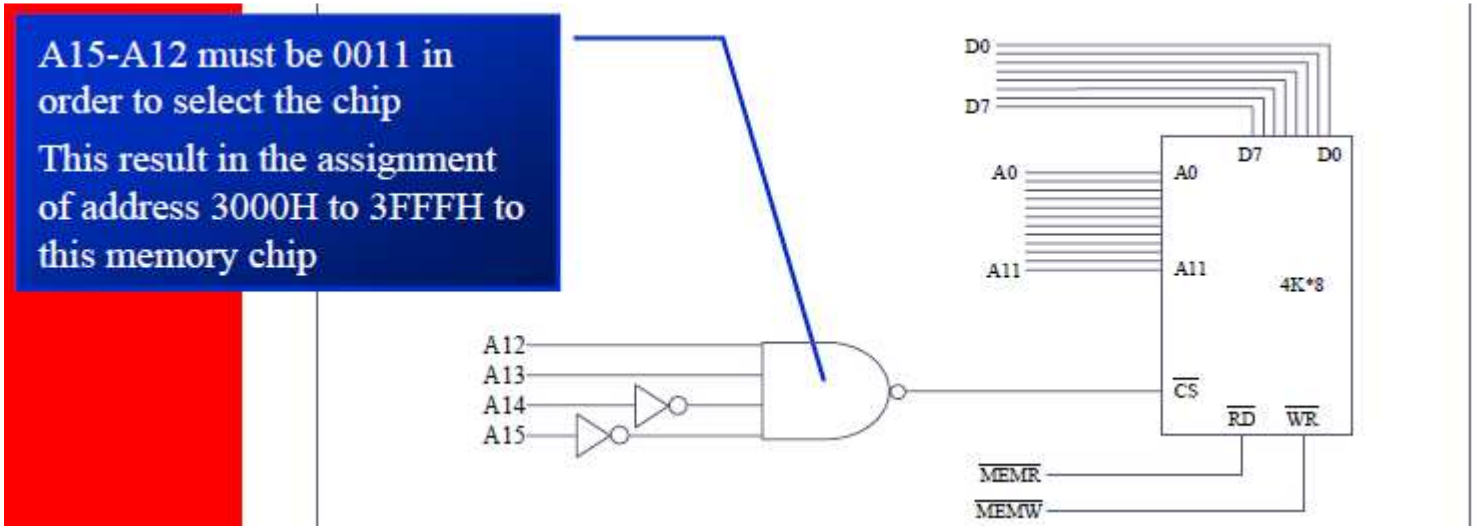
- The CPU provides the address of the data desired, but it is the job of the decoding circuitry to locate the selected memory block
  - Memory chips have one or more pins called CS (chip select), which must be activated for the memory's contents to be accessed
  - Sometimes the chip select is also referred to as chip enable (CE)

Memory Address Decoding

- In connecting a memory chip to the CPU, note the following points:
  - The data bus of the CPU is connected directly to the data pins of the memory chip
  - Control signals RD (read) and WR (memory write) from the CPU are connected to the OE (output enable) and WE (write enable) pins of the memory chip
  - In the case of the address buses, while the lower bits of the address from the CPU go directly to the memory chip address pins, the upper ones are used to activate the CS pin of the memory chip
- Normally memories are divided into blocks and the output of the decoder selects a given memory block
  - Using simple logic gates
  - Using the 74LS138
  - Using programmable logics

Simple Logic Gate Address Decoder

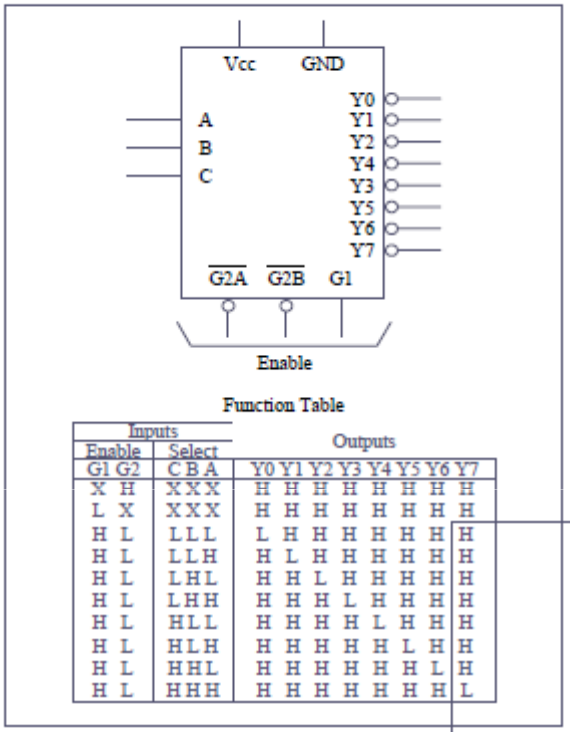
- The simplest way of decoding circuitry is the use of NAND or other gates
  - The fact that the output of a NAND gate is active low, and that the CS pin is also active low makes them a perfect match



