

| Dec | Hex | Bin      |
|-----|-----|----------|
| 10  | A   | 00001010 |

**ORG ; Week8**

## **Memory and Memory Interfacing**

# The x86 PC

assembly language,  
design, and interfacing

fifth edition

**MUHAMMAD ALI MAZIDI  
JANICE GILLISPIE MAZIDI  
DANNY CAUSEY**

**The x86 PC**  
assembly language, design, and interfacing

fifth  
edition

Prentice Hall

# OBJECTIVES

this chapter enables the student to:

- Define the terms *capacity*, *organization*, and *speed* as used in semiconductor memories.
- Calculate the chip capacity and organization of semiconductor memory chips.
- Compare and contrast the variations of ROM
  - PROM, EPROM, EEPROM, Flash EPROM, mask ROM.
- Compare and contrast the variations of RAM
  - SRAM, DRAM, NV-DRAM.
- Diagram methods of address decoding for memory chips.

## OBJECTIVES

(*cont*)

this chapter enables the student to:

- Diagram the memory map of the IBM PC in terms of RAM, VDR, and ROM allocation.
- Describe the checksum method of ensuring data integrity in ROM.
- Describe the parity bit method of ensuring data integrity in DRAM.
- Describe 16-bit memory design and related issues.

## 10.1: SEMICONDUCTOR MEMORIES

- In all computer design, semiconductor memories are used as primary storage for code & data.
  - Connected directly to the CPU, they are asked first by the CPU for information (code and data).
    - Referred to as *primary memory*.
- Primary memory must respond fast to the CPU.
  - Only semiconductor memories can do that
    - Among the most widely used are ROM and RAM.

# Semiconductor Memory Fundamentals

- In the design of all computers, semiconductor memories are used as primary storage for data and code.
- They are connected directly to the CPU and they are the memory that the CPU asks for information (code or data)
- Among the most widely used are RAM and ROM
- **Memory Capacity**
  - The number of bits that a semiconductor memory chip can store is called its chip capacity (bits or bytes)
- **Memory Organization**
  - Each memory chip contains  $2^x$  locations where  $x$  is the number of address pins on the chip
  - Each location contains  $y$  bits, where  $y$  is the number of data pins on the chip
  - The entire chip will contain  $2^x * y$  bits
  - Ex. Memory organization of 4K x 4:  $2^{12} = 4096$  locations, each location holding 4 bits
- **Memory Speed** (access time)

# 10.1: SEMICONDUCTOR MEMORIES

## memory capacity

- The number of bits a semiconductor memory chip can store is called its *chip capacity*.
  - In units of K bits (kilobits), M bits (megabits), etc.
    - Memory capacity of a *memory IC chip* is always given in *bits*.
    - Memory capacity of a *computer* is given in *bytes*.

# 10.1: SEMICONDUCTOR MEMORIES

## memory organization

- Memory chips are organized into a number of *locations* within the IC.
  - Each can hold 1, 4, 8, or even 16 bits.
    - Depending on internal design.
  - The number of bits each location can hold is always equal to the number of data pins on the chip.
  - The number of locations in a memory chip depends on the number of address pins.
    - Always  $2^x$ , where  $x$  is the number of address pins.

# 10.1: SEMICONDUCTOR MEMORIES

## memory organization summarized

- Each memory IC chip contains  $2^x$  locations, where  $x$  is the number of chip address pins.
  - Each location contains  $y$  bits, where  $y$  is the number of data pins on the chip.
- The entire chip contains  $2^x \times y$  bits, where  $x$  is the number of address pins and  $y$  the number of data pins.
  - The  $2^x \times y$  is referred to as the *organization* of the memory chip, with  $x$  expressed as the number of address pins, and  $y$  the number of data pins.
- $2^{10} = 1024 = 1\text{K}$ . (*Kilo* = 1000. 1 Kilobyte)
  - Note that in common speech, 1K is *1000*, but in computer terminology it is *1024*. See Table 10-1.

**Table 10-1:**  
**Powers of 2**

| $x$ | $2^x$ |
|-----|-------|
| 10  | 1K    |
| 11  | 2K    |
| 12  | 4K    |
| 13  | 8K    |
| 14  | 16K   |
| 15  | 32K   |
| 16  | 64K   |
| 17  | 128K  |
| 18  | 256K  |
| 19  | 512K  |
| 20  | 1M    |
| 21  | 2M    |
| 22  | 4M    |
| 23  | 8M    |
| 24  | 16M   |



## 10.1: SEMICONDUCTOR MEMORIES

### speed

- A most important characteristic of a memory chip is the speed at which data can be accessed from it.
  - To access the data, the address is presented to the address pins, 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 more expensive) the memory chip.
- The speed of the memory chip is commonly referred to as its access time.
  - Varies from a few nanoseconds to hundreds of nanoseconds.

# 10.1: SEMICONDUCTOR MEMORIES

## ROM read-only memory

- ROM is a type of memory that does not lose its contents when the power is turned off.
  - Also called nonvolatile memory.
  - There are different types of read-only memory:
    - PROM, EPROM, EEPROM, Flash ROM, and mask ROM.

## 10.1: SEMICONDUCTOR MEMORIES

### PROM programmable ROM or OTP ROM

- PROM refers to the kind of ROM that the user can burn information into.
  - A user-programmable memory.
- The programming process is called *burning*, using special equipment. (A ROM burner or programmer)
- For every bit of the PROM, there exists a fuse.
  - PROM is programmed by blowing the fuses.
  - If information burned into PROM is wrong, it must be discarded, as the internal fuses are blown permanently.
  - For this reason, PROM is also referred to as OTP. (one-time programmable)

## 10.1: SEMICONDUCTOR MEMORIES

### EPROM erasable programmable ROM

- EPROM was invented to allow changes in the contents of PROM after it is burned.
  - One can program/erase the memory chip many times.
    - Useful during prototyping of a microprocessor-based projects.
- All EPROM chips have a window, to shine ultraviolet (UV) radiation to erase the chip's contents.
  - EPROM is also referred to as UV-erasable EPROM or simply UV-EPROM.
  - Erasing EPROM contents can take up to 20 minutes.
  - It cannot be programmed while in the system board (motherboard).

# 10.1: SEMICONDUCTOR MEMORIES

## EPR0M programming steps

- 1. Erase the contents.
  - Remove it from its system board socket, and use EPROM erasure equipment to expose it to UV radiation.
- 2. Program the chip.
  - To burn code & data into EPROM, the ROM burner uses 12.5 volts or higher, (called VPP), depending on type.
    - EEPROM with VPP of 5–7 V is available, but it is more expensive.
- 3. Replace the chip in its socket.

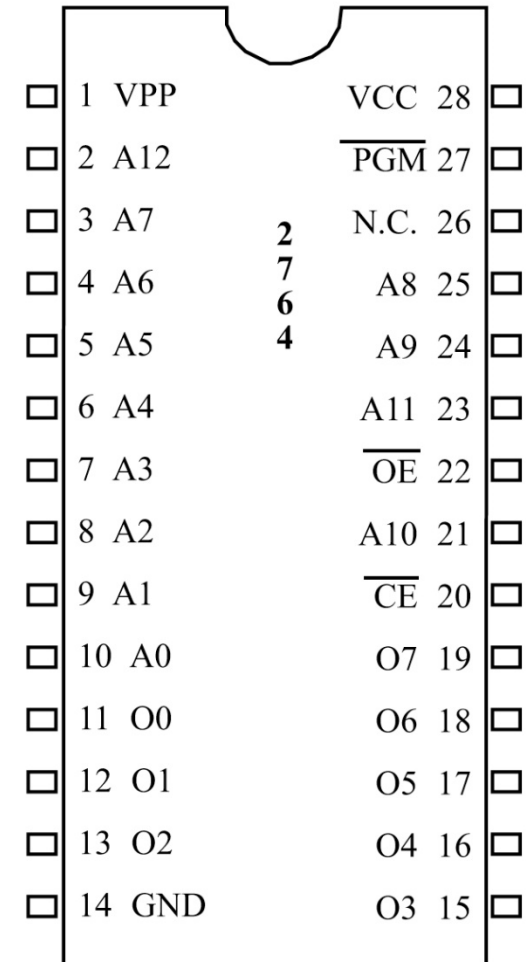


Fig. 10-1 UV-EPROM Chip

# 10.1: SEMICONDUCTOR MEMORIES

## EPROM erasable programmable ROM

- Note the **A0–A12** address pins and **O0–O7** (output) for D0–D7 data pins.
  - **OE** (out enable) is for the read signal.

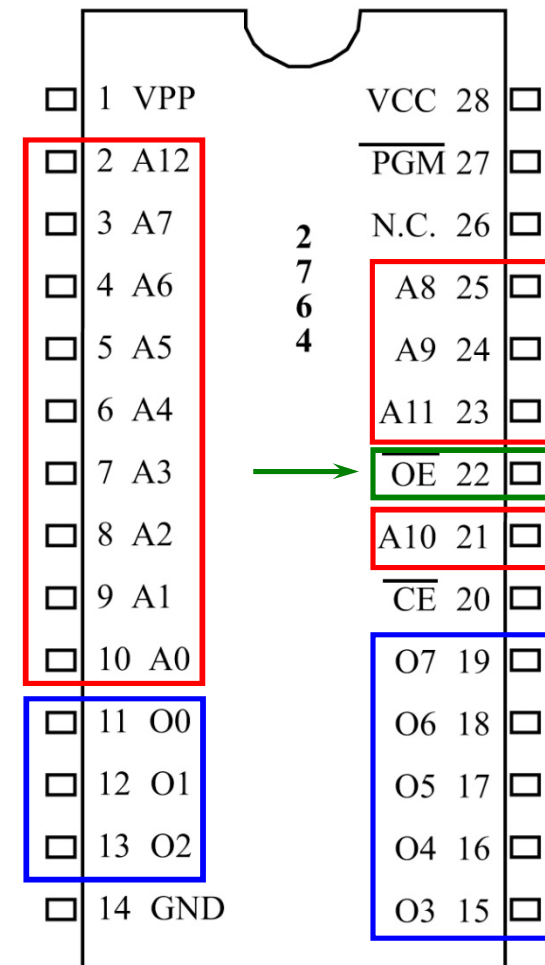


Fig. 10-1 UV-EPROM Chip

# 10.1: SEMICONDUCTOR MEMORIES

## flash memory

- Since the early 1990s, Flash ROM has become a popular user-programmable memory chip.
  - The process of erasure of the entire contents takes only a few seconds. (In a *flash*, hence the name)
  - Electrical erasure lends the nickname Flash EEPROM.
    - To avoid confusion, it is commonly called Flash ROM.
- When Flash memory's contents are erased the entire device is erased.
  - In contrast to EEPROM, where one sections or bytes.
  - Some Flash memories recently available are divided into blocks, and erasure can be done by block.
    - No byte erasure option is yet available.

# 10.1: SEMICONDUCTOR MEMORIES

## flash memory

- Because Flash ROM can be programmed while in its system board socket, it is widely used to upgrade PC BIOS ROM, or Cisco router operating systems.
  - Some designers believe that Flash memory will replace the hard disk as a mass storage medium.
- The program/erase cycle is 500,000 for Flash and EEPROM; 2000 for UV-EPROM; infinite for RAM & disks.
  - Program/erase cycle is the number of times a chip can be erased and programmed before it becomes unusable.



# 10.1: SEMICONDUCTOR MEMORIES

## memory identification

**Example 10-3** For ROM chip 27128, find the number of data and address pins, in Table 10-2.

**Solution:**

The 27128 has a capacity of 128K bits. Table 10-2 also shows that it has 16K × 8 organization, which indicates that there are 8 pins for data, and 14 pins for address ( $2^{14} = 16K$ ).

**Table 10-2: Examples of ROM Memory Chips**

| Type      | Part Number | Speed (ns) | Capacity | Organization | Pins | VPP  |
|-----------|-------------|------------|----------|--------------|------|------|
| UV-EPROM  | 2716        | 450        | 16K      | 2K × 8       | 24   | 25   |
|           | 27128-20    | 200        | 128K     | 16K × 8      | 28   | 12.5 |
|           | 2732A-45    | 450        | 32K      | 4K × 8       | 24   | 21   |
| EEPROM    | 28C16A-25   | 250        | 16K      | 2K × 8       | 24   | 5    |
|           | 2864A       | 250        | 64K      | 8K × 8       | 28   | 5    |
|           | 28C256-15   | 150        | 256K     | 32K × 8      | 28   | 5    |
| Flash ROM | 28F256-20   | 200        | 256K     | 32K × 8      | 32   | 12   |
|           | 28F256-15   | 150        | 256K     | 32K × 8      | 32   | 12   |

**See the entire table on page 259 of your textbook.**

# 10.1: SEMICONDUCTOR MEMORIES

## memory identification

- Capacity of the memory chip is indicated in the part number, with access time given with a zero dropped.
  - 27128-20** refers to UV-EPROM that has a capacity of **128K** bits and access time of **200** nanoseconds.

**Table 10-2: Examples of ROM Memory Chips**

| Type      | Part Number | Speed (ns) | Capacity | Organization | Pins | VPP  |
|-----------|-------------|------------|----------|--------------|------|------|
| UV-EPROM  | 2716        | 450        | 16K      | 2K × 8       | 24   | 25   |
|           | 27128-20    | 200        | 128K     | 16K × 8      | 28   | 12.5 |
|           | 2732A-45    | 450        | 32K      | 4K × 8       | 24   | 21   |
| EEPROM    | 28C16A-25   | 250        | 16K      | 2K × 8       | 24   | 5    |
|           | 2864A       | 250        | 64K      | 8K × 8       | 28   | 5    |
|           | 28C256-15   | 150        | 256K     | 32K × 8      | 28   | 5    |
| Flash ROM | 28F256-20   | 200        | 256K     | 32K × 8      | 32   | 12   |
|           | 28F256-15   | 150        | 256K     | 32K × 8      | 32   | 12   |

**See the entire table on page 259 of your textbook.**

# 10.1: SEMICONDUCTOR MEMORIES

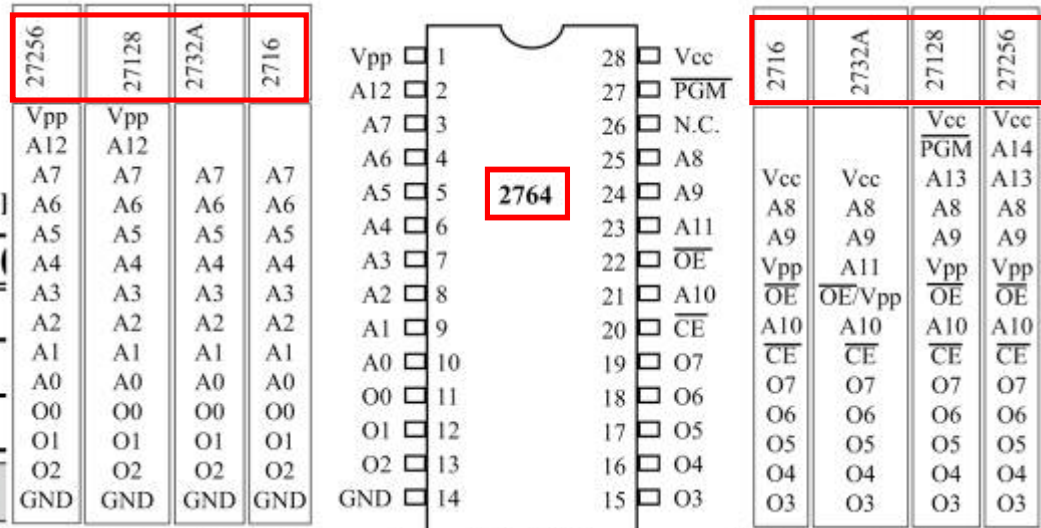
## memory identification

- In part numbers, C refers to CMOS technology.
  - 27xx** is for UV-E PROM
  - 28xx** for EEPROM.

Table 10-2: Examples of ROM Memori

| Type      | Part Number | Speed (ns) | Memory Size | Organization | Package | Pin Count |
|-----------|-------------|------------|-------------|--------------|---------|-----------|
| UV-E PROM | 2716        | 450        |             |              |         |           |
|           | 27128-20    | 200        |             |              |         |           |
|           | 2732A-45    | 450        |             |              |         |           |
| EEPROM    | 28C16A-25   | 250        |             |              |         |           |
|           | 2864A       | 250        | 64K         | 8K × 8       | 28      | 5         |
|           | 28C256-15   | 150        | 256K        | 32K × 8      | 28      | 5         |
| Flash ROM | 28F256-20   | 200        | 256K        | 32K × 8      | 32      | 12        |
|           | 28F256-15   | 150        | 256K        | 32K × 8      | 32      | 12        |

See the entire table on page 259 of your textbook.



## 10.1: SEMICONDUCTOR MEMORIES

### mask ROM

- Mask ROM refers to a kind of ROM whose contents are programmed by the IC manufacturer.
  - Not a user-programmable ROM.
  - The terminology *mask* is used in IC fabrication.
- Mask ROM is used when needed volume is high & it is absolutely certain the contents will not change.
  - Since the process is costly.
- The cost is significantly cheaper than other kinds of ROM, but if an error in the data is found, the *entire batch* must be discarded.

# 10.1: SEMICONDUCTOR MEMORIES

## RAM random access memory

- RAM memory is called *volatile memory* since cutting off the power to the IC will mean the loss of data.
  - Sometimes referred to as RAWM (read & write memory).
- There are three types of RAM:
  - Static RAM (SRAM)
  - Dynamic RAM (DRAM)
  - NV-RAM (nonvolatile RAM)

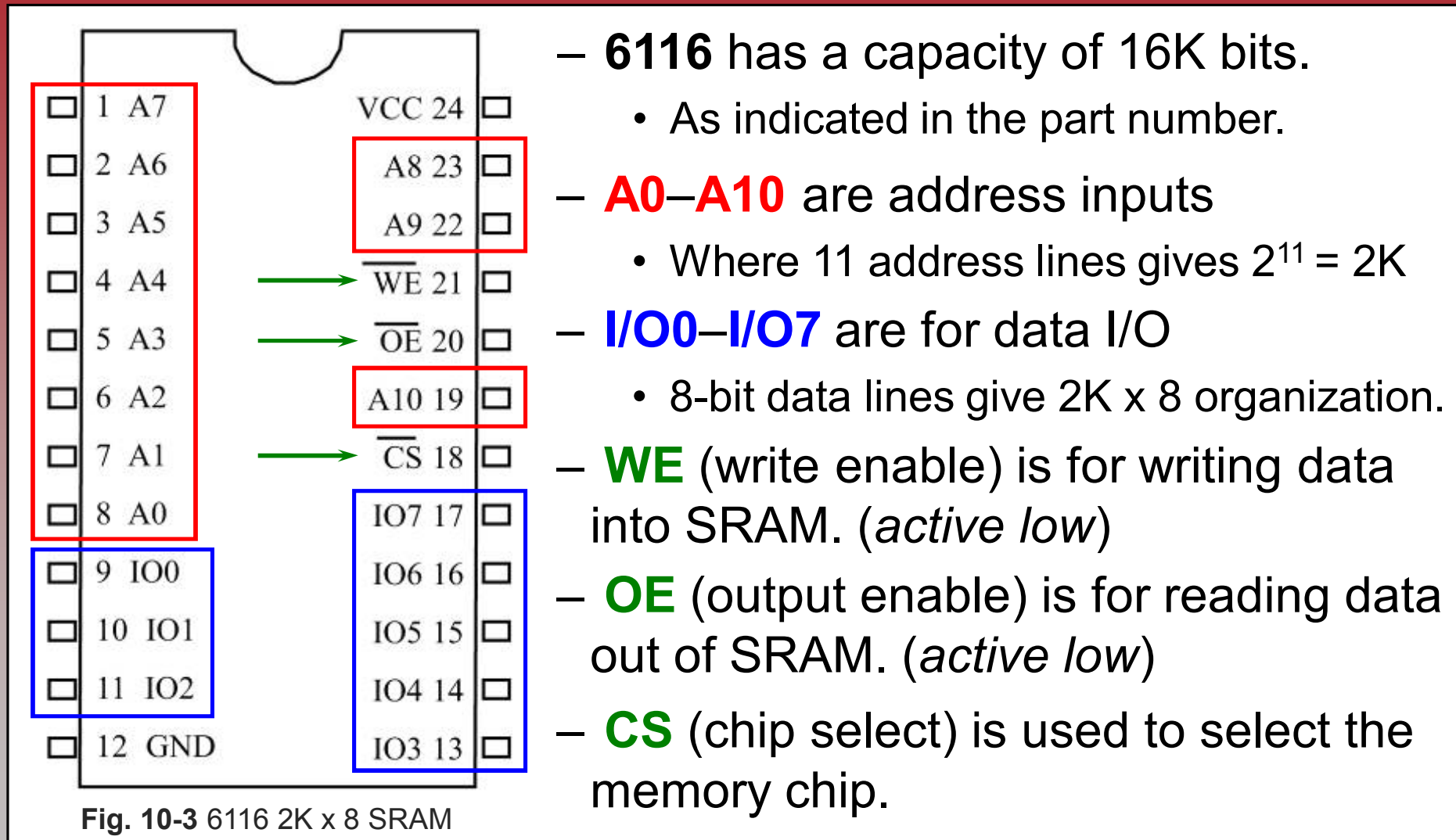
# 10.1: SEMICONDUCTOR MEMORIES

## SRAM static RAM

- Storage cells in static RAM memory are made of flip-flops & do not require refreshing to keep data.
- Each cell requires at least 6 transistors to build.
  - Each cell holds only 1 bit of data.
    - Recently 4-transistor cells have been made, still too many.
- 4-transistor cells, plus use of CMOS technology has given led to a high-capacity SRAM.
  - Still far below DRAM capacity.
- Table 10-3 shows some examples of SRAM.
  - See page 264.