Using 74LS138 3-8 Decoder

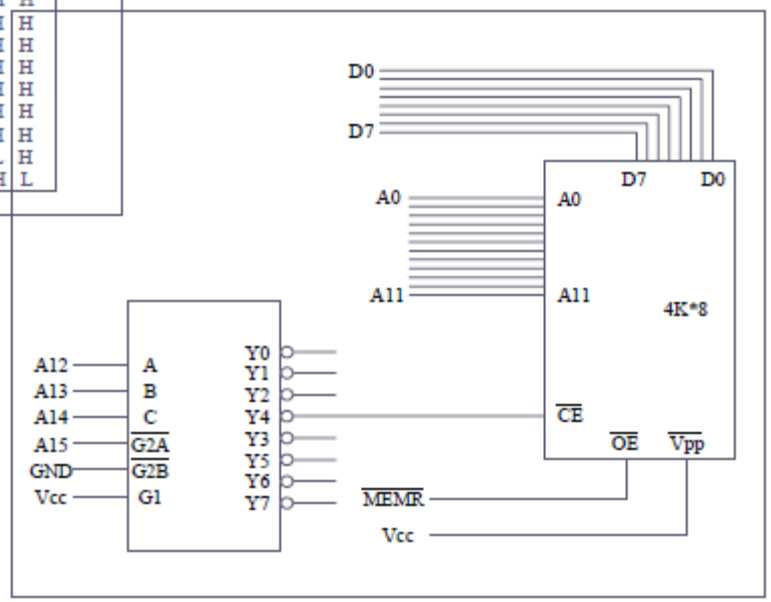
- This is one of the most widely used address decoders
  - The 3 inputs A, B, and C generate 8 active low outputs Y0 – Y7
    - Each Y output is connected to CS of a memory chip, allowing control of 8 memory blocks by a single 74LS138
- In the 74LS138, where A, B, and C select which output is activated, there are three additional inputs, G2A, G2B, and G1
  - G2A and G2B are both active low, and G1 is active high
    - If any one of the inputs G1, G2A, or G2B is not connected to an address signal, they must be activated permanently either by Vcc or ground, depending on the activation level

74LS138 Decoder



Function Table

Inputs			Outputs							
Enable		Select								
G1	G2	C B A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X X X	H	H	H	H	H	H	H	H
L	X	X X X	H	H	H	H	H	H	H	H
H	L	L L L	L	H	H	H	H	H	H	H
H	L	L L H	H	L	H	H	H	H	H	H
H	L	L H L	H	H	L	H	H	H	H	H
H	L	L H H	H	H	H	L	H	H	H	H
H	L	H L L	H	H	H	H	L	H	H	H
H	L	H L H	H	H	H	H	H	L	H	H
H	L	H H L	H	H	H	H	H	H	L	H
H	L	H H H	H	H	H	H	H	H	H	L



Looking at the design in Figure 14-6, find the address range for the Following. (a) Y4, (b) Y2, and (c) Y7.

Solution :

(a) The address range for Y4 is calculated as follows.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

The above shows that the range for Y4 is 4000H to 4FFFH. In Figure 14-6, notice that A15 must be 0 for the decoder to be activated. Y4 will be selected when A14 A13 A12 = 100 (4 in binary). The remaining A11-A0 will be 0 for the lowest address and 1 for the highest address.