# 10.1: SEMICONDUCTOR MEMORIES

## SRAM 6116 pinouts



# 10.1: SEMICONDUCTOR MEMORIES

## SRAM 6116 diagram

The functional block diagram for the 6116 SRAM

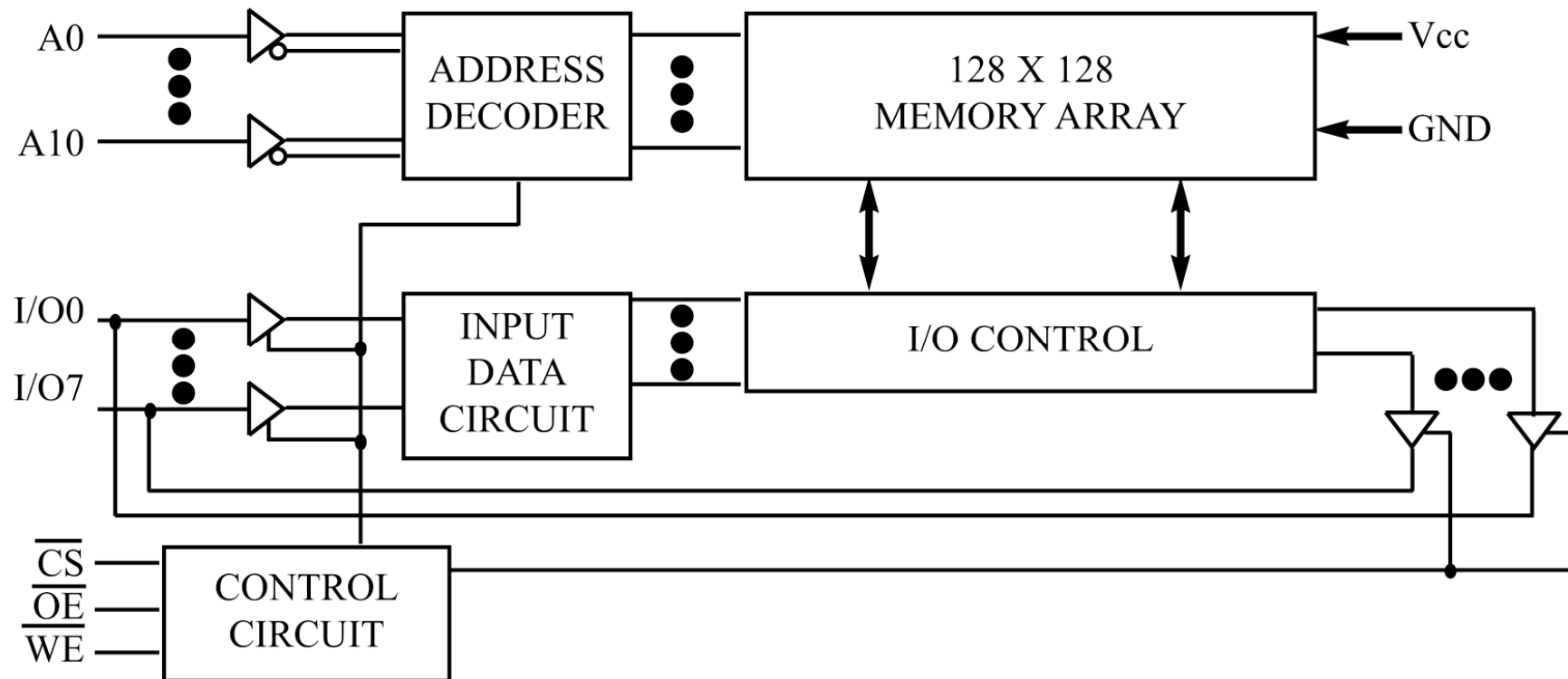


Fig. 10-4 6116 Functional Diagram



# 10.1: SEMICONDUCTOR MEMORIES

## SRAM data write steps

- 1. Provide the addresses to pins **A0–A10**.
- 2. Activate the **CS** pin.
- 3. Make **WE** = 0 while **RD** = 1.
- 4. Provide the data to pins **I/O0–I/O7**.
- 5. Make **CS** = 1 and data will be written into SRAM on the positive edge of the **CS** signal.

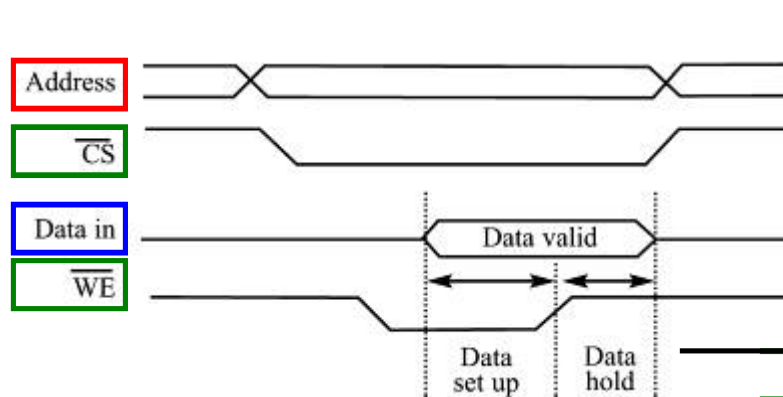


Fig. 10-5 Memory Write Timing for SRAM

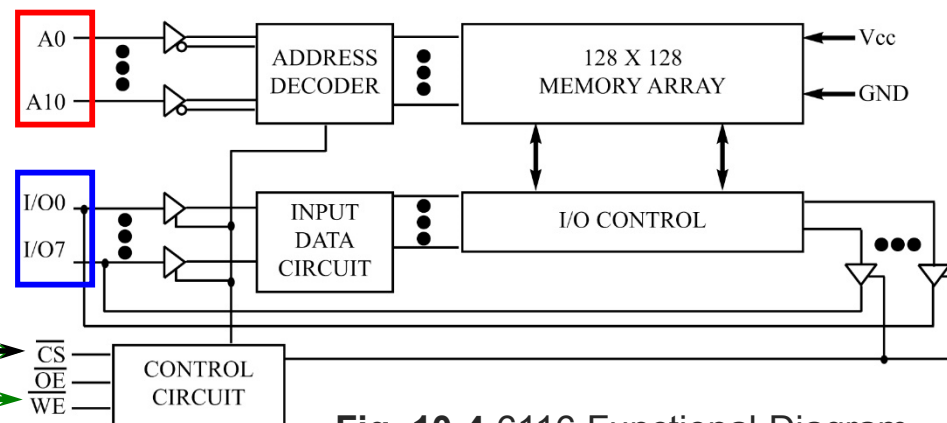


Fig. 10-4 6116 Functional Diagram

# 10.1: SEMICONDUCTOR MEMORIES

## SRAM data read steps

- 1. Provide the addresses to pins **A0–A10**, the start of the access time ( **$t_{AA}$** ).
- 2. Activate the **CS** pin.
- 3. While **WE** = 1 (or RD = 0), a high-to-low pulse on the **OE** pin will read the data out of the chip.

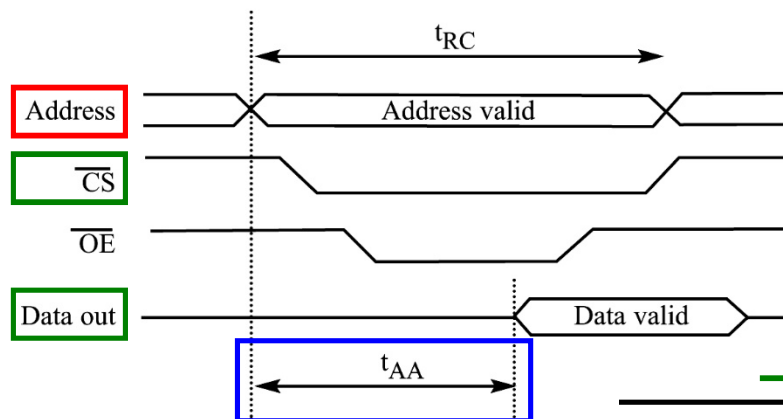


Fig. 10-6 Memory Read Timing for SRAM

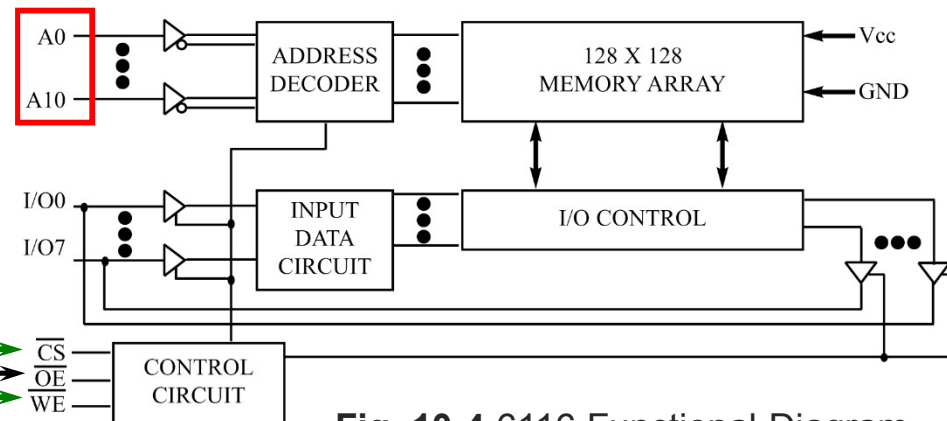


Fig. 10-4 6116 Functional Diagram

# 10.1: SEMICONDUCTOR MEMORIES

## SRAM 6116 access & read time

Access time,  $t_{AA}$ , is measured as time elapsed from the moment an address is provided to the address pins to the moment data is available at the pins. Access time for a 6116 chip can vary from **100 ns** to **15 ns**.

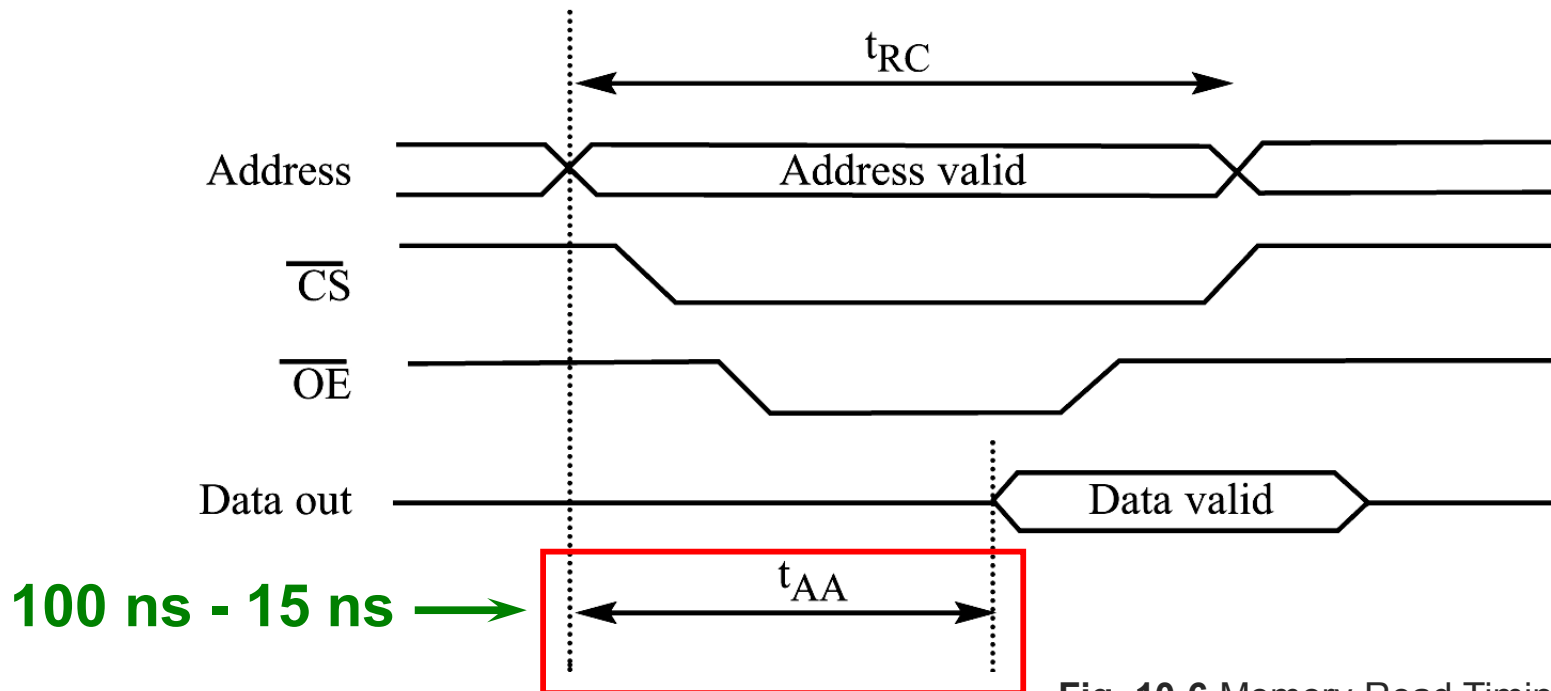


Fig. 10-6 Memory Read Timing for SRAM

# 10.1: SEMICONDUCTOR MEMORIES

## SRAM 6116 access & read time

Read cycle time,  $t_{RC}$ , is defined as the minimum amount of time required to read one byte of data, that is, from the moment the address of the byte is applied, to the moment the next read operation can begin.

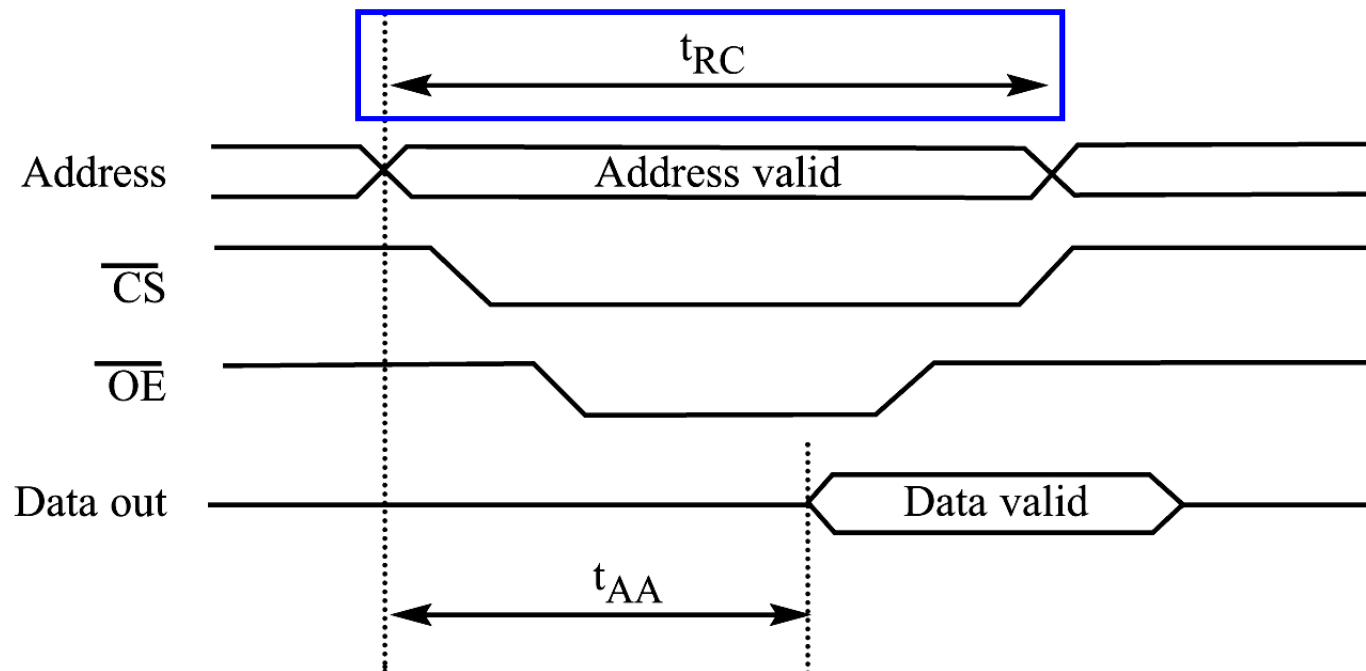


Fig. 10-6 Memory Read Timing for SRAM

# 10.1: SEMICONDUCTOR MEMORIES

## SRAM 6116 access & read time

In SRAM for which  $t_{AA} = 100 \text{ ns}$ ,  $t_{RC}$  is also  $100 \text{ ns}$ , which implies the contents of consecutive addresses can be read with each taking no more than 100 ns, hence, in SRAM and ROM:  
 $t_{AA} = t_{RC}$ .

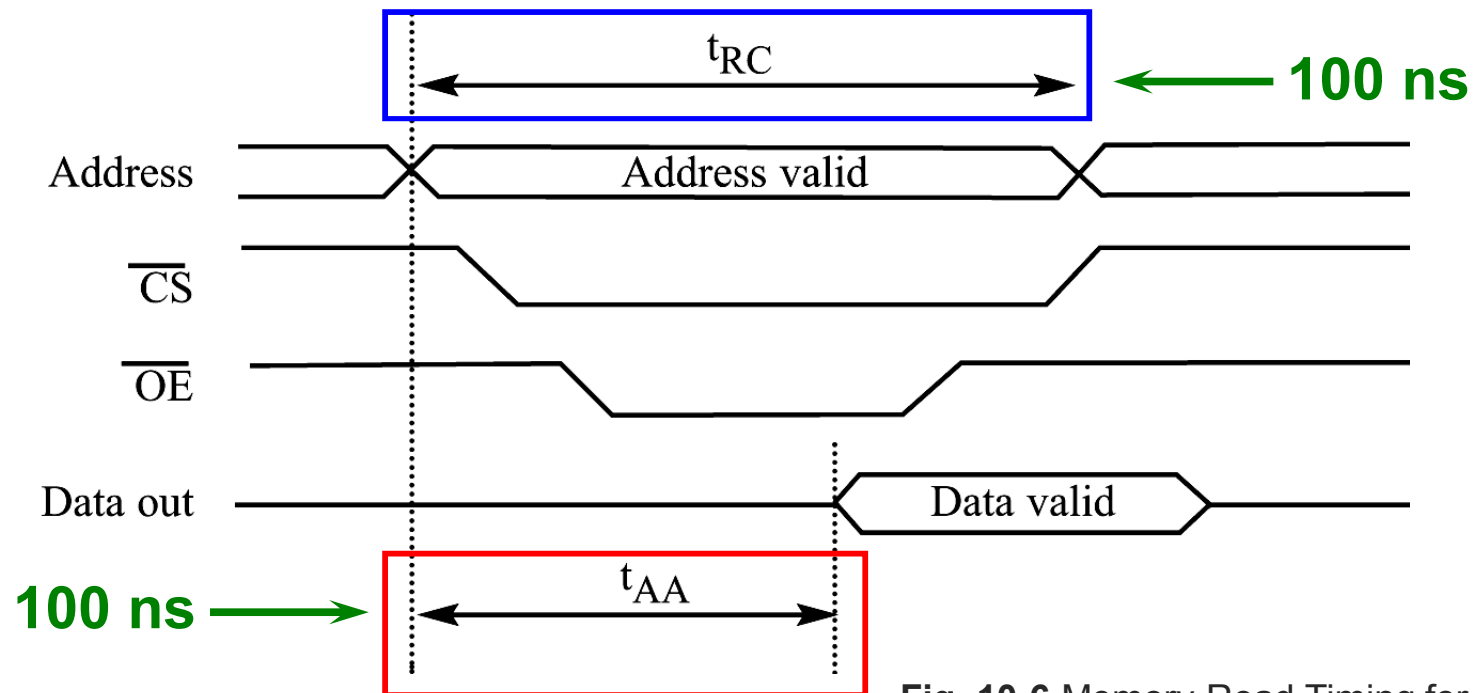


Fig. 10-6 Memory Read Timing for SRAM

# 10.1: SEMICONDUCTOR MEMORIES

## DRAM dynamic RAM

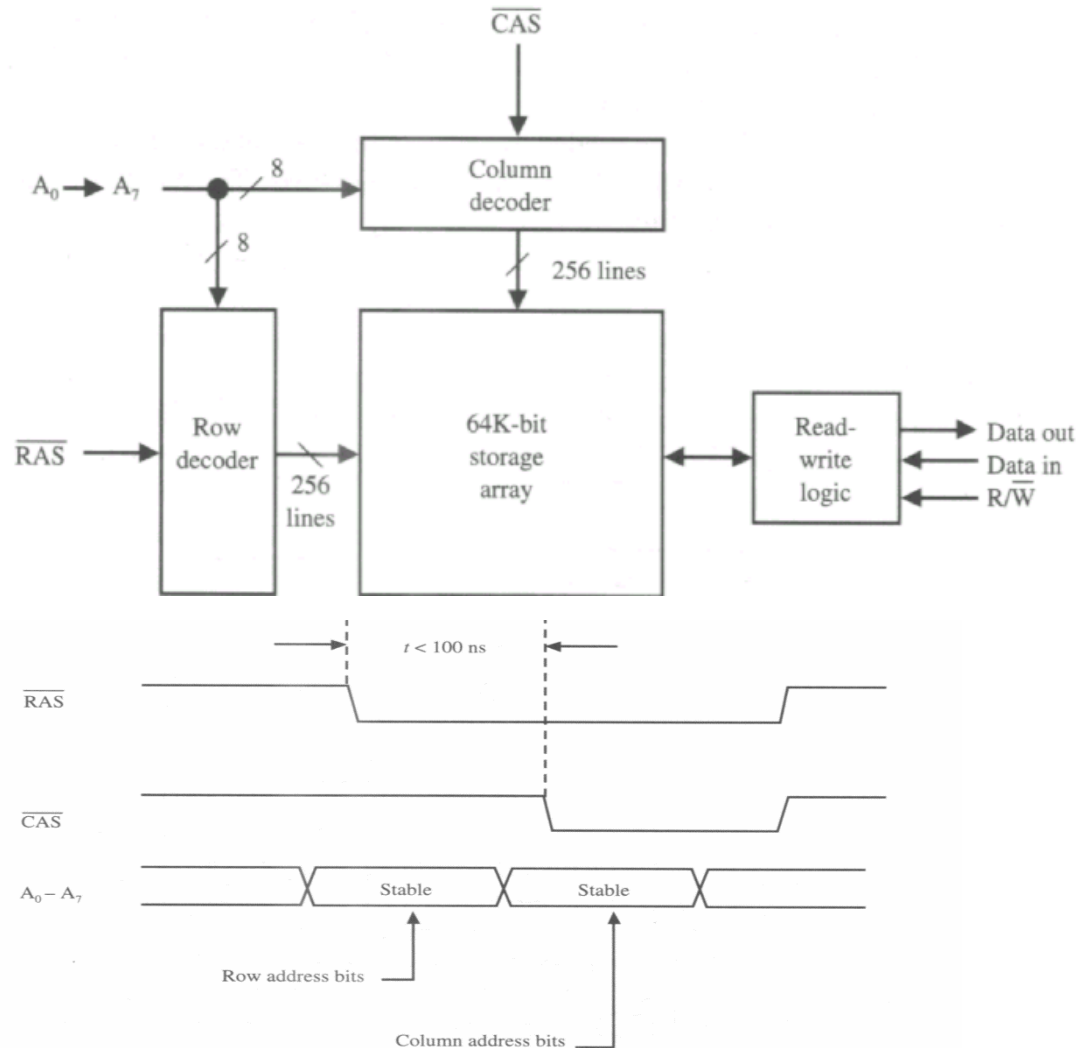
- In 1970, Intel Corporation introduced the first dynamic RAM (random access memory).
  - Density (capacity) was 1024 bits.
    - It used a capacitor to store each bit.
  - After the 1K-bit (1024) chip came the 4K-bit in 1973.
    - Advancing steadily, until the 256M DRAM chip in the 1990s.
- Using a capacitor to store data reduces the total number of transistors needed to build the cell.
  - Use of capacitors as storage cells in DRAM results in a much smaller net memory cell size.
    - They require constant refreshing due to leakage.

## 10.1: SEMICONDUCTOR MEMORIES

### DRAM advantages/disadvantages/terms

- Major advantages are high density (capacity), cheaper cost & lower power consumption per bit.
- The disadvantage is that it must be refreshed periodically, as the capacitor cell loses its charge.
  - While it is being refreshed, the data *cannot* be accessed.
- When referring to IC memory chips, capacity is always assumed to be in *bits*.
  - A 1G chip means a 1-*gigabit* chip and a 256M chip means a 256M-*bit* chip.

# DRAM



- In DRAM, the 8 address lines are latched accordingly by the strobe of the RAS and CAS signals.
- For example: To load a 16 bit address into the DRAM 8 bits of the address are first latched by pulling RAS low, then other 8 bits are presented to  $A_0$ - $A_7$  and CAS is pulled low.



# DRAM Addressing

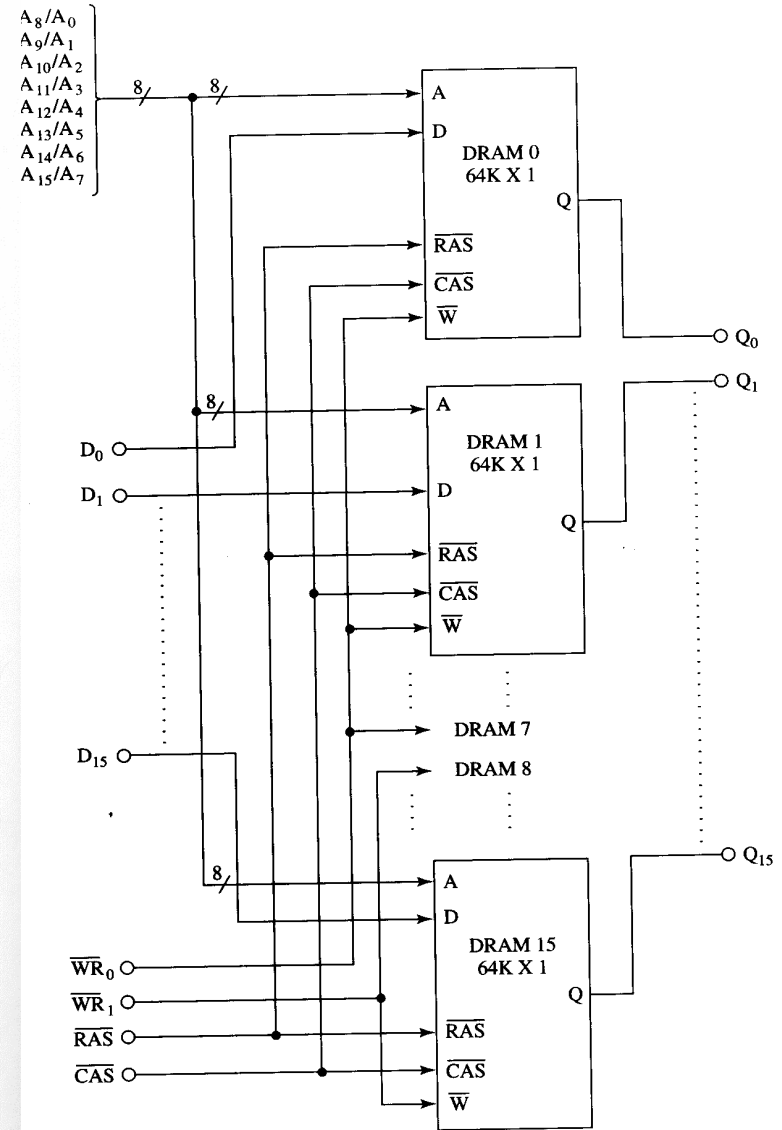
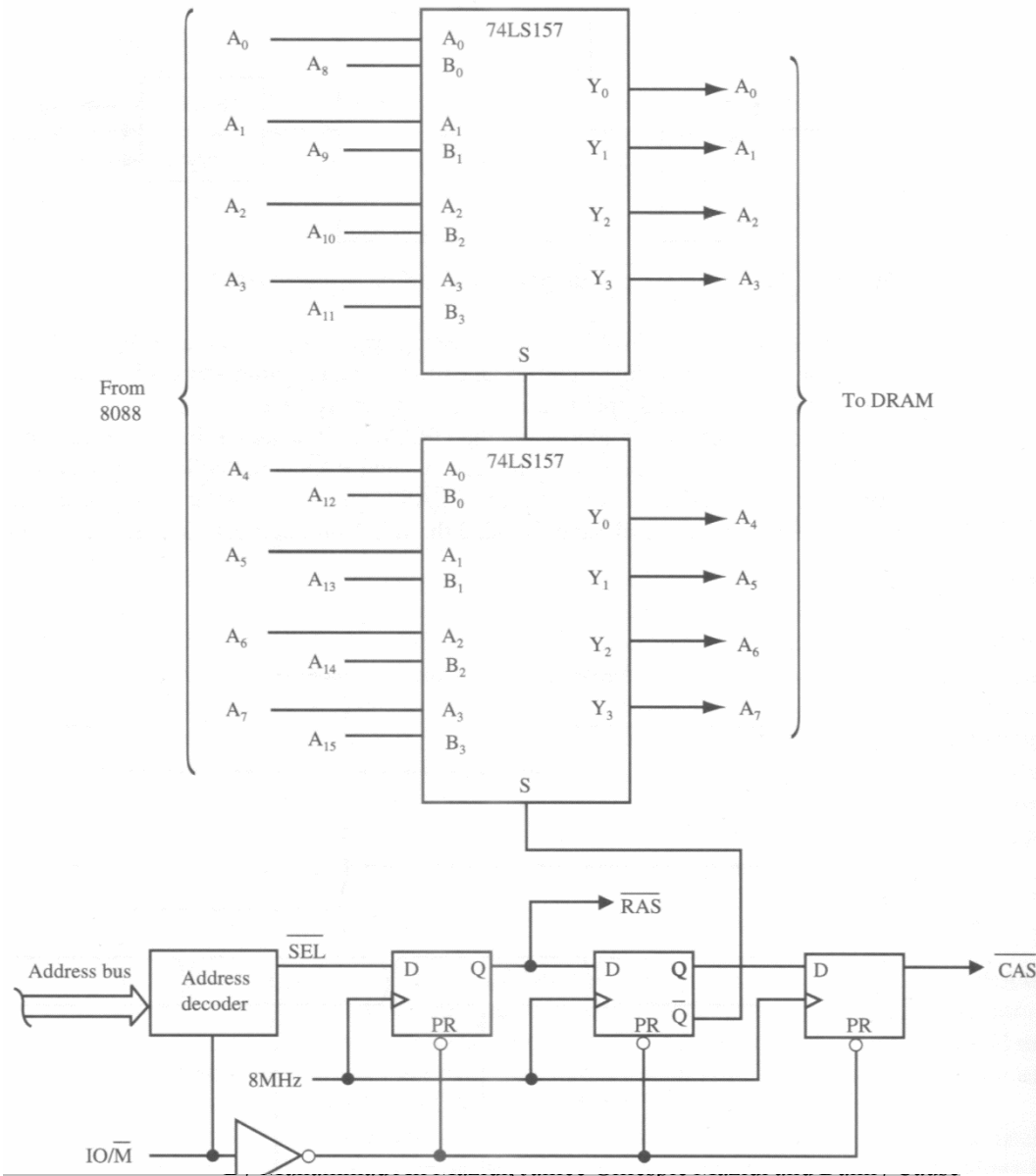


Figure 9-22 64K × 16-bit DRAM circuit.

# 10.1: SEMICONDUCTOR MEMORIES

## DRAM packaging issues

- It is difficult to pack large numbers of cells into single DRAM chips, with normal numbers of address pins.
  - A 64K-bit chip (64K x 1) must have 16 address lines and one data line, requiring 16 pins to send in the address.
    - In addition to VCC power, ground & read/write control pins.
  - Multiplexing/demultiplexing reduces the number of pins.
    - Addresses are split & each half is sent through the same pins.
- Internally, the DRAM structure is divided into a square of rows and columns.
  - The first half of the address is called the *row*.
  - The second half is called the *column*.

# 10.1: SEMICONDUCTOR MEMORIES

## DRAM packaging issues

- In a 64K x 1 organization, the first half of the address is sent through pins **A0–A7**.
  - Internal latches grab the first half.
    - Using **RAS** (row address strobe)
- The second address half is sent through the same pins
  - Activating **CAS** (column address strobe), latches the second half.
- 8 address pins, plus **RAS** & **CAS** make a total of 10 pins
  - Instead of 16, without multiplexing.

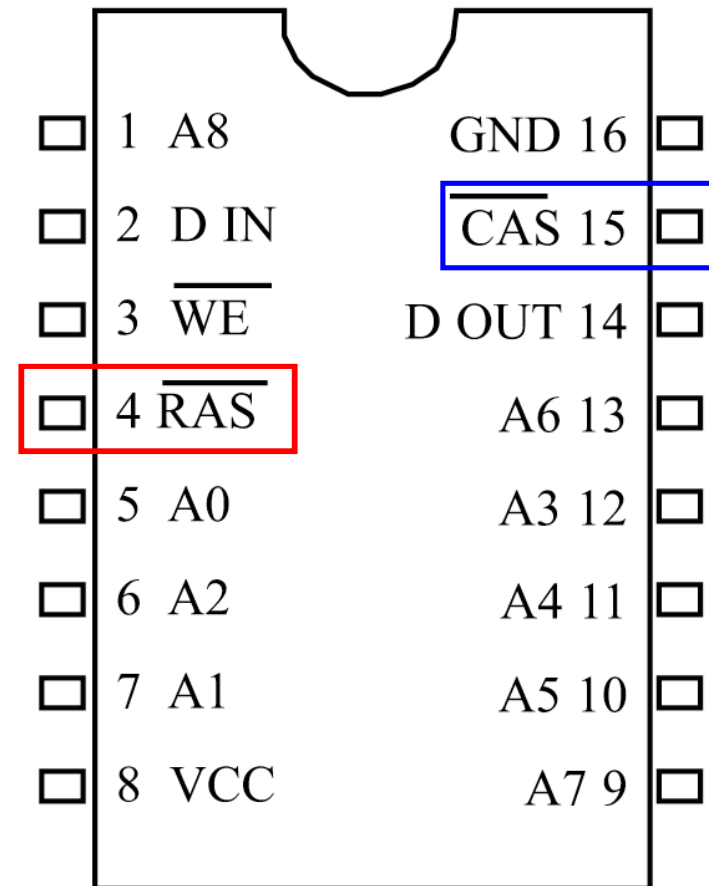


Fig. 10-7 256K × 1 DRAM

# 10.1: SEMICONDUCTOR MEMORIES

## DRAM packaging issues

- There must be a 2-by-1 multiplexer outside the DRAM chip, which has its own internal demultiplexer.
  - To access a bit of data from, both row & column address must be provided.
- The **WE** (write enable) pin is for read and write actions.

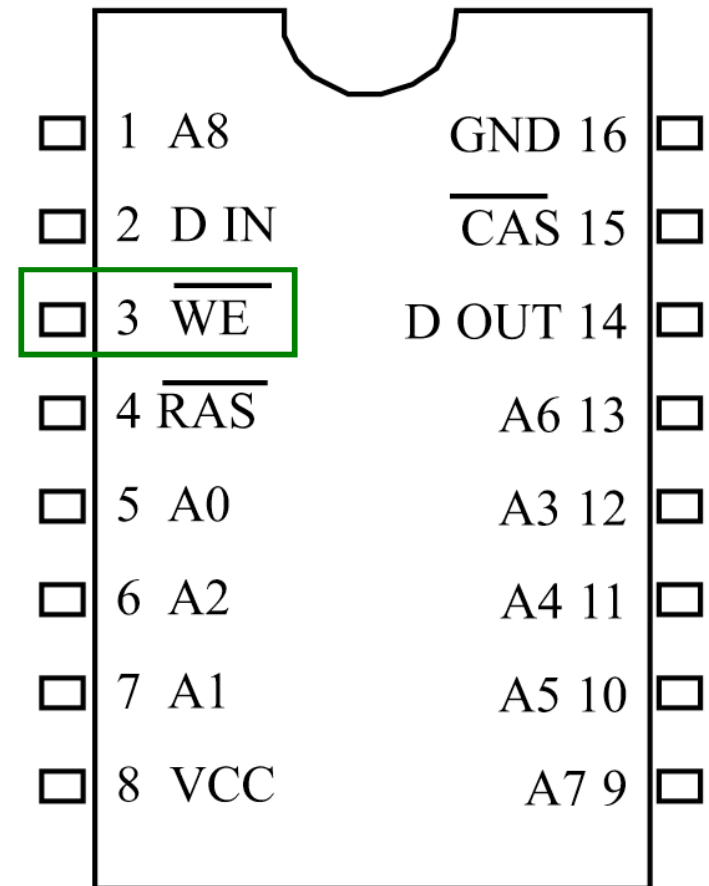
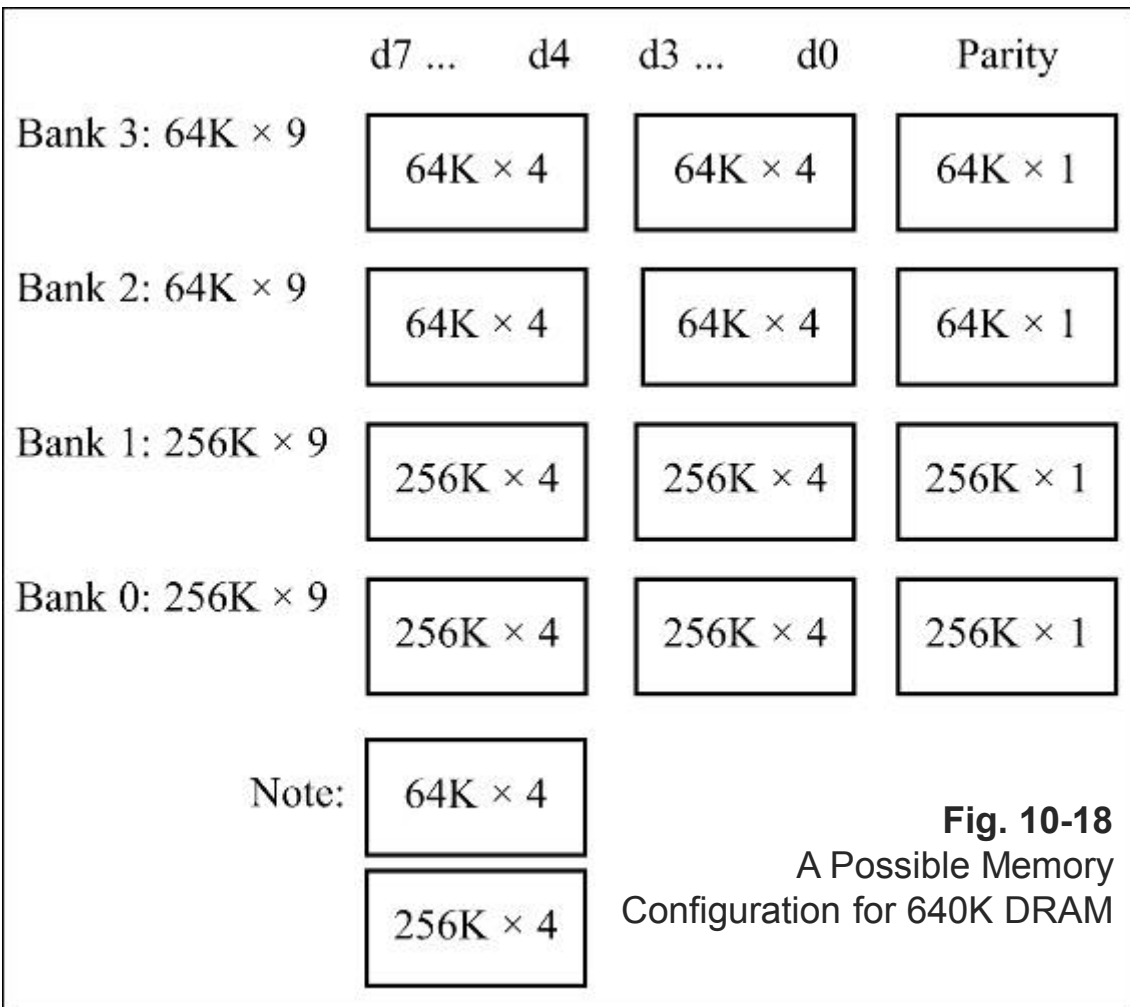


Fig. 10-7 256K × 1 DRAM

# 10.4: DATA INTEGRITY IN RAM AND ROM

## DRAM memory banks

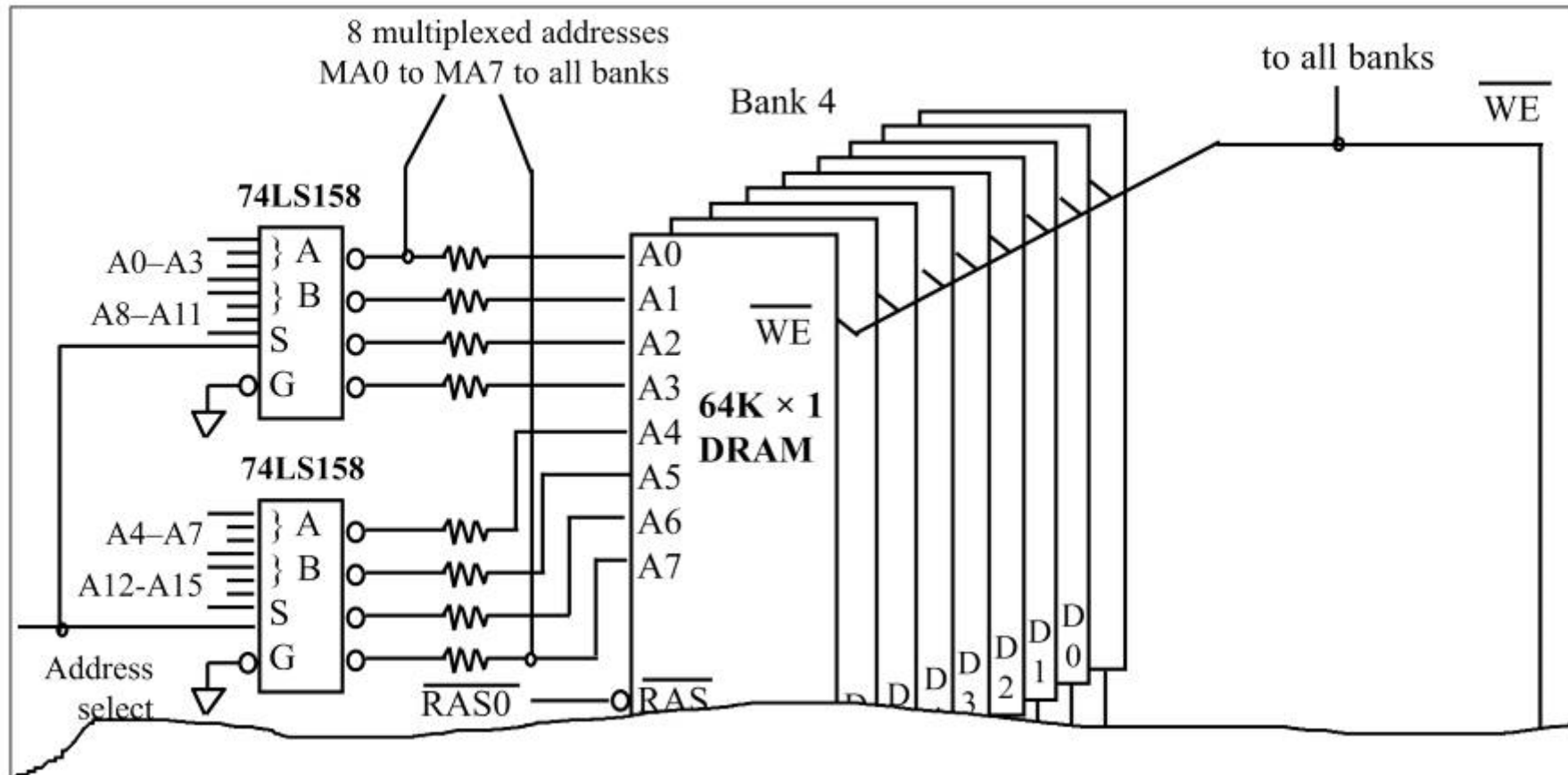
64K DRAM can be arranged as one bank of 8 chips of 64K x 1, or 4 banks of 16K x 1 organization. Note the extra bit for every byte of data, to store the parity bit. Every bank requires an extra chip of x 1 organization for parity check.



# 10.4: DATA INTEGRITY IN RAM AND ROM

## DRAM memory banks

DRAM design/parity bit circuitry for a bank of DRAM.



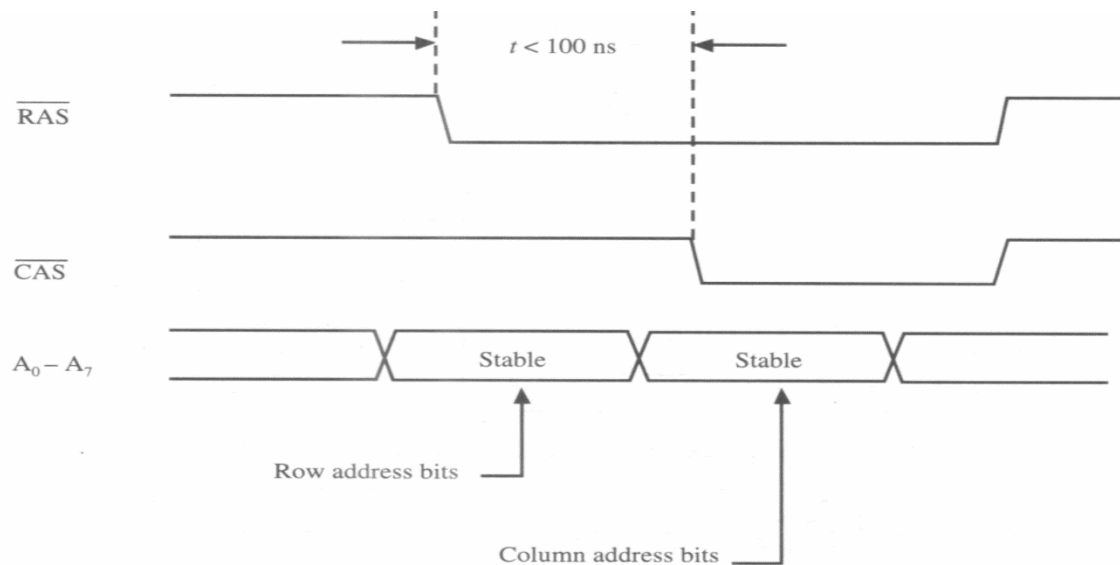
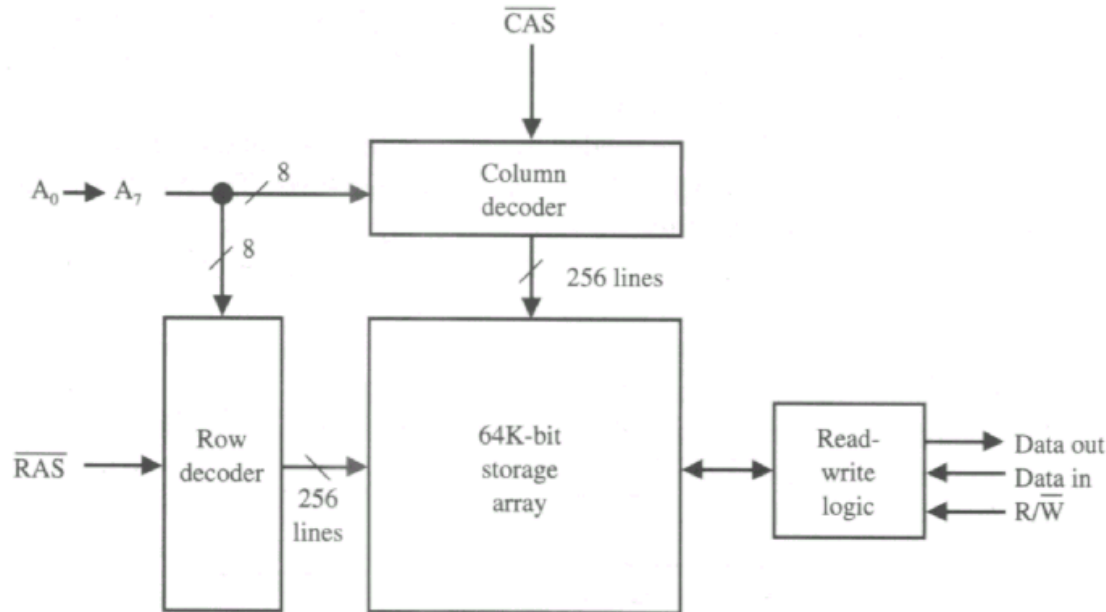
**See the entire diagram on page 276 of your textbook.**

## 10.4: DATA INTEGRITY IN RAM AND ROM

### DRAM memory banks

- Observations about Fig. 10-19:
  - The output of multiplexer addresses MA0–MA7 will go to all the banks.
  - Memory data MD0–MD7 and memory data parity MDP will go to all the banks.
  - 74LS245 buffers the data bus MD0–MD7, also boosts it to drive all DRAM inputs.