(b) The address range for Y2 is 2000H to 2FFFH.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

(c) The address range for Y7 is 7000H to 7FFFH.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Using Programmable Logic

- Other widely used decoders are programmable logic chips such as PAL and GAL chips
  - One disadvantage of these chips is that one must have access to a PAL/GAL software and burner, whereas the 74LS138 needs neither of these
  - The advantage of these chips is that they are much more versatile since they can be programmed for any combination of address ranges

INTERFACING EXTERNAL ROM

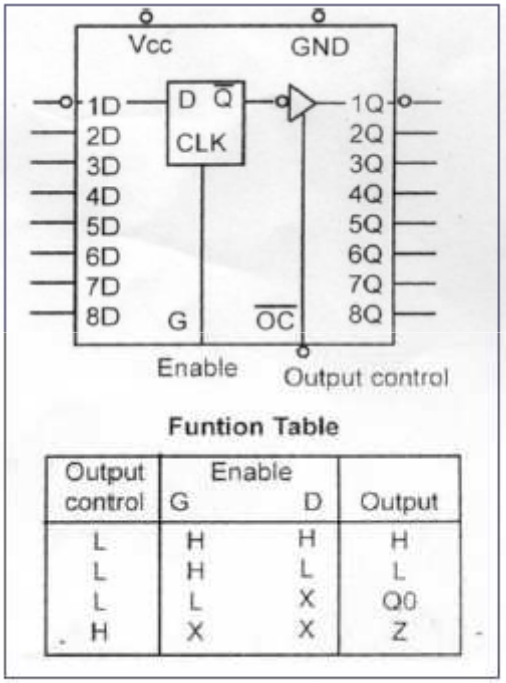
- The 8031 chip is a ROMless version of the 8051
  - It is exactly like any member of the 8051 family as far as executing the instructions and features are concerned, but it has no on-chip ROM
  - To make the 8031 execute 8051 code, it must be connected to external ROM memory containing the program code
- 8031 is ideal for many systems where the on-chip ROM of 8051 is not sufficient, since it allows the program size to be as large as 64K bytes
- For 8751/89C51/DS5000-based system, we connected the EA pin to Vcc to indicate that the program code is stored in the microcontroller's on-chip ROM
- To indicate that the program code is stored in external ROM, this pin must be connected to GND

P0 and P2 in Providing Address

- Since the PC (program counter) of the 8031/51 is 16-bit, it is capable of accessing up to 64K bytes of program code
  - In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory
    - P0 provides the lower 8 bit address A0 – A7, and P2 provides the upper 8 bit address A8 – A15
    - P0 is also used to provide the 8-bit data bus D0 – D7
  - P0.0 – P0.7 are used for both the address and data paths
    - address/data multiplexing

P0 and P2 in Providing Address

- ALE (address latch enable) pin is an output pin for 8031/51
- ALE = 0, P0 is used for data path
- ALE = 1, P0 is used for address path
- To extract the address from the P0 pins we connect P0 to a 74LS373 and use the ALE pin to latch the address