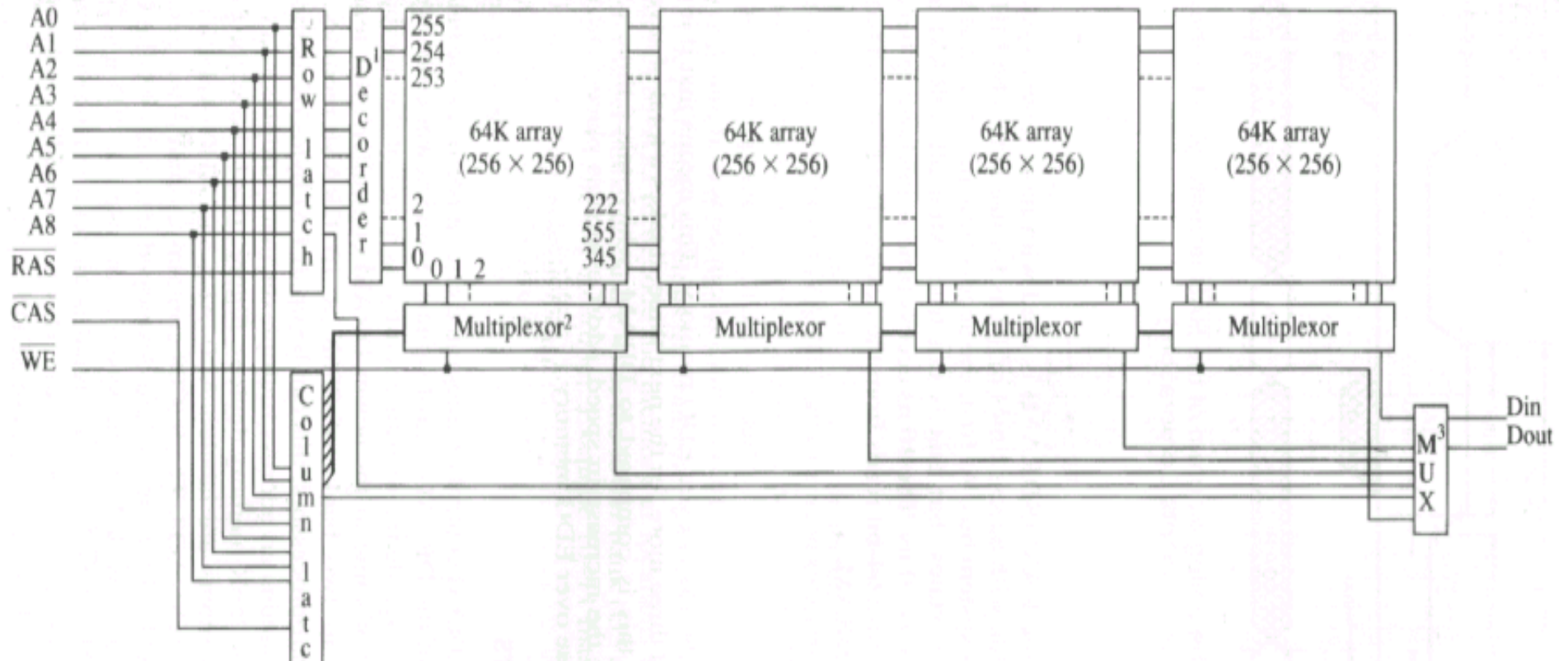
# DRAM



- In DRAM, the 8 address lines are latched accordingly by the strobe of the RAS and CAS signals.
- For example: To load a 16 bit address into the DRAM 8 bits of the address are first latched by pulling RAS low, then other 8 bits are presented to A0-A7 and CAS is pulled low.

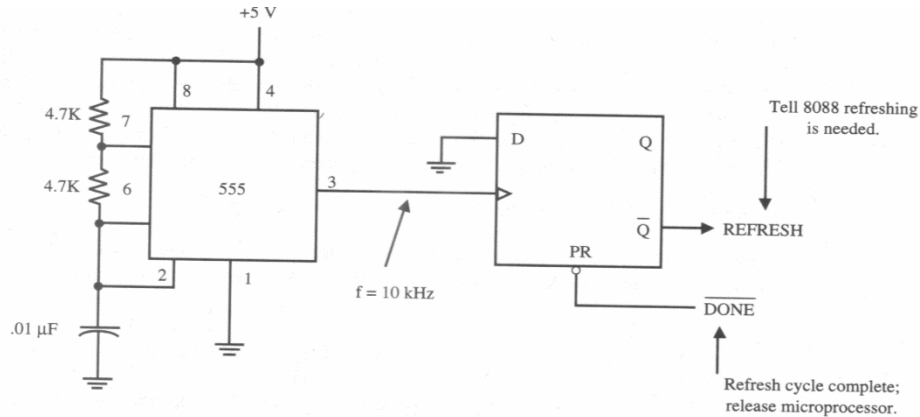


# DRAM Internal

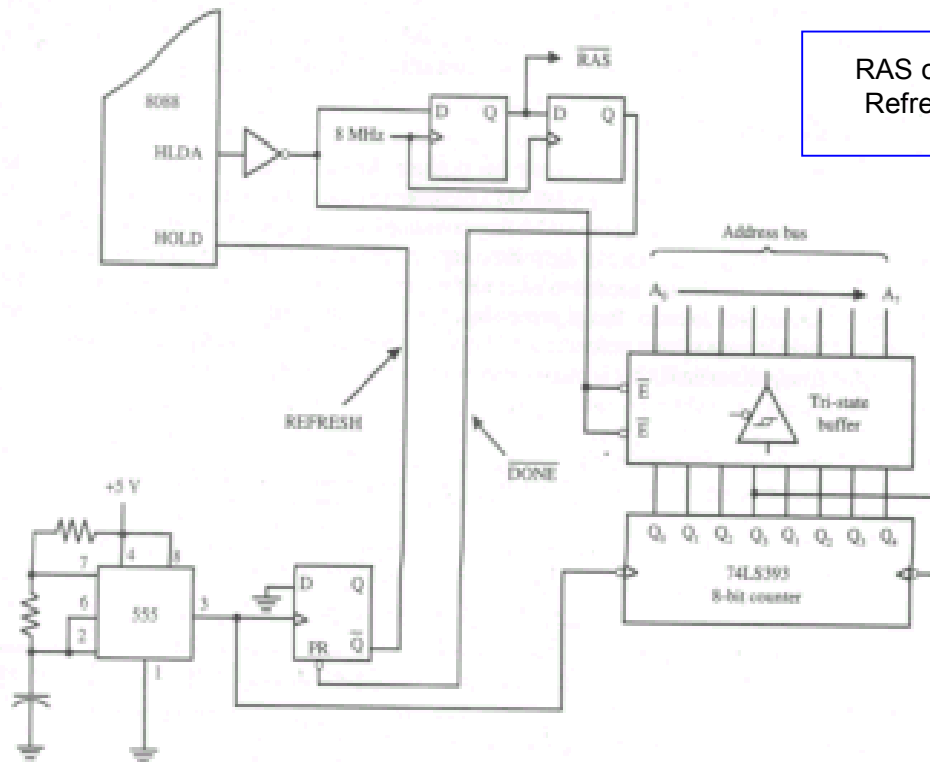


Refresh time example:  
 For a 256K X 1 DRAM with 256 rows, a refresh must occur every 15.6us (4ms/256).  
 For the 8086, a read or write occurs every 800ns.  
 This allows **19** memory reads/writes per refresh or **5%** of the time.

# DRAM Refresh



RAS only Refresh



- DRAM can be refreshed by an external circuitry including an 8 bit counter
- HOLD/HLDA used
- Only the columns of the matrix (256 x 256 for a 64K bit matrix is needed to be refreshed.
- The refresh rate can be adjusted using a 555 timer circuitry.

# DRAM in PC

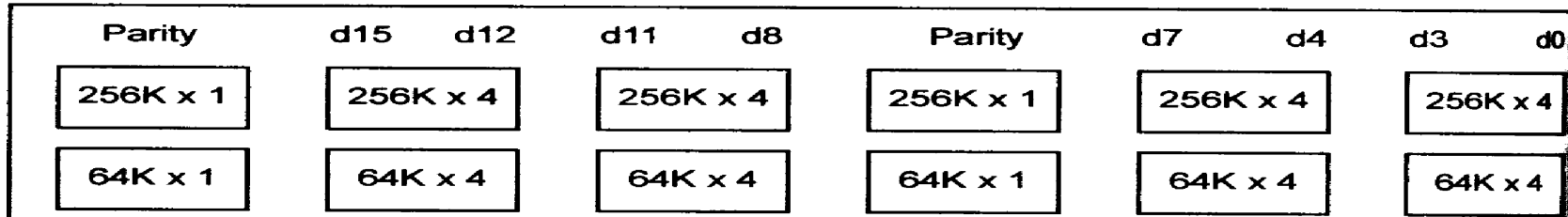


Figure 10-21. 640K Bytes of DRAM with odd and even banks designation

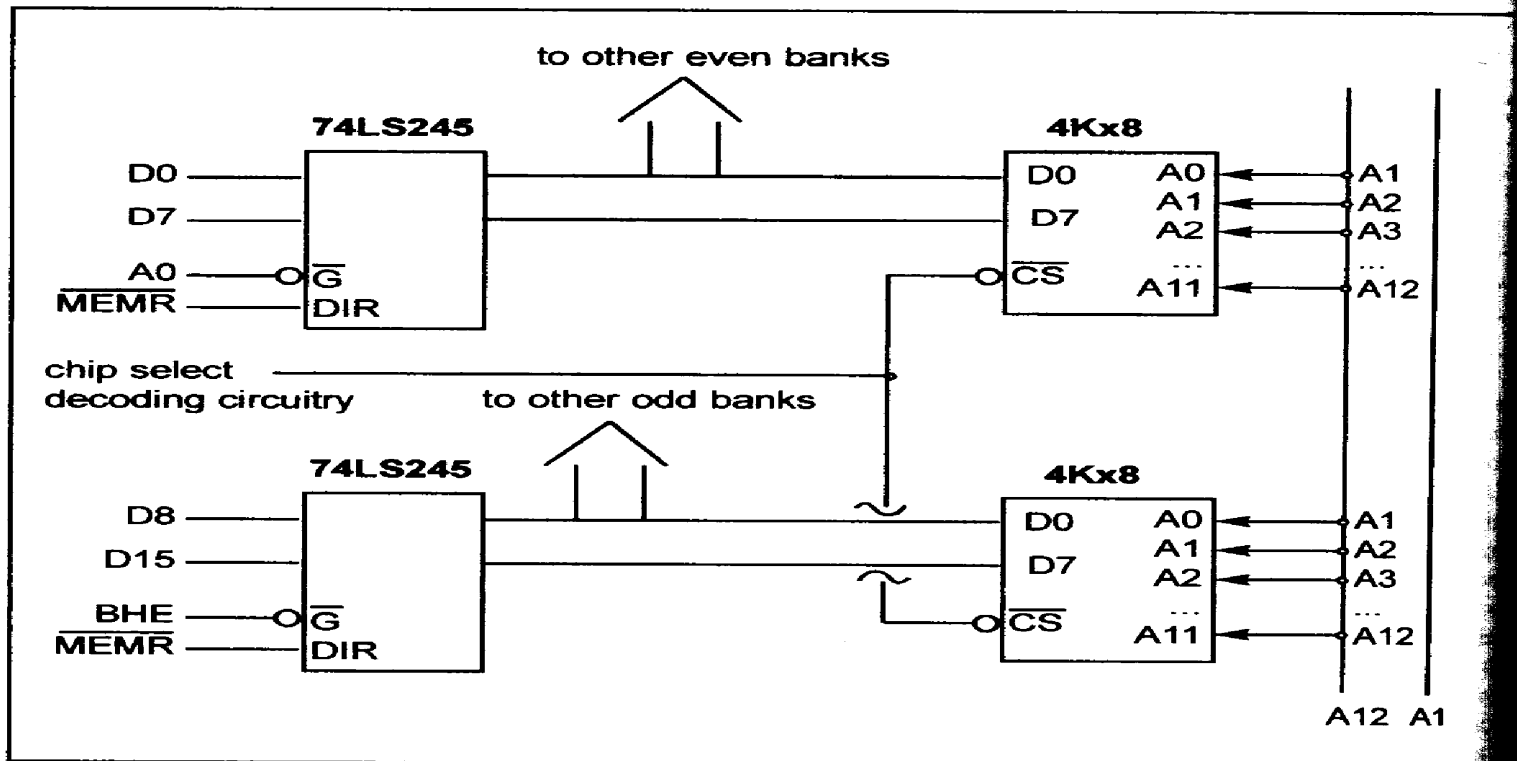


Figure 10-22. 16-bit Data Connection in the 80286 System

# 10.1: SEMICONDUCTOR MEMORIES

## DRAM, SRAM, ROM organizations

- Organizations for SRAMs & ROMs are always x 8.
  - DRAM can have x 1, x 4, x 8, or x 16 organizations.
- In some memory chips (notably SRAM), the data pins are called I/O.
  - In some DRAMs, there are separate pins **Din** and **Dout**.
    - DRAMs with x1 organization are widely used for parity bit.

### Example 10-5

Discuss the number of pins set aside for addresses in each of the following memory chips.

(a) 16K × 4 DRAM

(b) 16K × 8 SRAM

#### Solution:

Since  $2^{14} = 16K$ :

(a) For DRAM we have 7 pins (A0–A6) for the address pins and 2 pins for RAS and CAS.

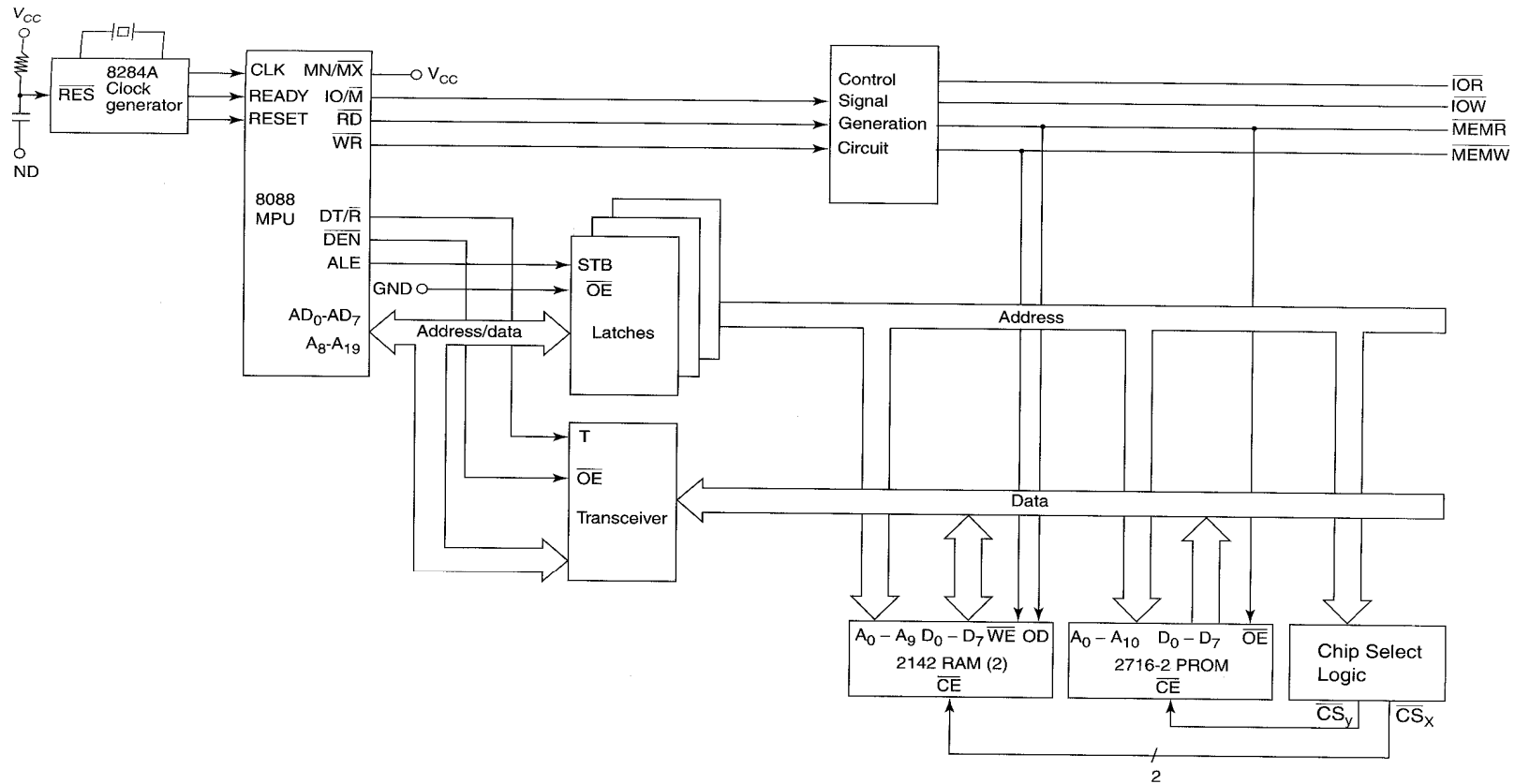
(b) For SRAM we have 14 pins (A0–A13) for address and no pins for RAS and CAS since they are associated only with DRAM.

# 10.1: SEMICONDUCTOR MEMORIES

## NV-RAM nonvolatile RAM

- Nonvolatile RAM allows the CPU to read & write to it & when power is turned off, contents are not lost.
- NV-RAM chip internal components:
  - 1. Extremely power-efficient (low power consumption) SRAM cells, built out of CMOS.
  - 2. Internal lithium battery as a backup energy source.
  - 3. Intelligent control circuitry.
    - To monitor the VCC pin, and automatically switch to the lithium battery on for loss of external power.
- NV-RAM is used to save x86 PC system setup.
  - Commonly referred to as CMOS RAM.

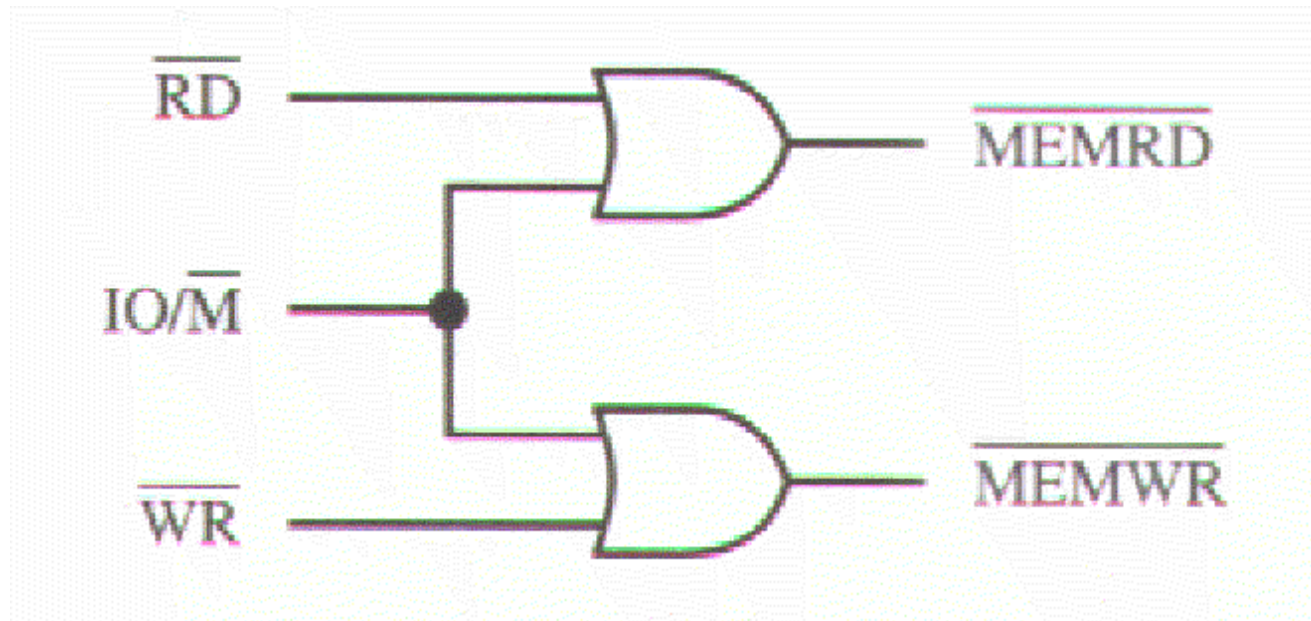
# Minmode 8088 Microcomputer system memory circuitry



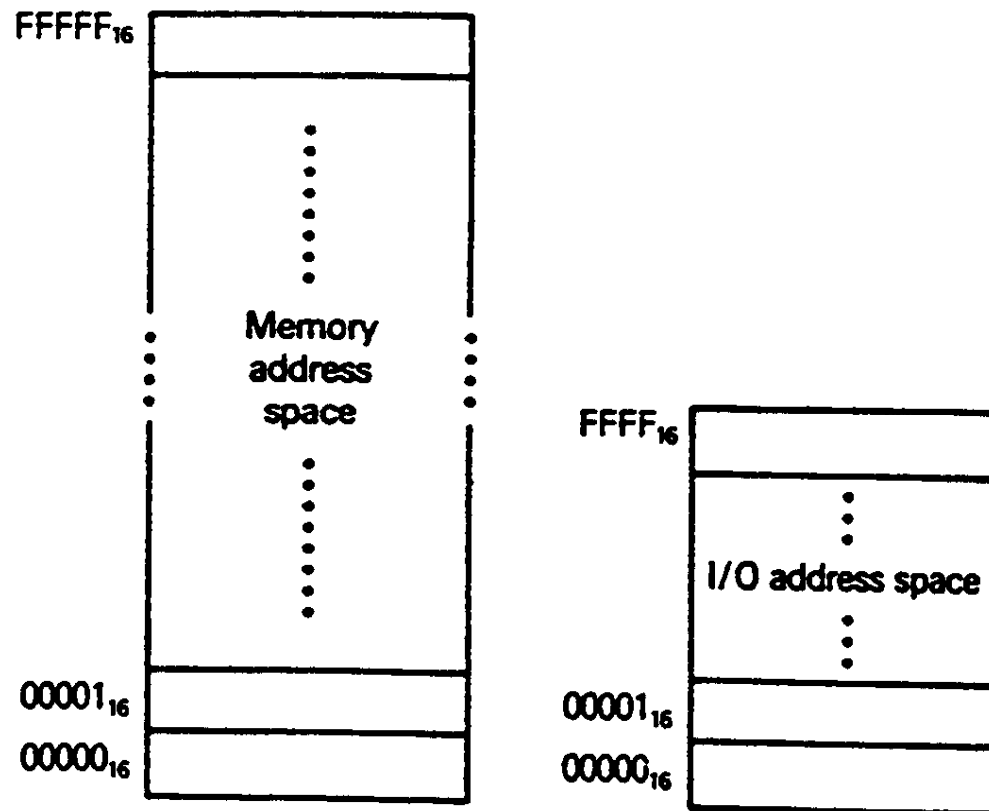
(a)

**Figure 9-37** (a) Minimum-mode 8088 system memory interface. (Reprinted with permission of Intel Corporation, Copyright/Intel Corp. 1981) (b) Minimum-mode 8086 system memory interface. (Reprinted with permission of Intel Corporation, Copyright/Intel Corp. 1979) (c) Maximum-mode 8088 system memory interface. (Reprinted with permission of Intel Corporation, Copyright/Intel Corp. 1981)

# Control Signal Generation Circuitry



# 8088 Memory and I/O address spaces



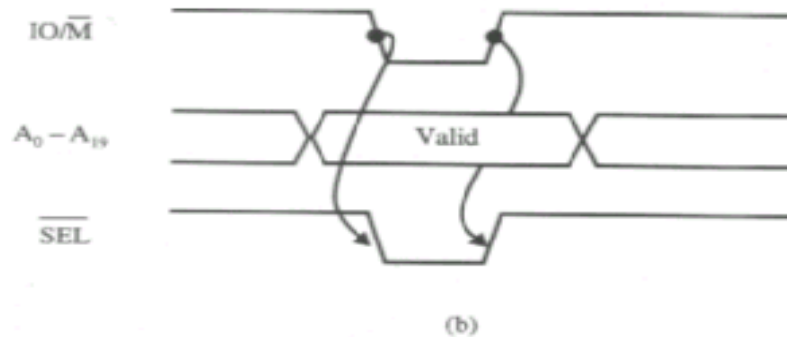
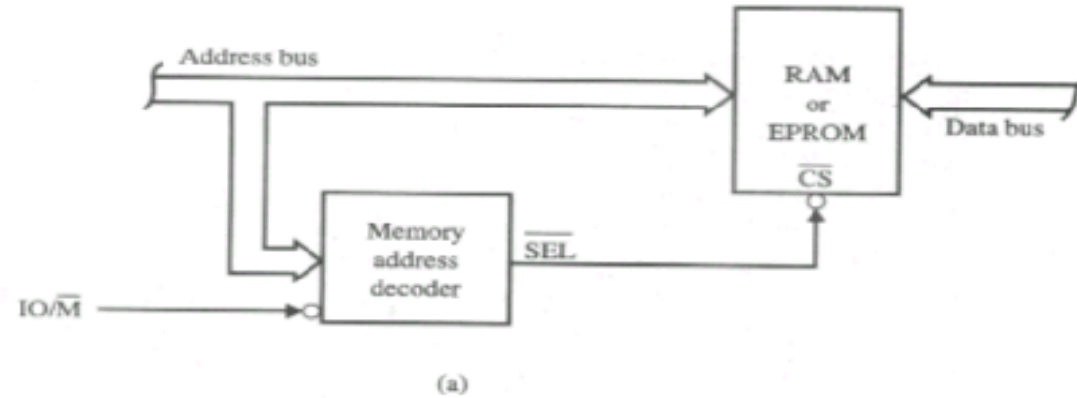
(a)

We first look at the memory addressing

**Figure 8-46** 8088/8086 memory and I/O address spaces.

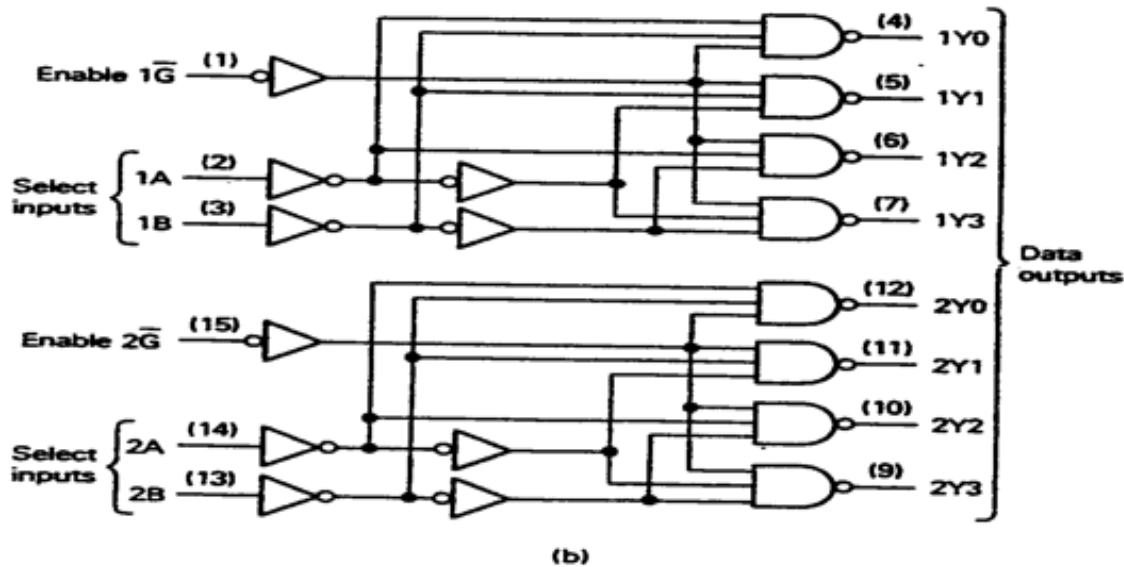
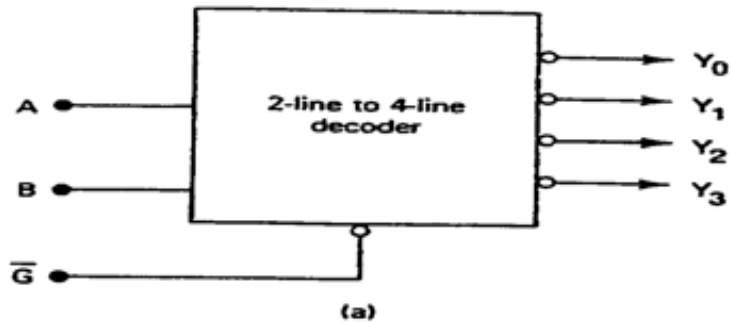


# Alternative Address Selection



An alternative to address selection.  
The Address is combined with IO/M signal for address selection. The RD/WR pin is now on the EPROM/RAM

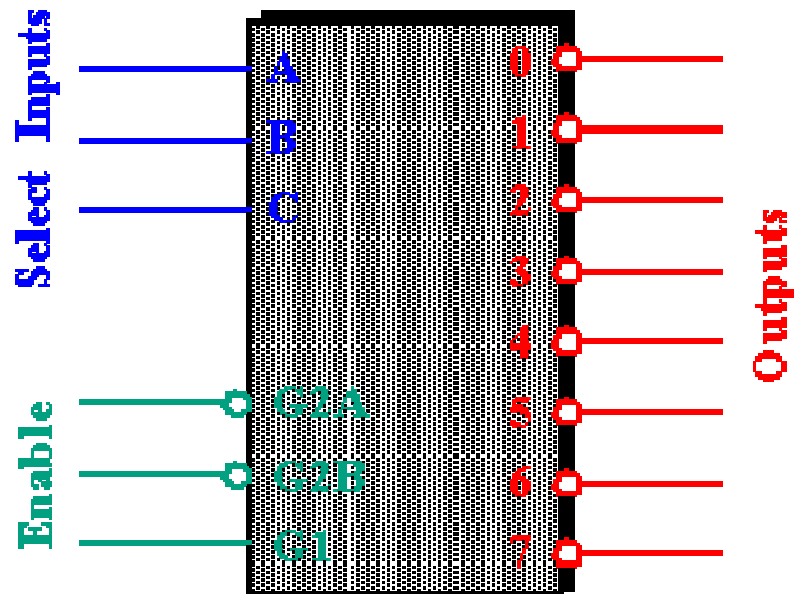
# 74F139 2-line to 4-line decoder



| INPUTS    |        |   | OUTPUTS |    |    |    |
|-----------|--------|---|---------|----|----|----|
| ENABLE    | SELECT |   | Y0      | Y1 | Y2 | Y3 |
| $\bar{Z}$ | B      | A |         |    |    |    |
| H         | X      | X | H       | H  | H  | H  |
| L         | L      | L | L       | H  | H  | H  |
| L         | L      | H | H       | L  | H  | H  |
| L         | H      | L | H       | H  | L  | H  |
| L         | H      | H | H       | H  | H  | L  |

(c)

# Memory Address Decoding



3-8 Decoder  
(for example: 74LS138)

| Inputs |     |    |        |   |   | Output |   |   |   |   |   |   |   |
|--------|-----|----|--------|---|---|--------|---|---|---|---|---|---|---|
| Enable |     |    | Select |   |   |        |   |   |   |   |   |   |   |
| G2A    | G2B | G1 | C      | B | A | 0      | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1      | X   | X  | X      | X | X | 1      | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X      | 1   | X  | X      | X | X | 1      | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X      | X   | 0  | X      | X | X | 1      | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0      | 0   | 1  | 0      | 0 | 0 | 0      | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0      | 0   | 1  | 0      | 0 | 1 | 1      | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0      | 0   | 1  | 0      | 1 | 0 | 1      | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0      | 0   | 1  | 0      | 1 | 1 | 1      | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0      | 0   | 1  | 1      | 0 | 0 | 1      | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0      | 0   | 1  | 1      | 1 | 0 | 1      | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0      | 0   | 1  | 1      | 1 | 1 | 1      | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

# Address Decoder Circuit

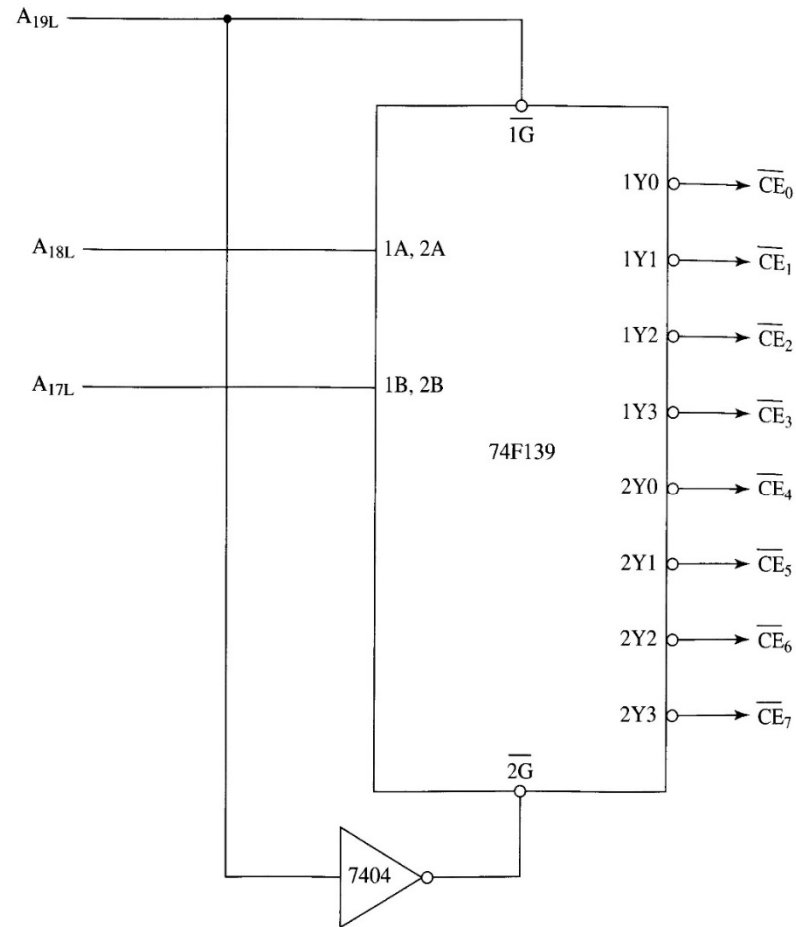
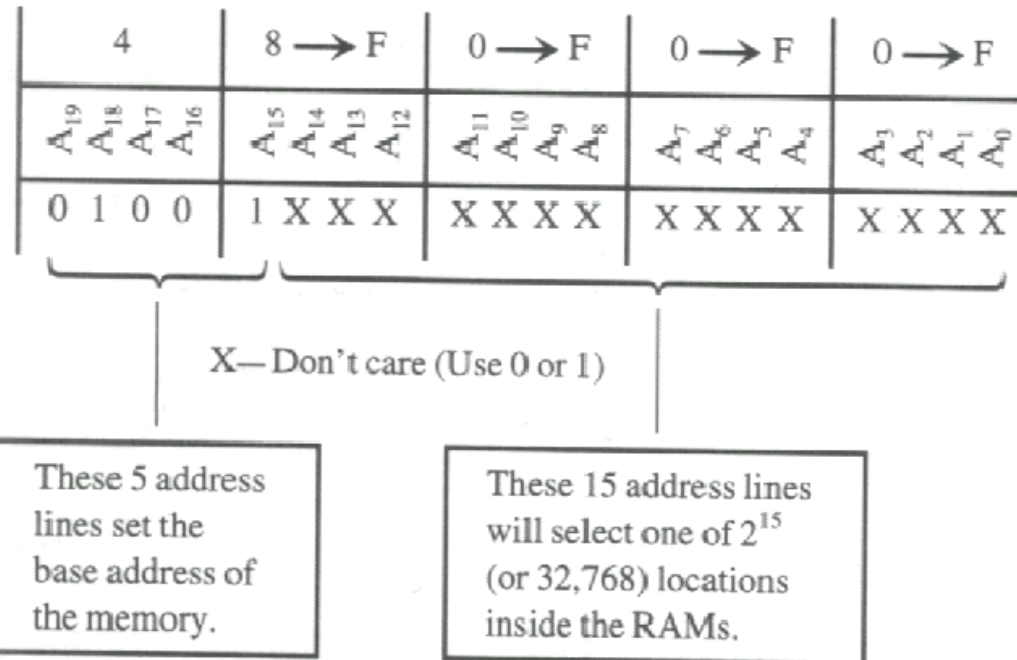


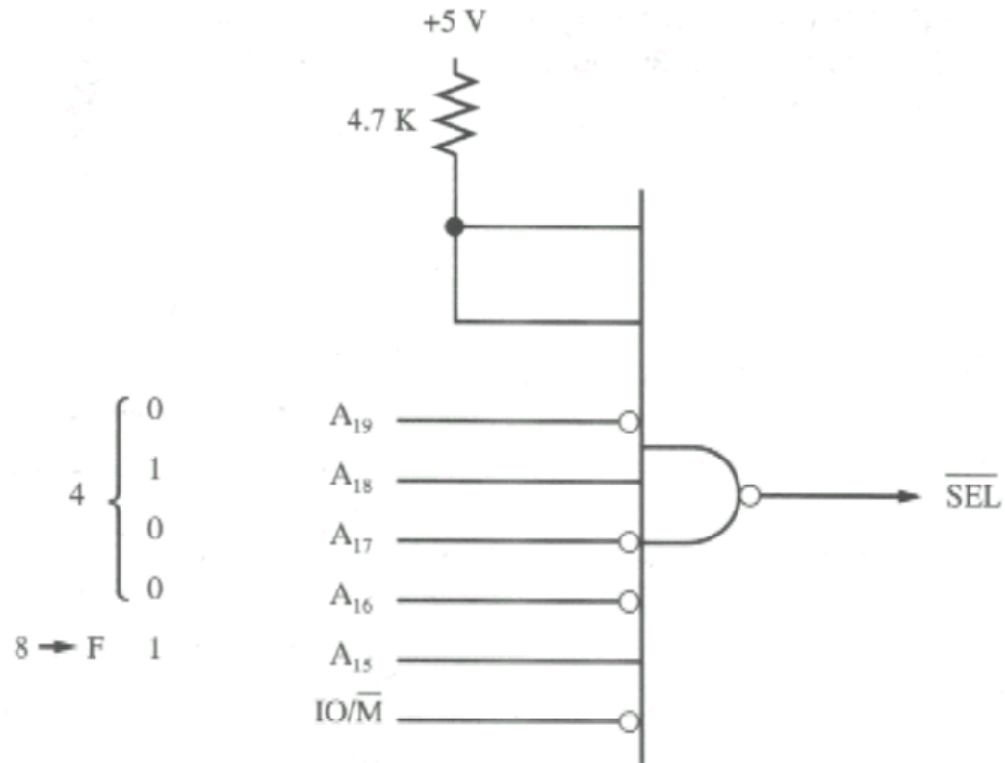
Figure 8-35 Address decoder circuit

# Example on Address Decoding

A circuit containing 32KB of RAM is to be interfaced to an 8088 based system, so that the first address of the RAM is at 48000H. What is the entire range of the RAM Address? How is the address bus used to enable the RAMs? What address lines should be used?

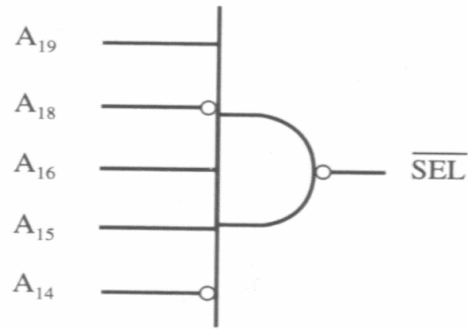


# Example on Address Decoding

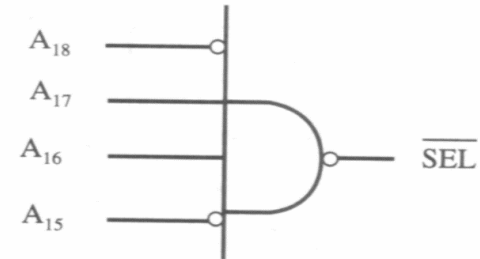


Memory Address Decoder for 48000 to 4FFFF Range

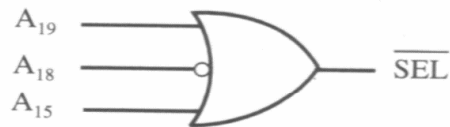
Examples: Find different addressing for CS (A0-A13 used by memories)



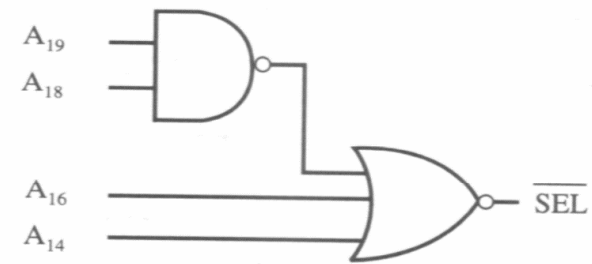
10x1 10/xx xxxx xxxxxxxx



x011 0x/xx xxxx xxxxxxxx



01xx x0/xx xxxx xxxxxxxx



(A19 and A18=0) or A16=1 or A14=1



xx1x xx/xx xxx x xxxxxxxx

## 10.2: MEMORY ADDRESS DECODING

### simple logic gate as address decoder

- Current system designs use CPLDs.  
(complex programmable logic devices)
  - Memory & address decoding circuitry are integrated into one programmable chip.
  - A task which *can* be performed with common logic gates.
    - NAND and 74LS138 chips act as decoders.



## 10.2: MEMORY ADDRESS DECODING

### simple logic gate as address decoder

- In connecting a memory chip to the CPU, the data bus is connected directly to the data pins of the memory.
  - Control signals **MEMR** & **MEMW** are connected to the **OE** & **WR** pins.

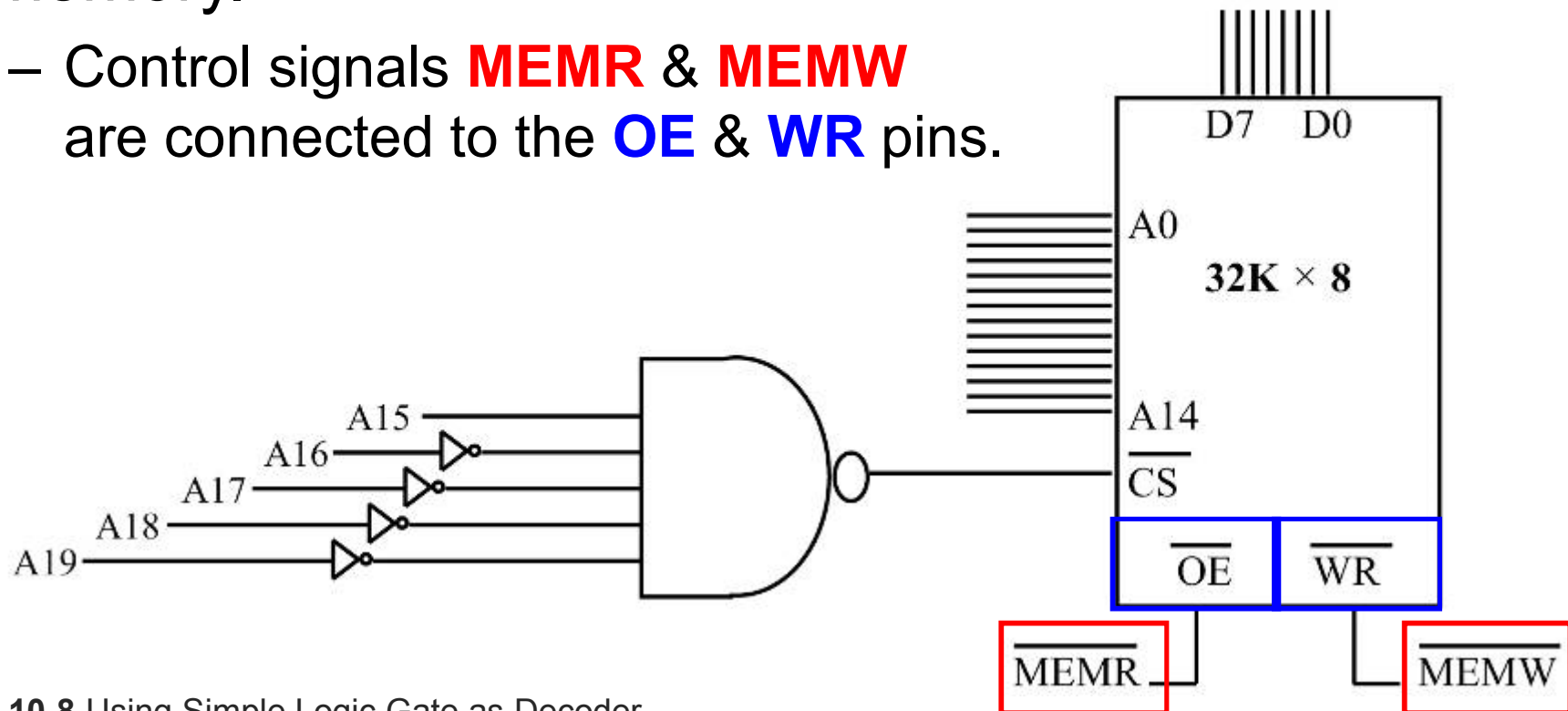


Fig. 10-8 Using Simple Logic Gate as Decoder

## 10.2: MEMORY ADDRESS DECODING

### simple logic gate as address decoder

- On the address buses, the lower bits of the address go directly to memory chip address pins.
  - The upper ones activate the **CS** pin of the memory chip.
    - **CS** pin, with **RD/WR** allows data flow in/out of the chip.
  - No data can be written into or read from the memory chip unless CS is activated.
- **CS** input is *active-low* and can be activated using simple logic gates, such as NAND and inverters.
  - For every block of memory, we need a NAND gate.
    - See Figs. 10-9 & 10-10 on page 266.