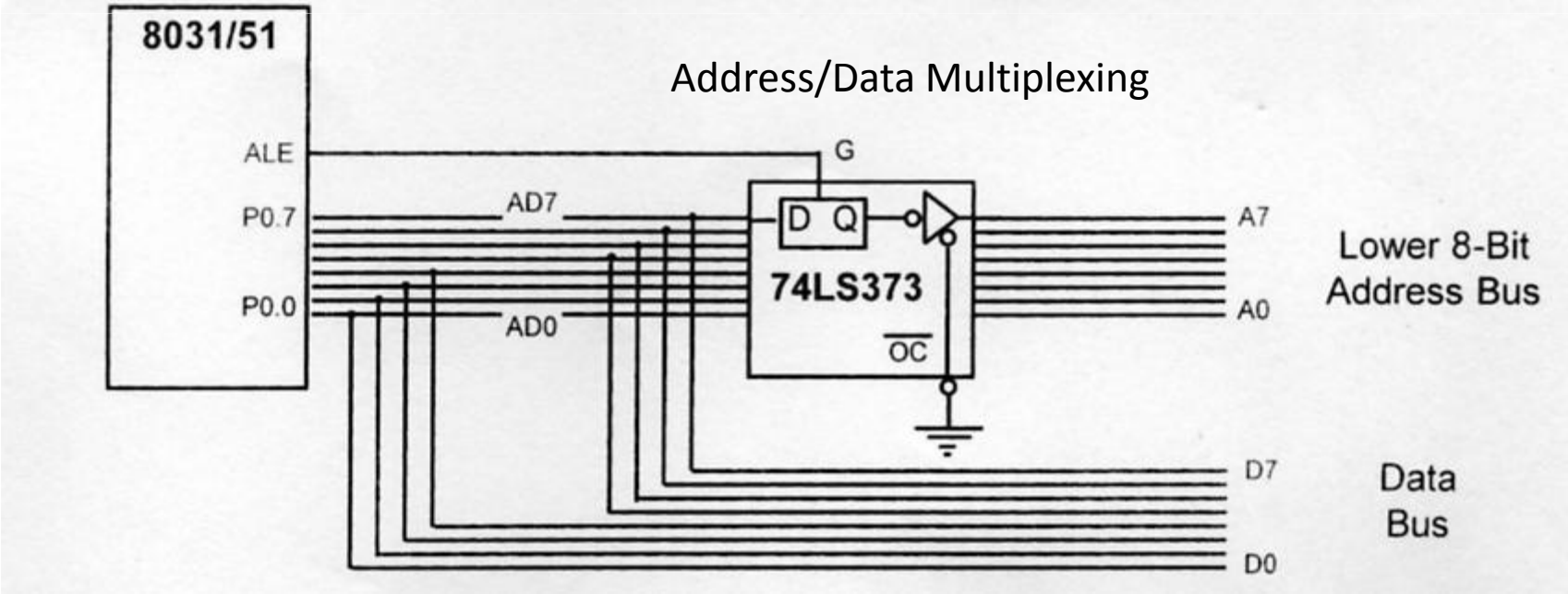
Funtion Table

Output control	Enable G	Enable D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

74LS373 D Latch

P0 and P2 in Providing Address

- Normally ALE = 0, and P0 is used as a data bus, sending data out or bringing data in
- Whenever the 8031/51 wants to use P0 as an address bus, it puts the addresses A0 – A7 on the P0 pins and activates ALE = 1