## 10.2: MEMORY ADDRESS DECODING

### simple logic gate as address decoder

- Example 10-6 shows the address range calculation for Fig. 10-10 on page 266.
  - Note the output of the NAND gate is *active-low*.
    - The CS pin is also *active-low* .

#### Example 10-6

Referring to Figure 10-10 we see that the memory chip has 64K bytes of space. Show the calculation that verifies that address range 90000 to 9FFFFH is comprised of 64K bytes.

#### Solution:

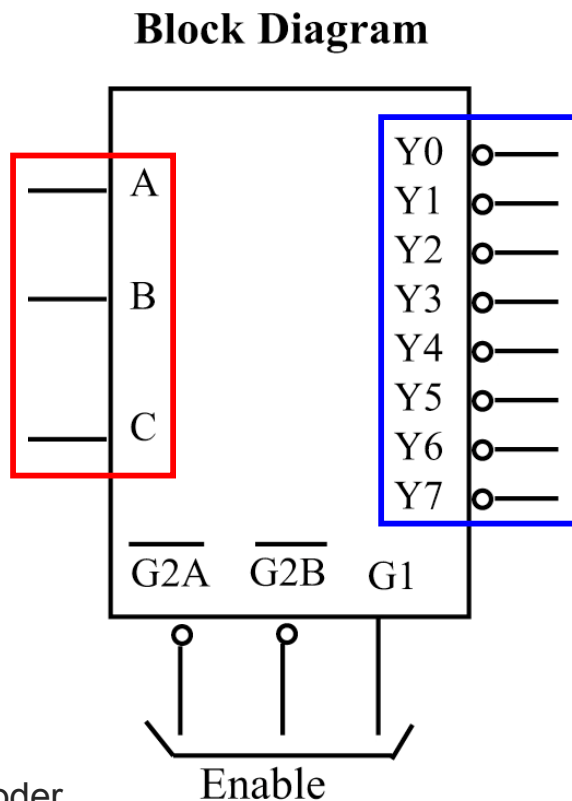
To calculate the total number of bytes for a given memory address range, subtract the two addresses and add 1 to get the total bytes in hex. Then the hex number is converted to decimal and divided by 1024 to get K bytes.

$$\begin{array}{r} 9FFFF \\ -90000 \\ \hline 0FFFF \end{array} \qquad \begin{array}{r} FFFF \\ + \quad 1 \\ \hline 10000 \text{ hex} = 65,536 \text{ decimal} = 64\text{K} \end{array}$$

# 10.2: MEMORY ADDRESS DECODING using the 74LS138 as decoder

- In the absence of CPLD or FPGA as address decoders, the 74LS138 chip is an excellent choice.

Three inputs:  
**A, B & C**,  
generate  
eight  
*active-low*  
outputs:  
**Y0–Y7.**



**Function Table**

| Inputs |        | Outputs |    |    |    |    |    |    |    |
|--------|--------|---------|----|----|----|----|----|----|----|
| Enable | Select |         |    |    |    |    |    |    |    |
| G1G2   | 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  |

Fig. 10-11 74LS138 Decoder

## 10.2: MEMORY ADDRESS DECODING using the 74LS138 as decoder

- The need for NAND and inverter gates is eliminated when using 74SL138.

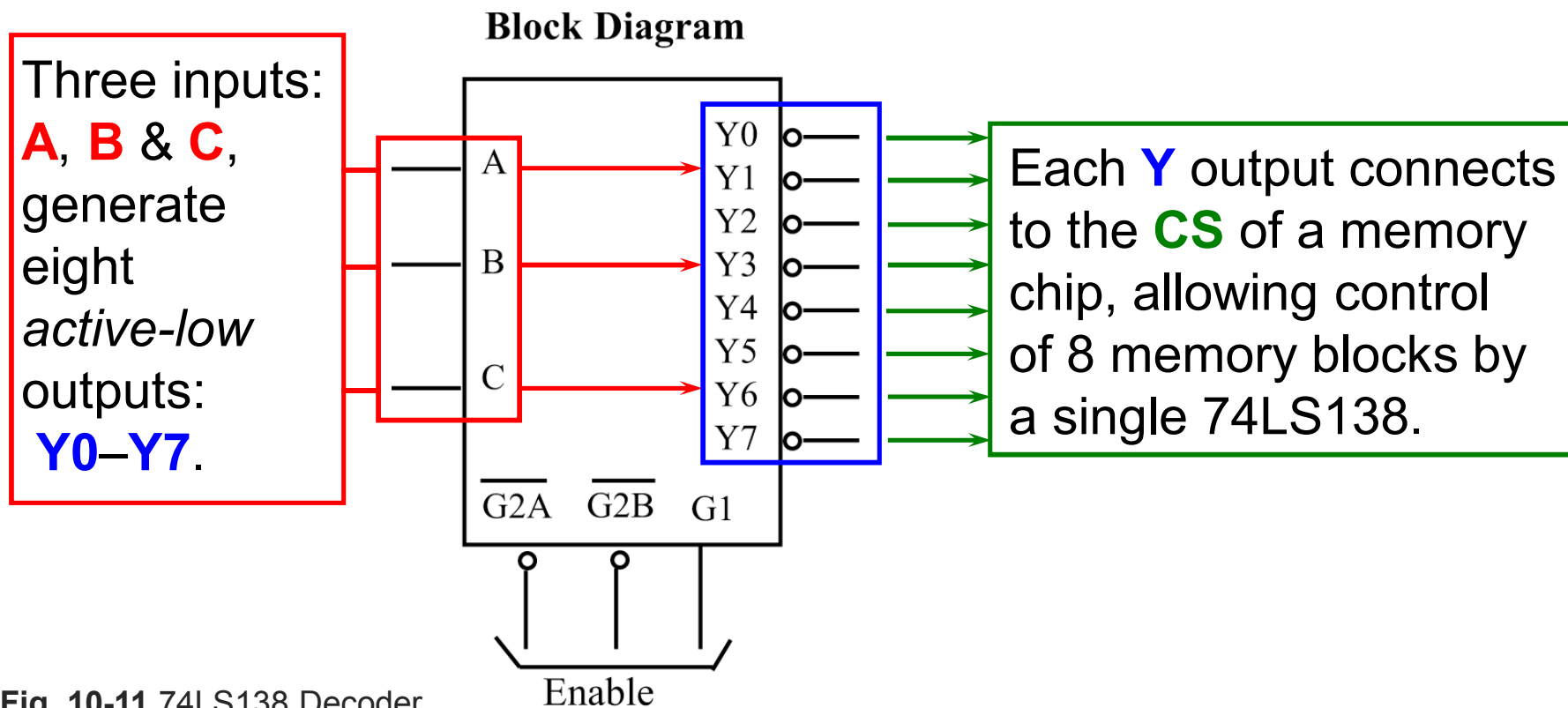


Fig. 10-11 74LS138 Decoder

## 10.2: MEMORY ADDRESS DECODING using the 74LS138 as decoder

- To enable 74SL138: **G2A** = 0, **G2B** = 0, **G1** = 1.
  - **G2A** & **G2B** are grounded; **G1** = 1 selects *this* 74LS138.

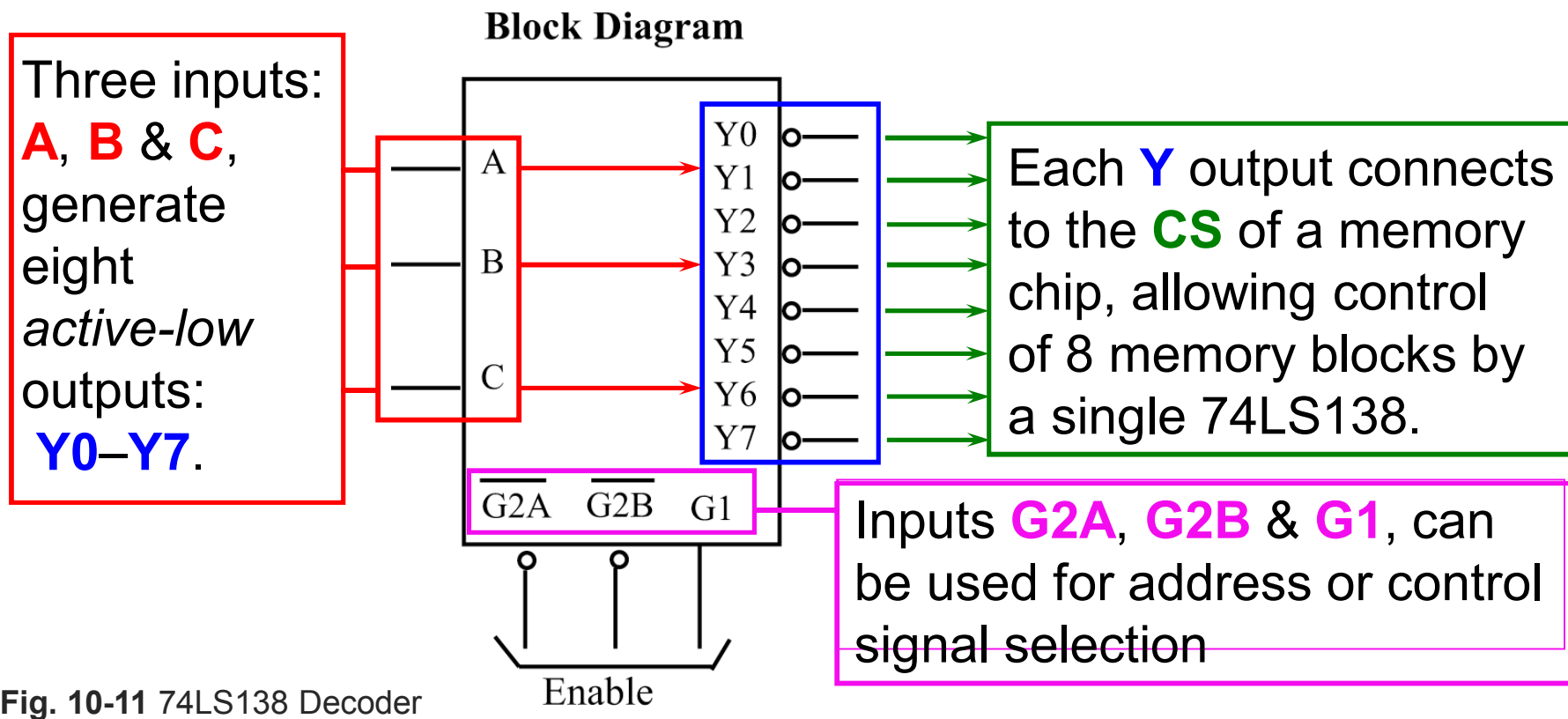
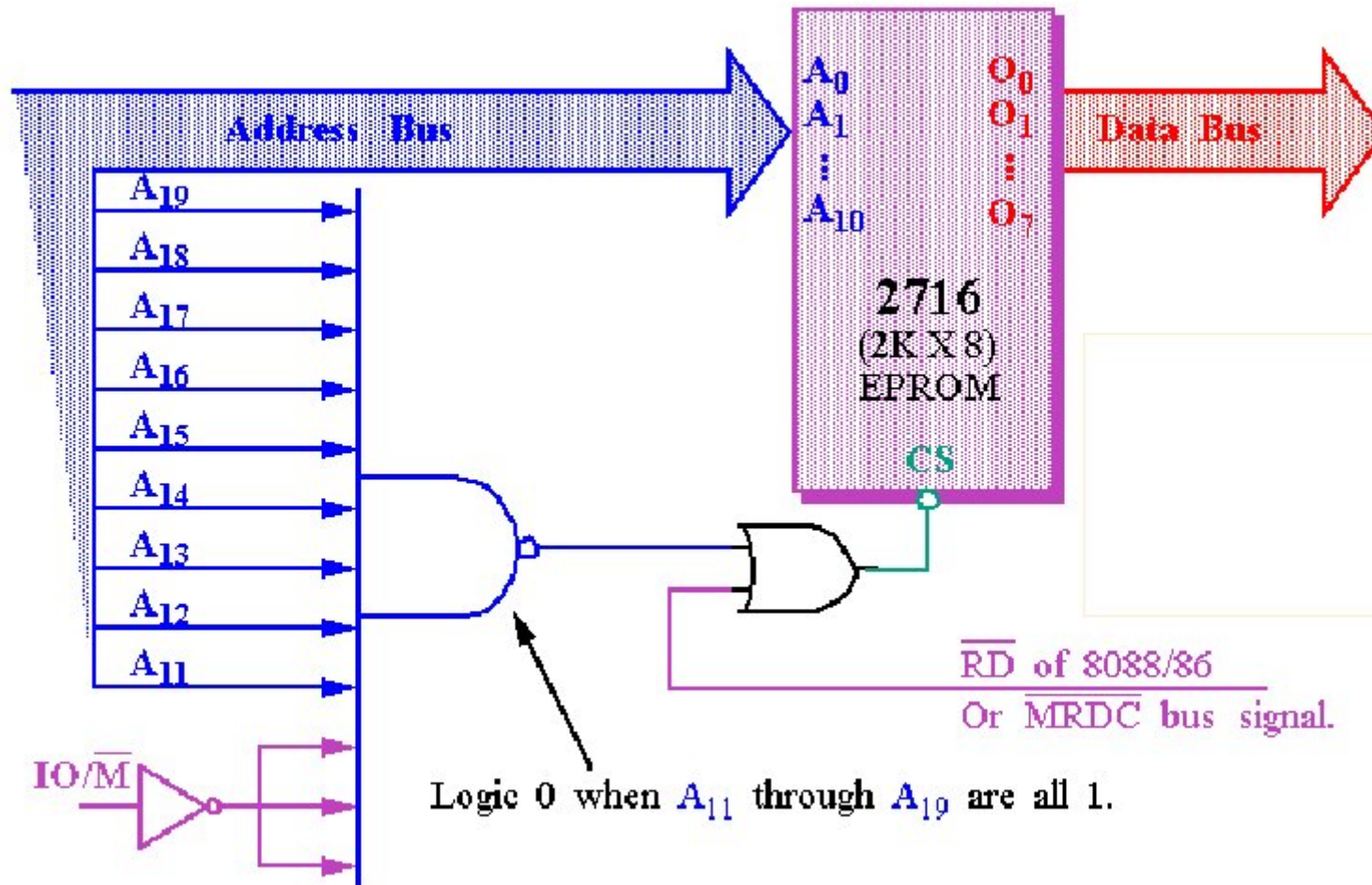
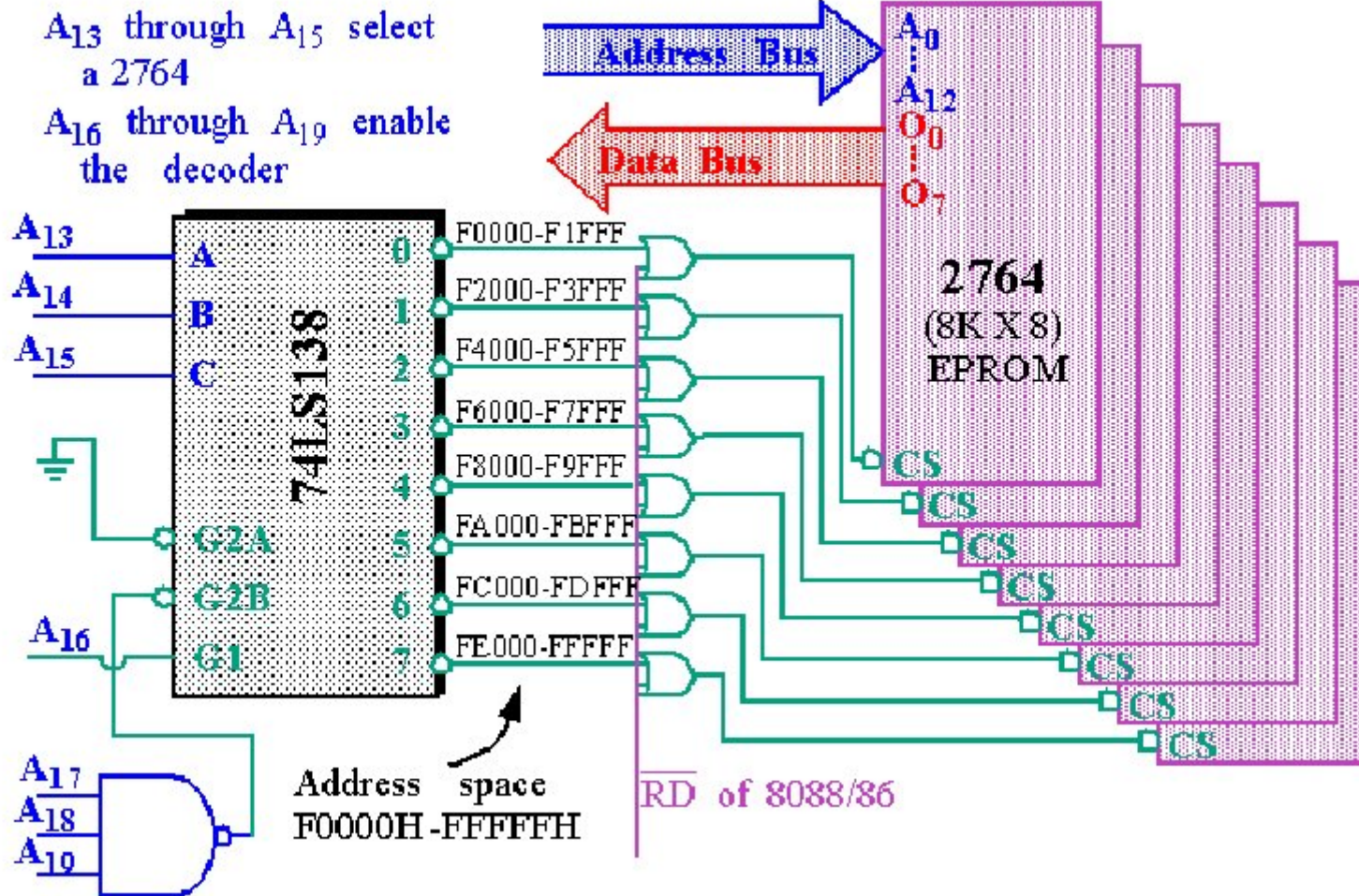


Fig. 10-11 74LS138 Decoder

# Memory Address Decoding

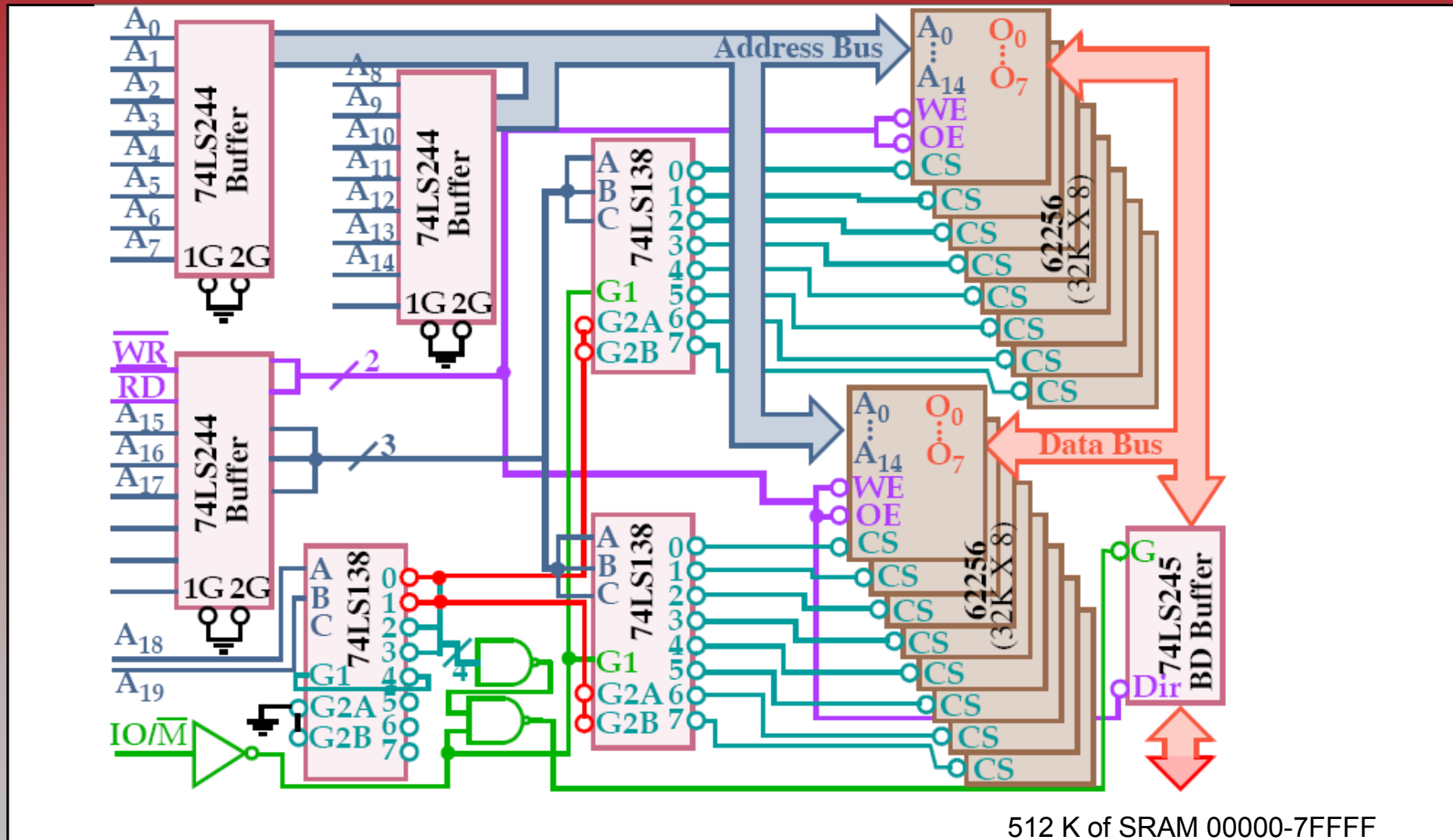


# Memory Address Decoding



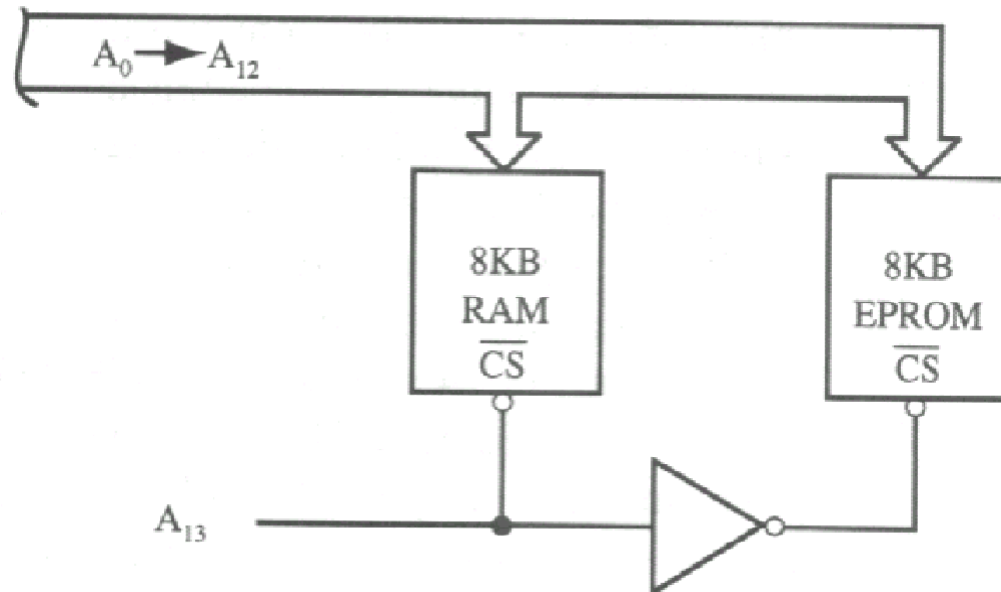


# Memory Addressing



# Partial Address Decoding

- Not all the address lines need to be used. (A14-A19 not used). So FFFF0, 3BFF0, 07FF0 or C3FF0 get the same data.
- (+) The purpose is get the job done in minimum hardware.
- (-) Feature expansion of the memory is impossible, and may cause invalid data reads due to overlapping memory segment reads (a fatal error)

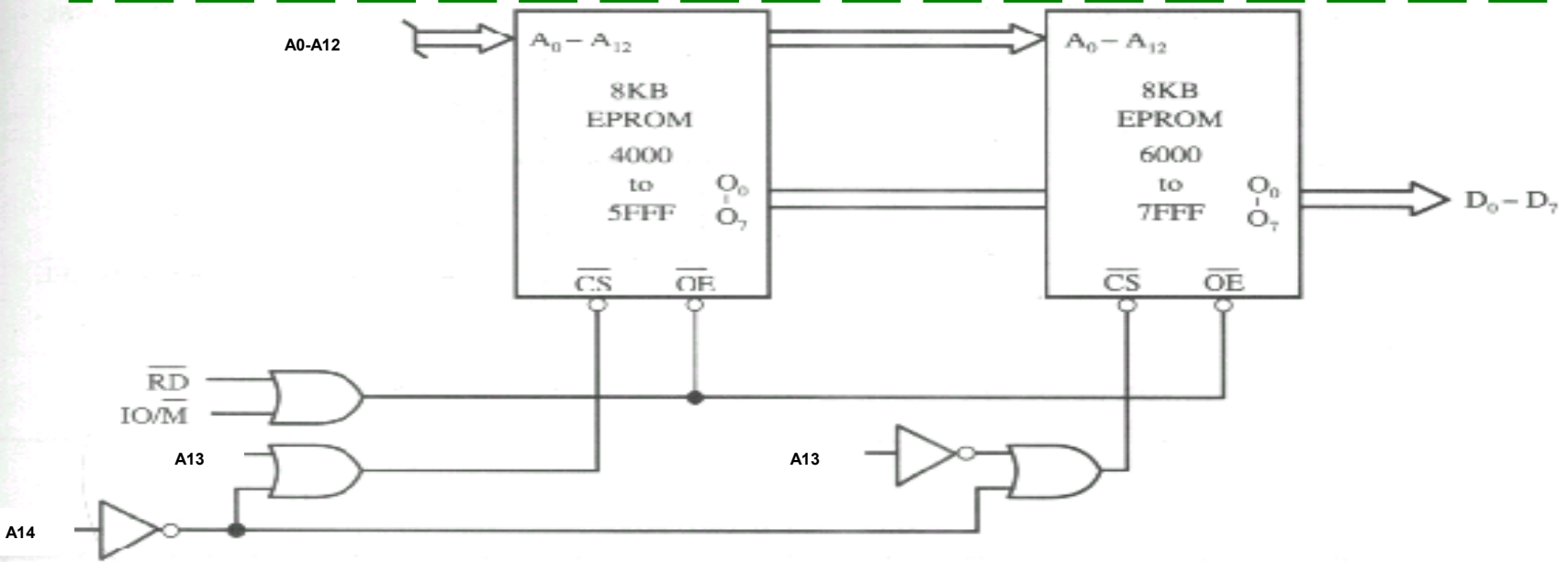


# Partial Address Decoding

| A <sub>19</sub> | A <sub>18</sub> | A <sub>17</sub> | A <sub>16</sub> | A <sub>15</sub> | A <sub>14</sub> | A <sub>13</sub> | A <sub>12</sub> | A <sub>11</sub> | A <sub>10</sub> | A <sub>9</sub> | A <sub>8</sub> | A <sub>7</sub> | A <sub>6</sub> | A <sub>5</sub> | A <sub>4</sub> | A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> | A <sub>0</sub> |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0               | 0               | 0               | 0               | 0               | 1               | 0               | 0               | 0               | 0               | 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              | 1              |
| 0               | 0               | 0               | 0               | 0               | 1               | 1               | 0               | 0               | 0               | 0              | 0              | 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              |

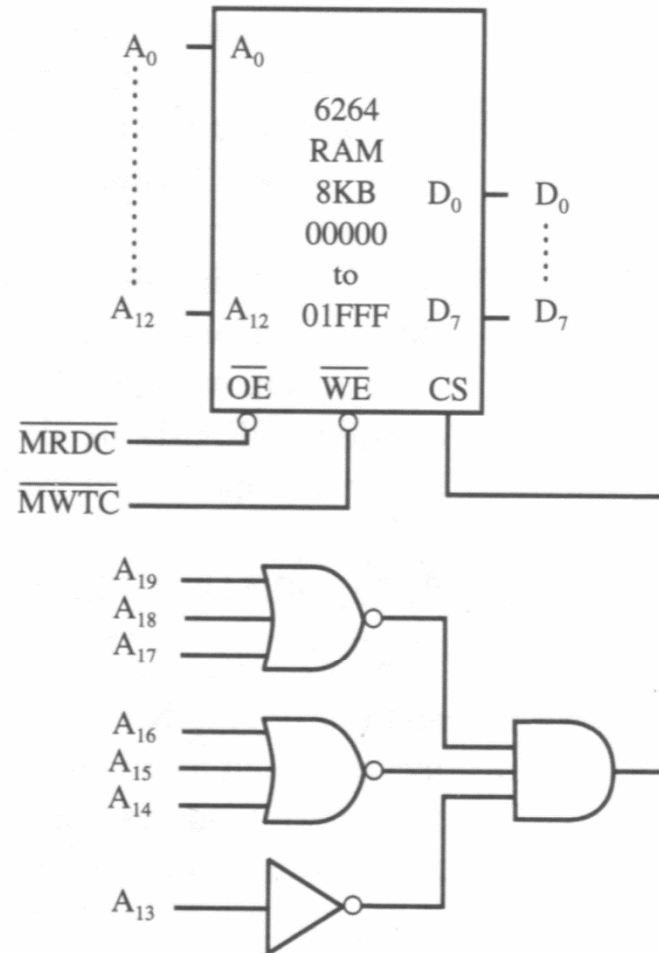
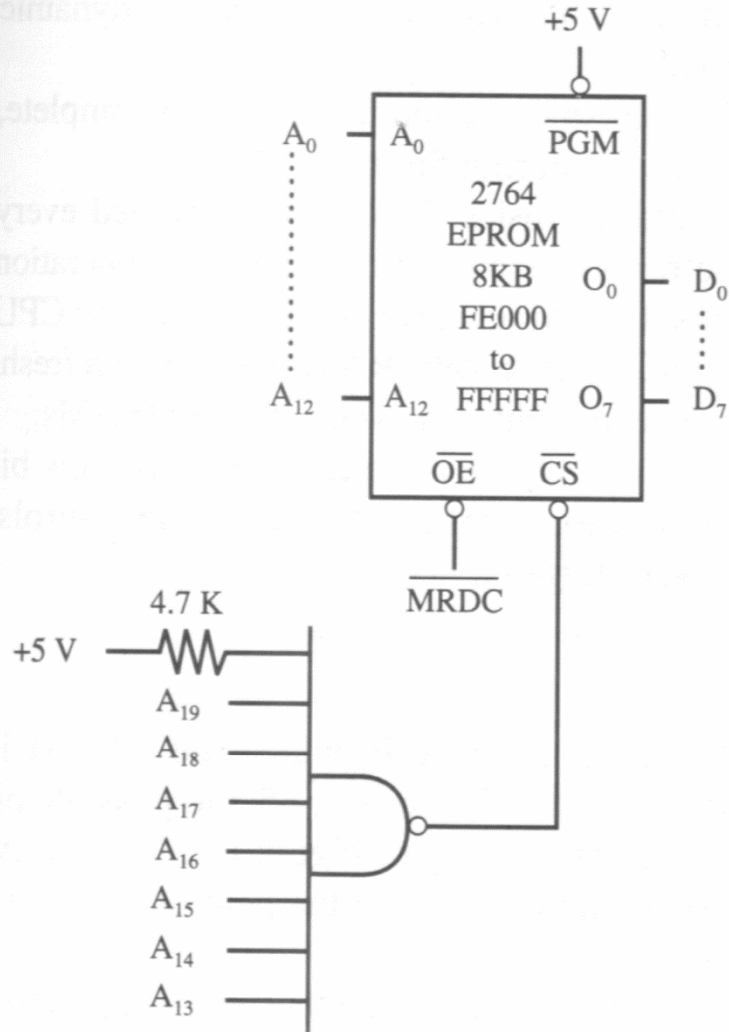
For DECODER (A<sub>19</sub>-A<sub>13</sub>)      To EPROM (A<sub>12</sub>-A<sub>0</sub>)

Foldback memory exists



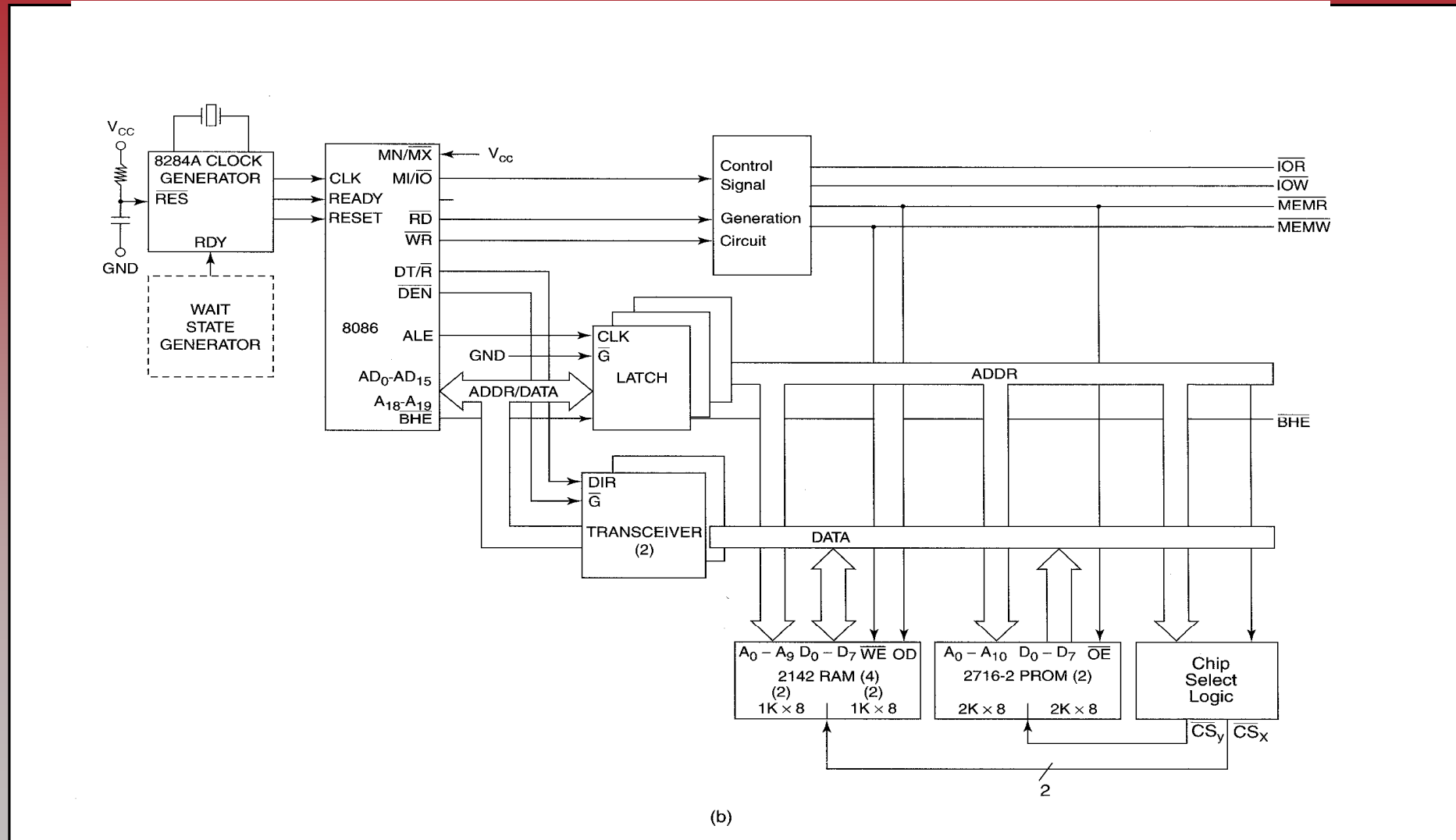
(b)

# A complete RAM/EPROM Memory

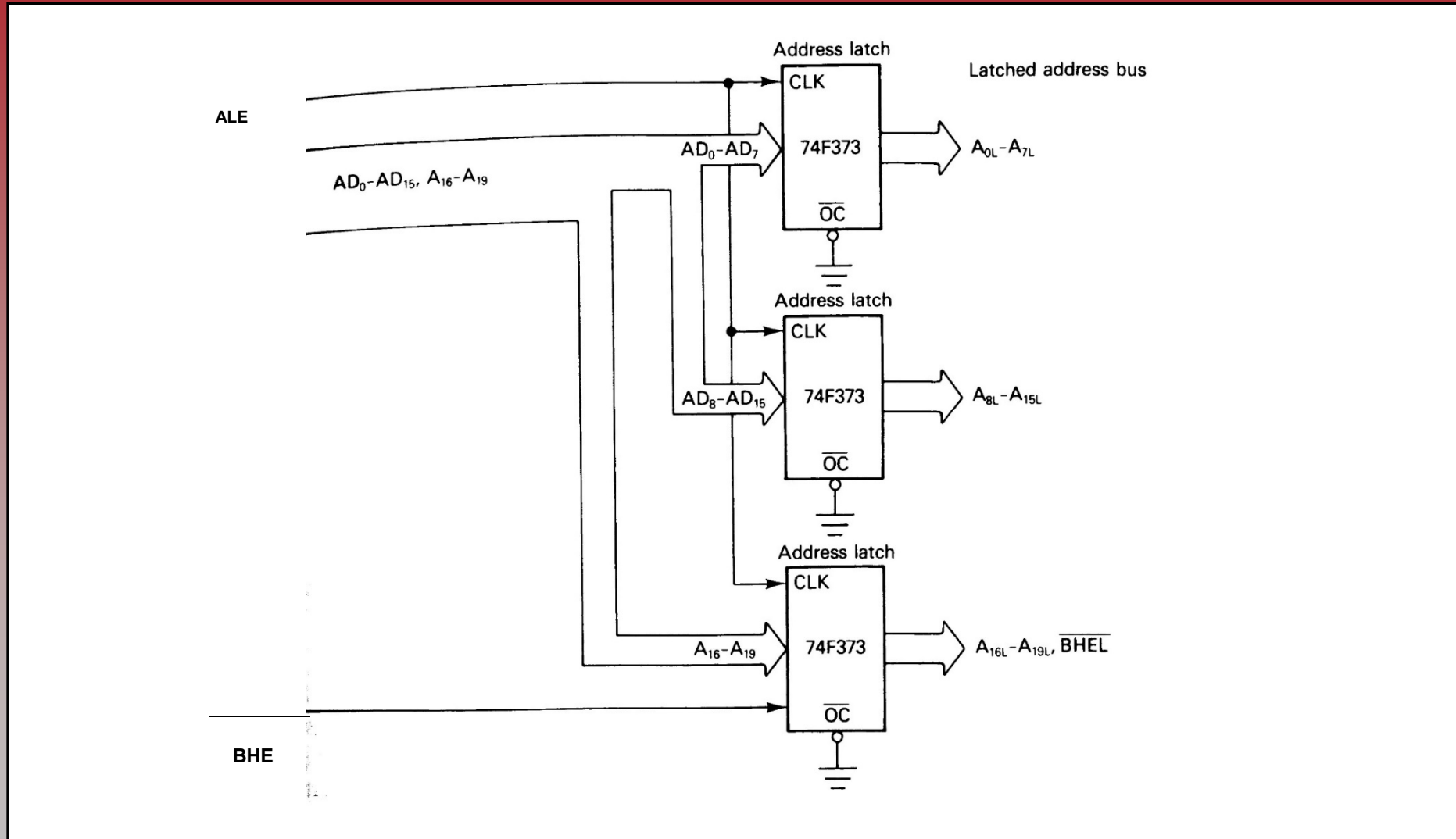


(b)

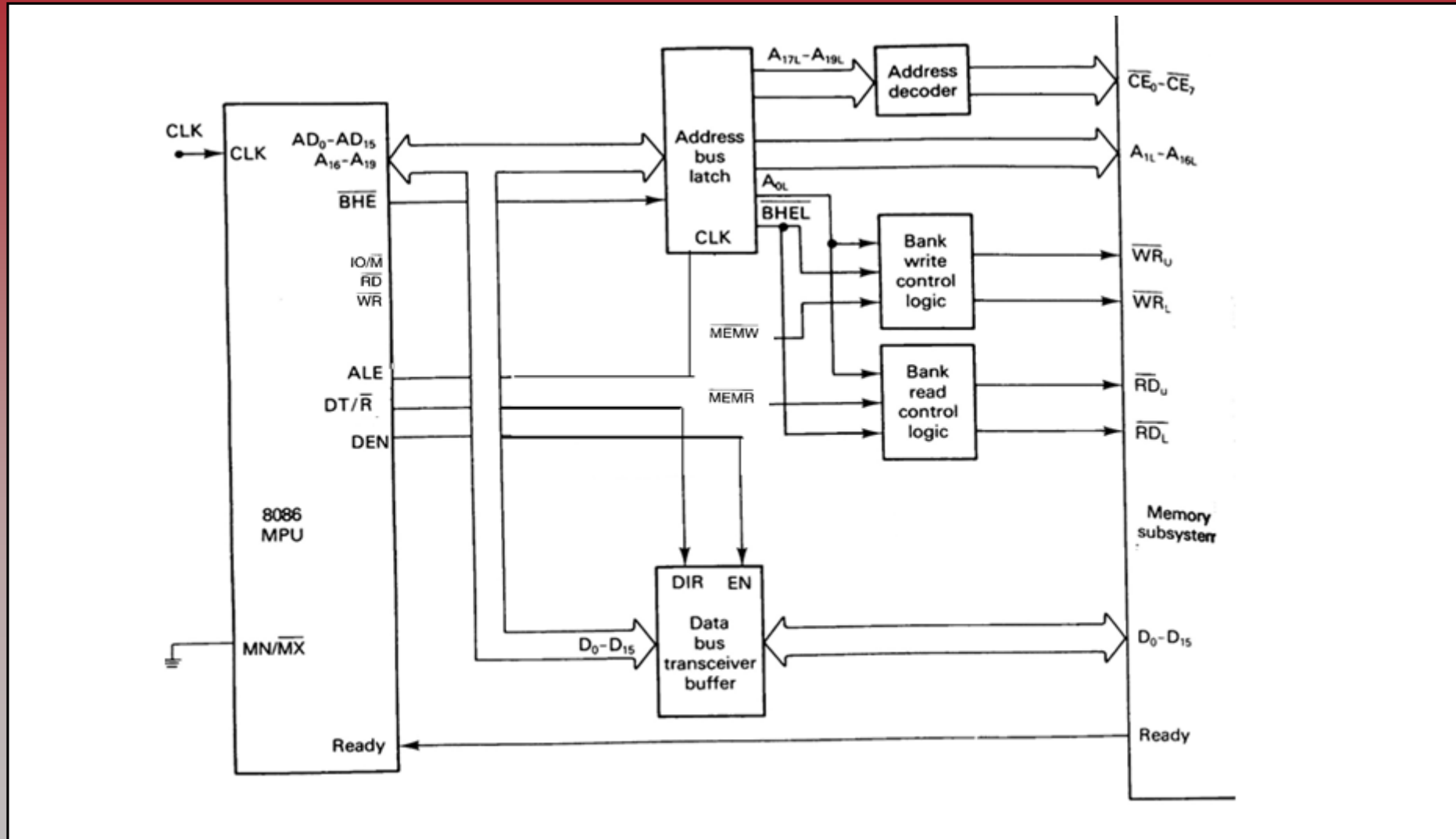
# Minmode 8086 Microcomputer system memory circuitry



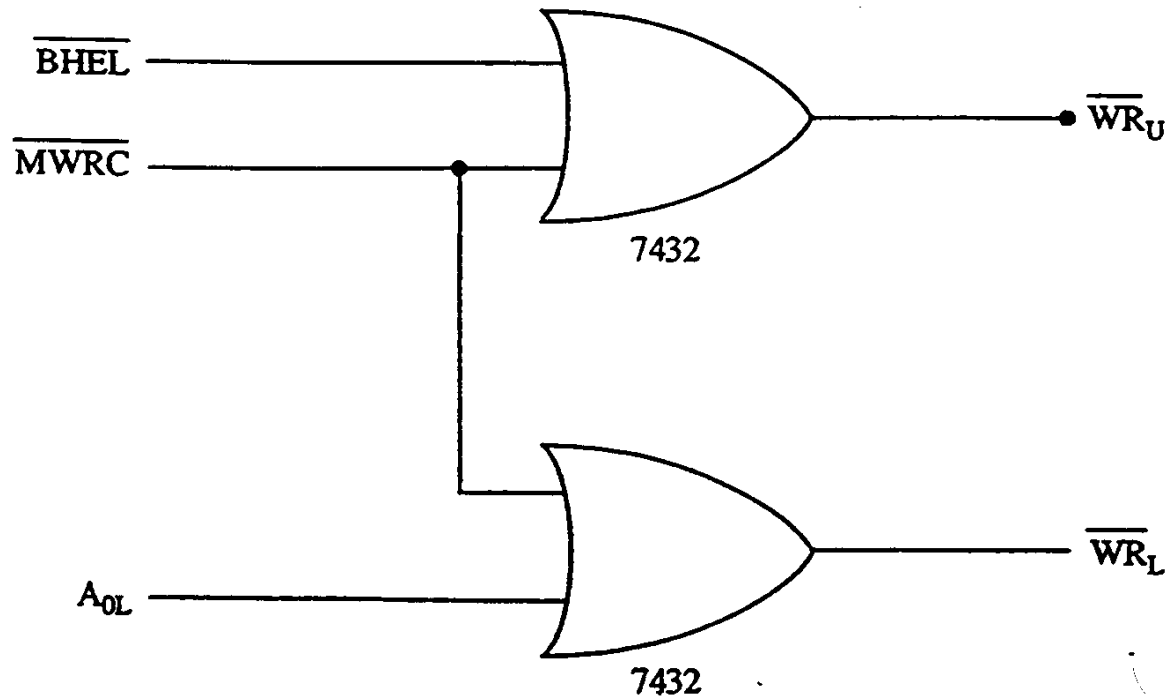
# Address Latch Circuit (8086)