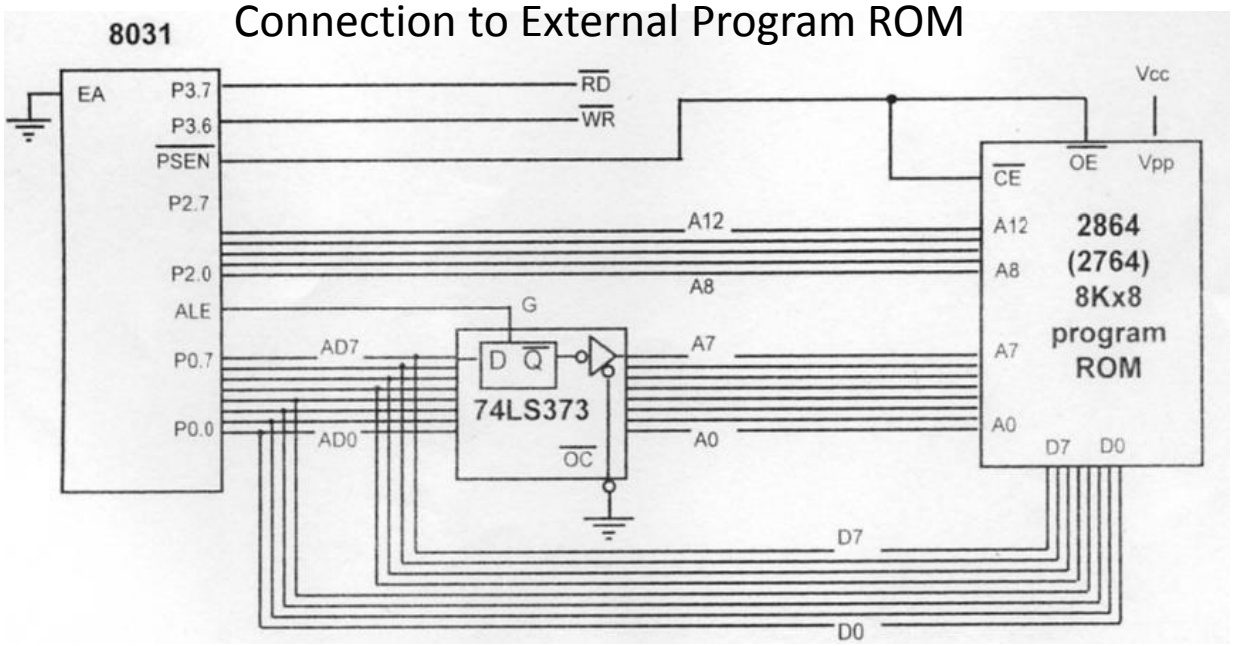
PSEN

- PSEN (program store enable) signal is an output signal for the 8031/51 microcontroller and must be connected to the OE pin of a ROM containing the program code
- It is important to emphasize the role of EA and PSEN when connecting the 8031/51 to external ROM
- When the EA pin is connected to GND, the 8031/51 fetches opcode from external ROM by using PSEN

PSEN

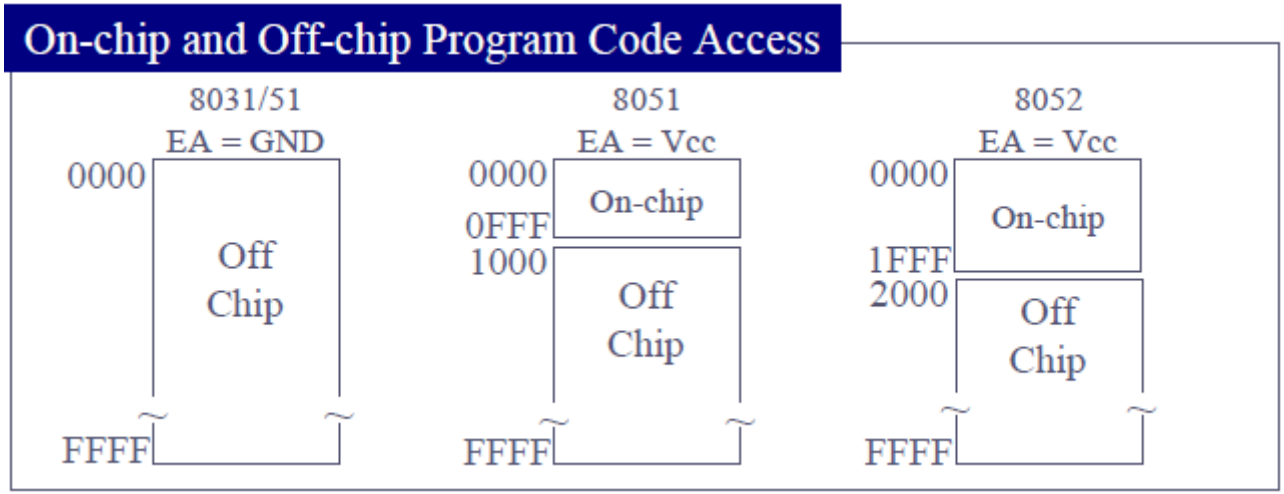
- The connection of the PSEN pin to the OE pin of ROM
  - In systems based on the 8751/89C51/ DS5000 where EA is connected to Vcc, these chips do not activate the PSEN pin
    - This indicates that the on-chip ROM contains program code

Connection to External Program ROM



On-Chip and Off-Chip Code ROM

- In an 8751 system we could use onchip ROM for boot code and an external ROM will contain the user's program
  - We still have EA = Vcc,
    - Upon reset 8051 executes the on-chip program first, then
    - When it reaches the end of the on-chip ROM, itswitches to external ROM for rest of program



Discuss the program ROM space allocation for each of the following cases.

(a) EA = 0 for the 8751 (89C51) chip.

(b) EA = V_{cc} with both on-chip and off-chip ROM for the 8751.

(c) EA = V_{cc} with both on-chip and off-chip ROM for the 8752.

Solution:

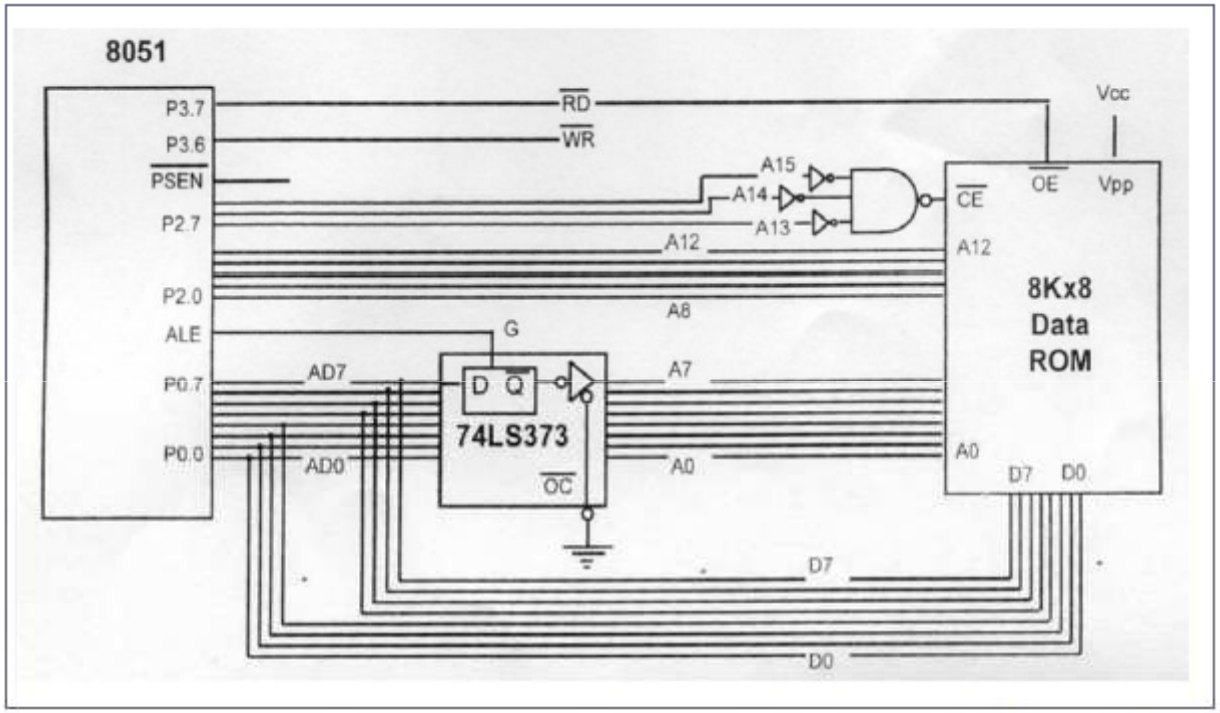
(a) When EA = 0, the EA pin is strapped to GND, and all program fetches are directed to external memory regardless of whether or not the 8751 has some on-chip ROM for program code. This external ROM can be as high as 64K bytes with address space of 0000 – FFFFH. In this case an 8751(89C51) is the same as the 8031 system.

(b) With the 8751 (89C51) system where EA=V_{cc}, it fetches the program code of address 0000 – 0FFFH from on-chip ROM since it has 4K bytes of on-chip program ROM and any fetches from addresses 1000H – FFFFH are directed to external ROM.

(c) With the 8752 (89C52) system where EA=V_{cc}, it fetches the program code of addresses 0000 – 1FFFH from on-chip ROM since it has 8K bytes of on-chip program ROM and any fetches from addresses 2000H – FFFFH are directed to external ROM

Data Memory space

- The 8051 has 128K bytes of address space
  - 64K bytes are set aside for program code
    - Program space is accessed using the program counter (PC) to locate and fetch instructions
    - In some example we placed data in the code space and used the instruction MOVC A,@A+DPTR to get data, where C stands for code
  - The other 64K bytes are set aside for data
    - The data memory space is accessed using the DPTR register and an instruction called MOVX, where X stands for external
      - The data memory space must be implemented externally



8051 Connection to External Data ROM

- MOVX is a widely used instruction allowing access to external data memory space
  - To bring externally stored data into the CPU, we use the instruction

    `MOVX A,@DPTR`

An external ROM uses the 8051 data space to store the look-up table (starting at 1000H) for DAC data. Write a program to read 30 Bytes of these data and send it to P1.

Solution:

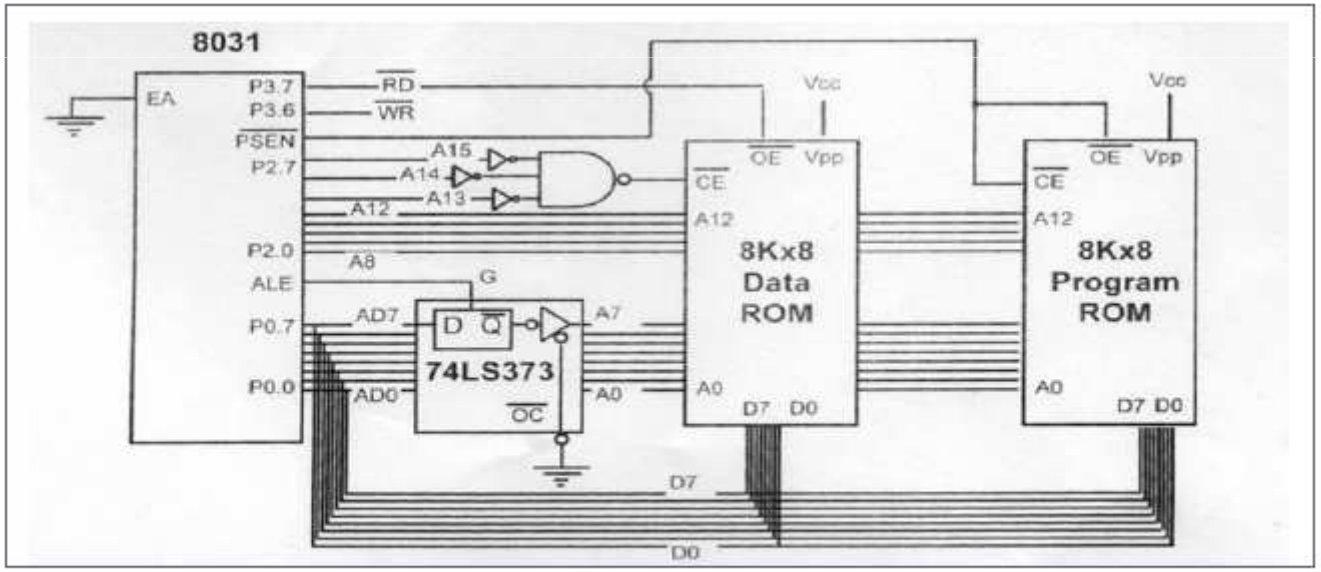
```
MYXDATA EQU   1000H
COUNT   EQU   30
         ...
        MOV   DPTR,#MYXDATA
        MOV   R2,#COUNT
AGAIN:  MOVX  A,@DPTR
        MOV   P1,A
        INC   DPTR
        DJNZ  R2,AGAIN
```

Although both `MOVC A,@A+DPTR` and `MOVX A,@DPTR` look very similar, one is used to get data in the code space and the other is used to get data in the data space of the microcontroller

Show the design of an 8031-based system with 8K bytes of program ROM and 8K bytes of data ROM.

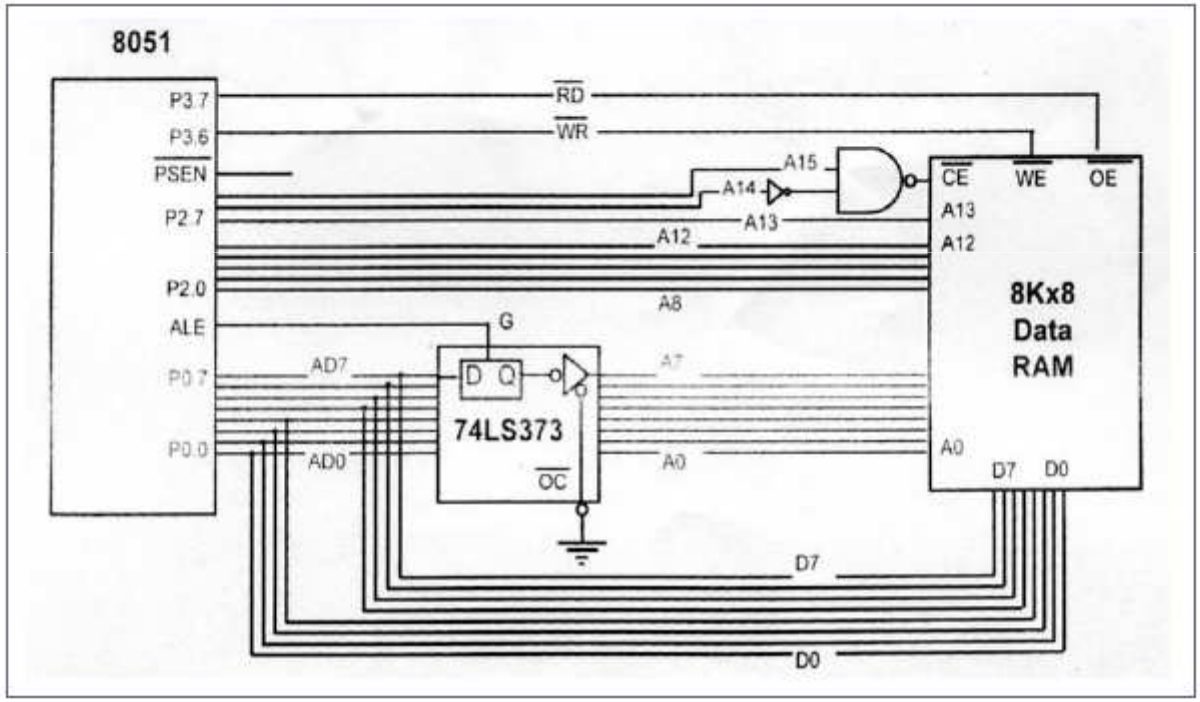
Solution:

Figure 14-14 shows the design. Notice the role of PSEN and RD in each ROM. For program ROM, PSEN is used to activate both OE and CE. For data ROM, we use RD to active OE, while CE is activated by a Simple decoder.



8031 Connection to External Data ROM and External Program ROM

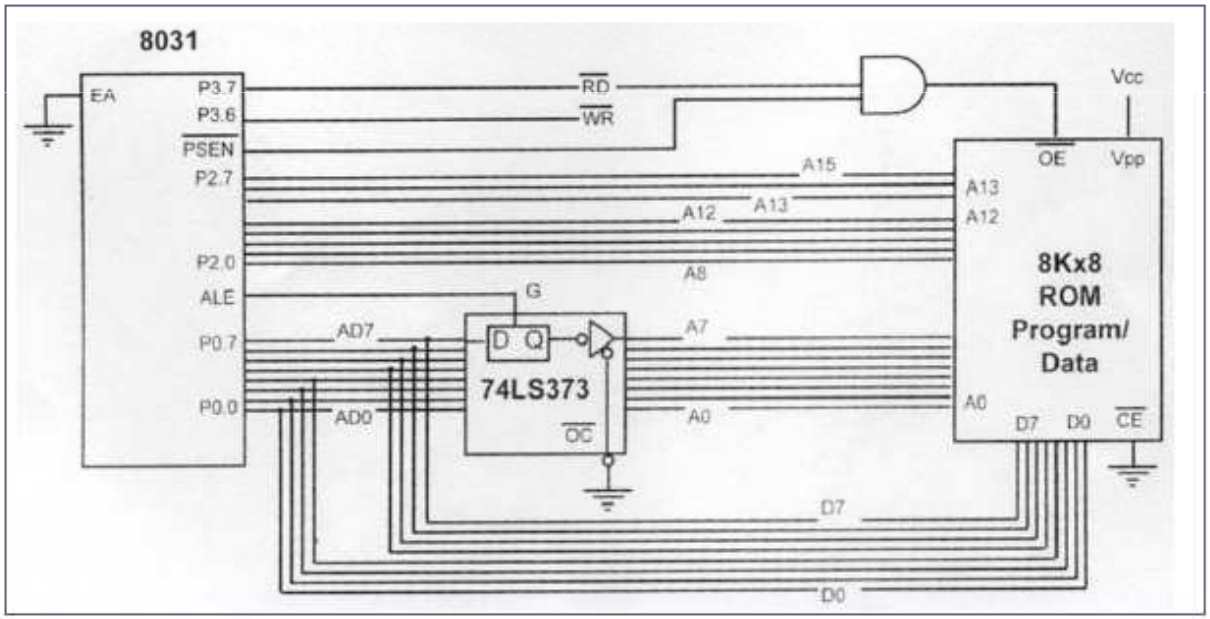
- To connect the 8051 to an external SRAM, we must use both RD (P3.7) and WR (P3.6)



8051 Connection to External Data RAM

Single External ROM for Code and Data

- To allow a single ROM chip to provide both program code space and data space, we use an AND gate to signal the OE pin of the ROM chip



A Single ROM for BOTH Program and Data