# Memory Interface (8086)

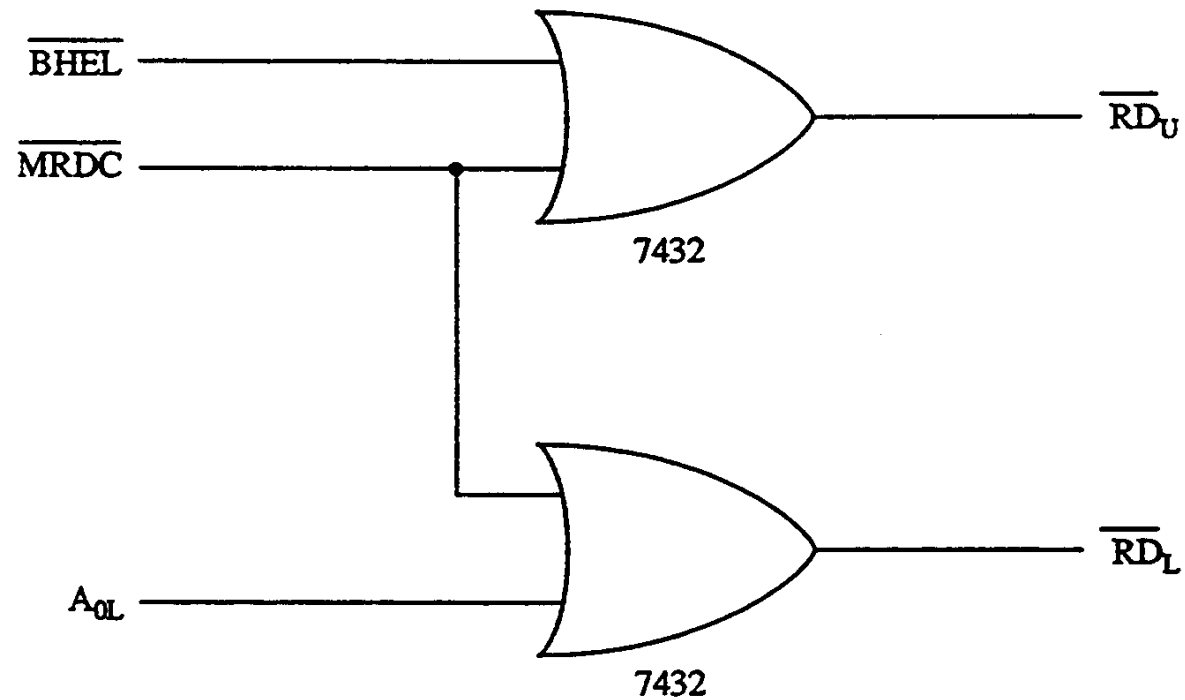


# Bank Write Control Logic (8086)

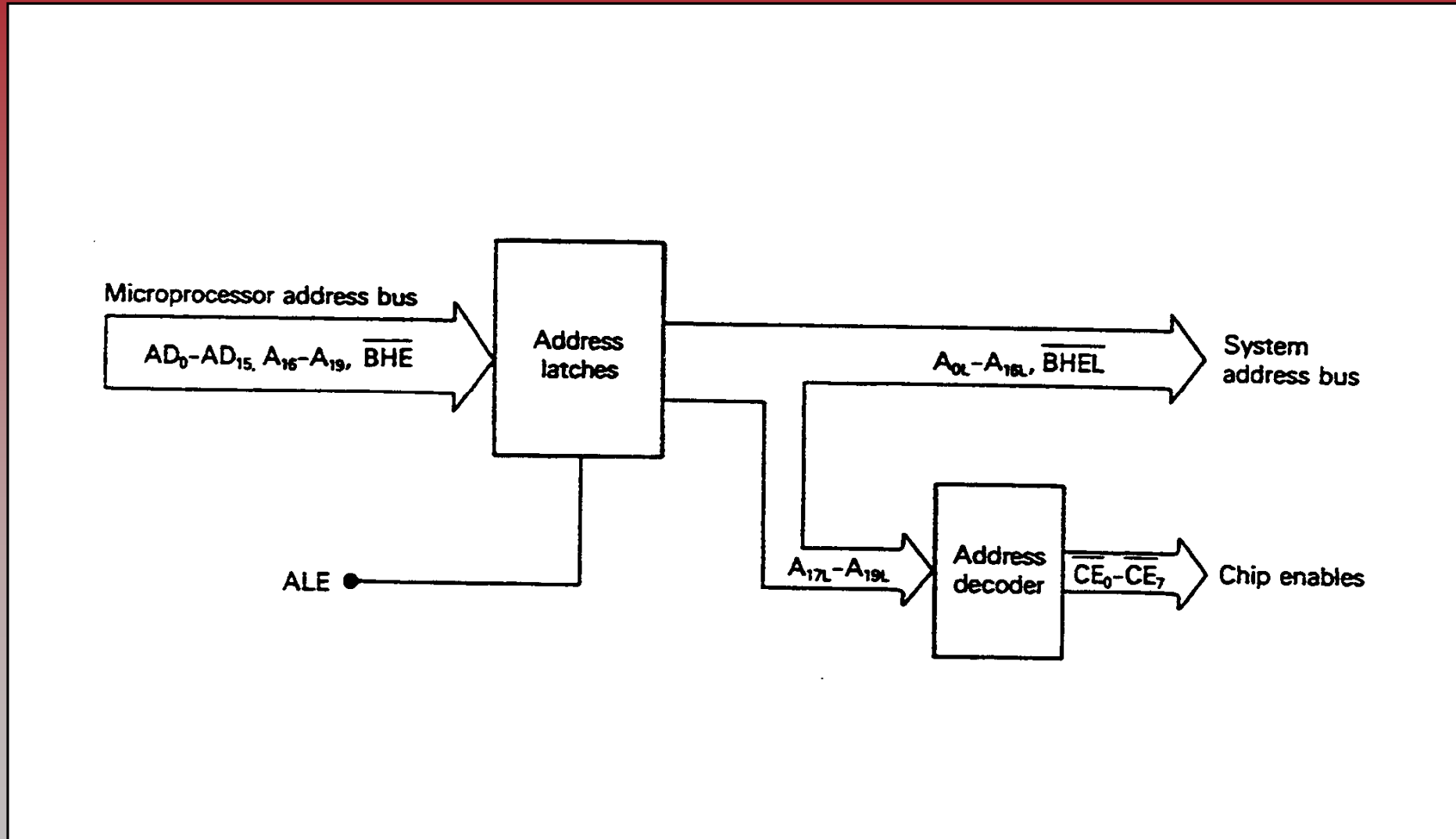




# Bank Read Control Logic (8086)



# Address Bus Configuration with Address Decoding (8086)



## 10.5: 16-BIT MEMORY INTERFACING

### bus bandwidth

- The main advantage of the 16-bit data bus is a doubling of the rate of transfer of information between the CPU and the outside world.
  - The rate of data transfer is called bus bandwidth.
    - The wider the data bus, the higher the bus bandwidth.
- Bus bandwidth is measured in MB (megabytes) per second and is calculated as follows:

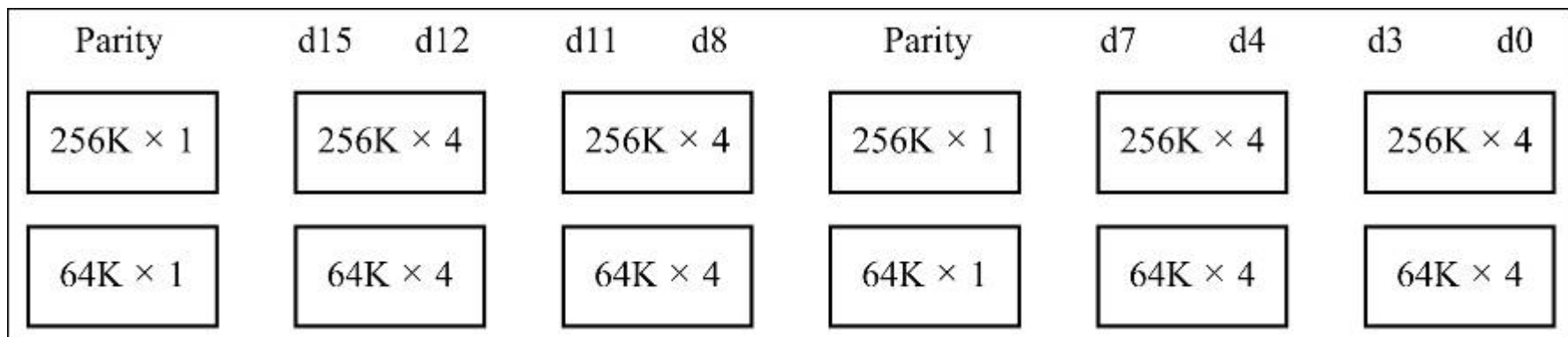
$$\text{bus bandwidth} = (1/\text{bus cycle time}) \times \text{bus width in bytes}$$

- There are two ways to increase the bus bandwidth:
  - Use a wider data bus, shorten bus cycle time, or do both.
    - 386, 486, and Pentium® processors have done this.

## 10.5: 16-BIT MEMORY INTERFACING

- In the design of current x86 PCs, details of CPU connection to memory and other peripherals are not visible for educational purposes.
  - 16-bit bus interfacing to memory chips is a detail now buried within a current PCs chipset.
    - Concepts from 80286 apply to any 16-bit microprocessor.

### 640 KB of DRAM for 16-bit buses.



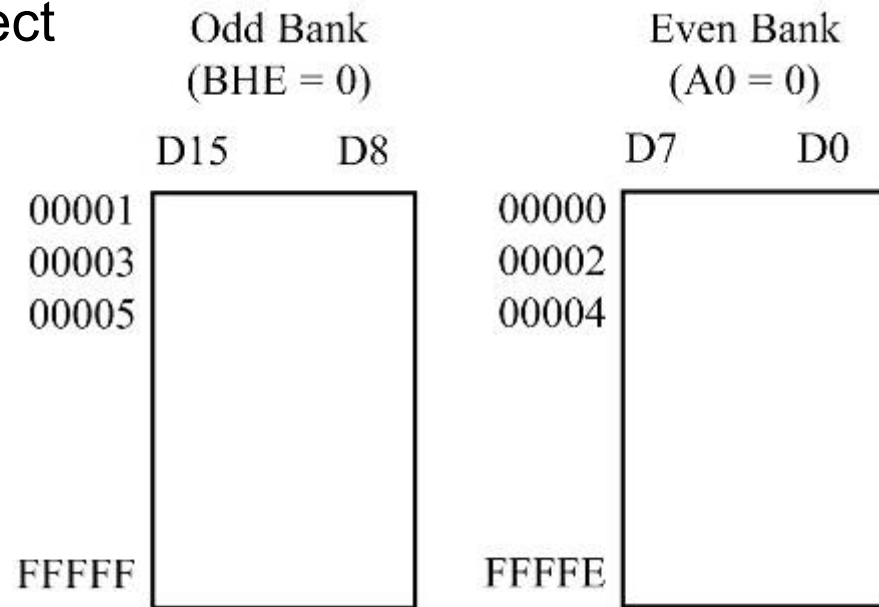
# 10.5: 16-BIT MEMORY INTERFACING

## ODD and EVEN banks

- In a 16-bit CPU, memory locations 00000–FFFFFF are designated as odd and even bytes.
  - To distinguish between odd & even bytes, the CPU provides a signal called **BHE** (bus high enable).
    - BHE, with A0 is used to select odd/even bytes.

**Table 10-7: Distinguishing Between Odd and Even Bytes**

| BHE | A0 |           |        |
|-----|----|-----------|--------|
| 0   | 0  | Even word | D0–D15 |
| 0   | 1  | Odd byte  | D8–D15 |
| 1   | 0  | Even byte | D0–D7  |
| 1   | 1  | None      |        |

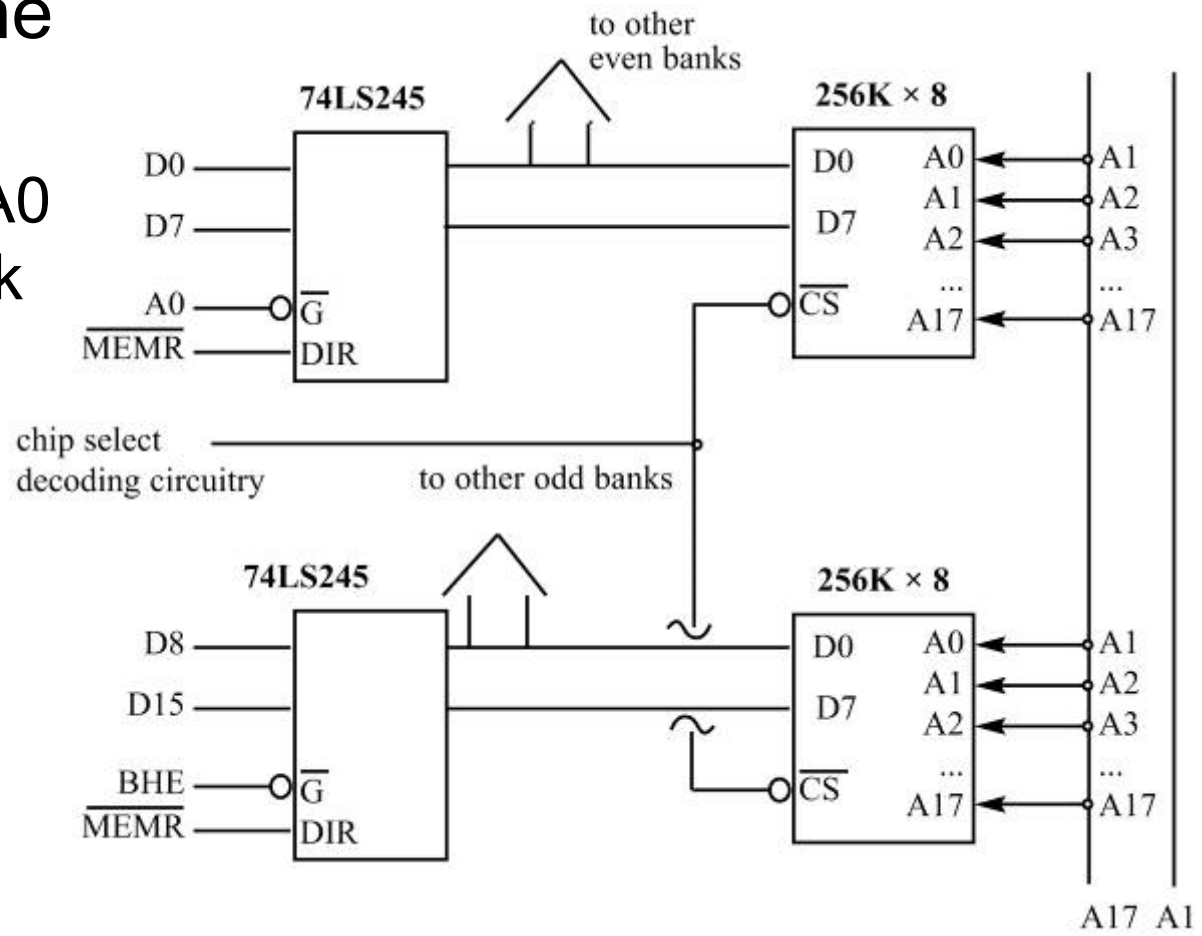


**Fig. 10-20** Odd & Even Banks of Memory

# 10.5: 16-BIT MEMORY INTERFACING

## ODD and EVEN banks

- Connection for the 16-bit data bus.
  - Note the use of A0 and BHE as bank selectors.
  - Also use of the 74LS245 chip as a data bus buffer.



**Fig. 10-22** 16-bit Data Connection in the Systems with 16-bit Data Bus

## 10.5: 16-BIT MEMORY INTERFACING

### memory cycle time & inserting wait states

- To access an external device such as memory or I/O, the CPU provides a fixed amount of time called a *bus cycle time*.
  - During this time, the read & write operation of memory or I/O must be completed.
- Bus cycle time used for accessing memory is often referred to as **MC** (memory cycle) time.
- The time from when the CPU provides addresses at its address pins, to when the data is expected at its data pins is called *memory read cycle time*.

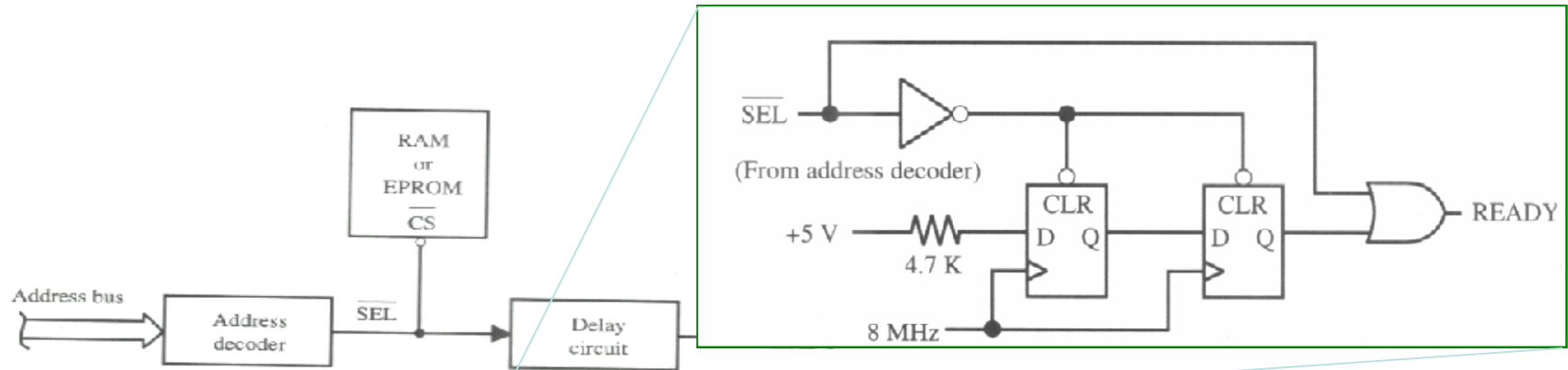
## 10.5: 16-BIT MEMORY INTERFACING

### memory cycle time & inserting wait states

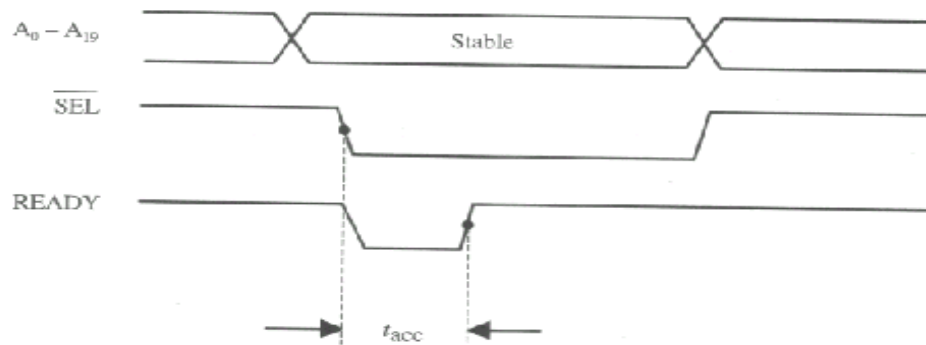
- If memory is slow and its access time does not match MC time of the CPU, extra time can be requested from the CPU to extend the read cycle.
  - Called a *wait state* (**WS**).
- To avoid too many wait states in interfacing memory to CPU, cache memory & other high-speed DRAMs were invented.
- Memory access time is not the only factor in slowing down the CPU, even though it is the largest one.
  - The other factor is the delay associated with signals going through the data and address path.



# Generating Wait States in Hardware



(a)



(b)

- ✓ Ready signal can be generated by special hardware using the SEL' signal from the address decoder.
- ✓ The circuit above will generate two clock periods of zero signal for the READY output.

## 10.5: 16-BIT MEMORY INTERFACING

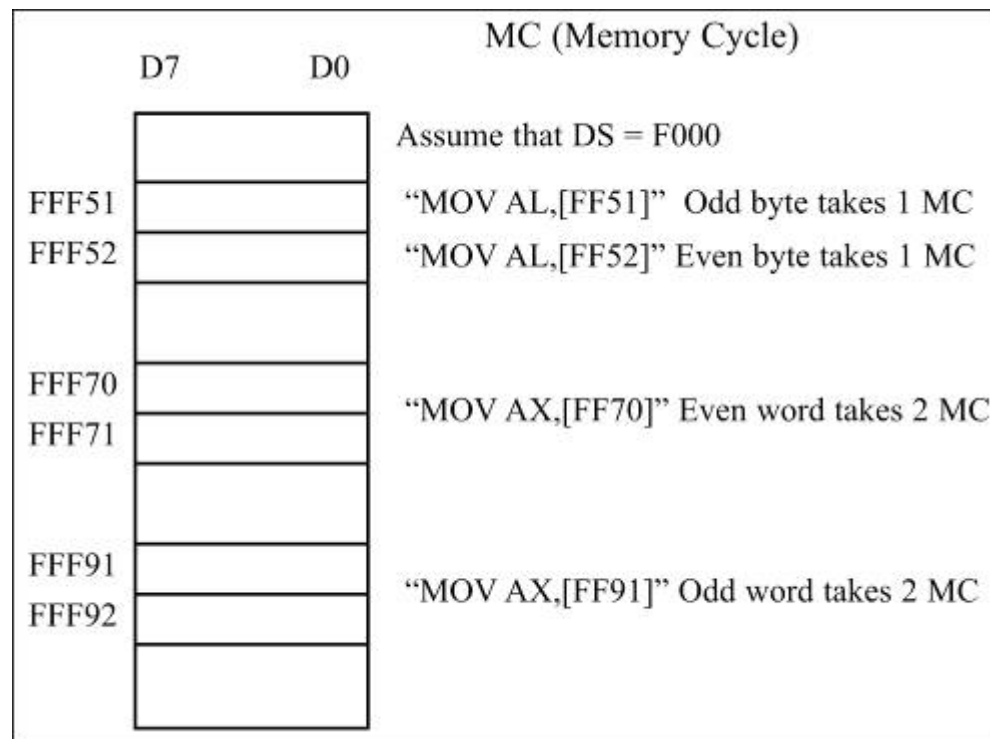
### accessing EVEN and ODD words

- Intel defines 16-bit data as a word, and the address of a word can start at an even or an odd number.
  - In systems with a 16-bit data bus, accessing a word from an odd addressed location can be slower.
    - In the instruction “**MOV AX, [2000]**” the address of the word being fetched into AX starts at an *even* address.
    - In the case of “**MOV AX, [2007]**” the address starts at an *odd* address.

## 10.5: 16-BIT MEMORY INTERFACING

### accessing EVEN and ODD words

- In the 8-bit system, accessing a word is treated like accessing two bytes.
  - Regardless of whether the address is odd or even.



Accessing a byte takes one memory cycle; accessing any word will take two cycles. In the 16-bit system, accessing a word with an even address takes one memory cycle.

## 10.3: IBM PC MEMORY MAP

- All x86 CPUs in real mode provide 20 address bits.
  - Maximum memory access is one megabyte.
- The 20 system address bus lines, A0–A19, can take the lowest value of all 0s to the highest value of all 1s in binary.
  - Converted to hex, an address range 00000H to FFFFFH.

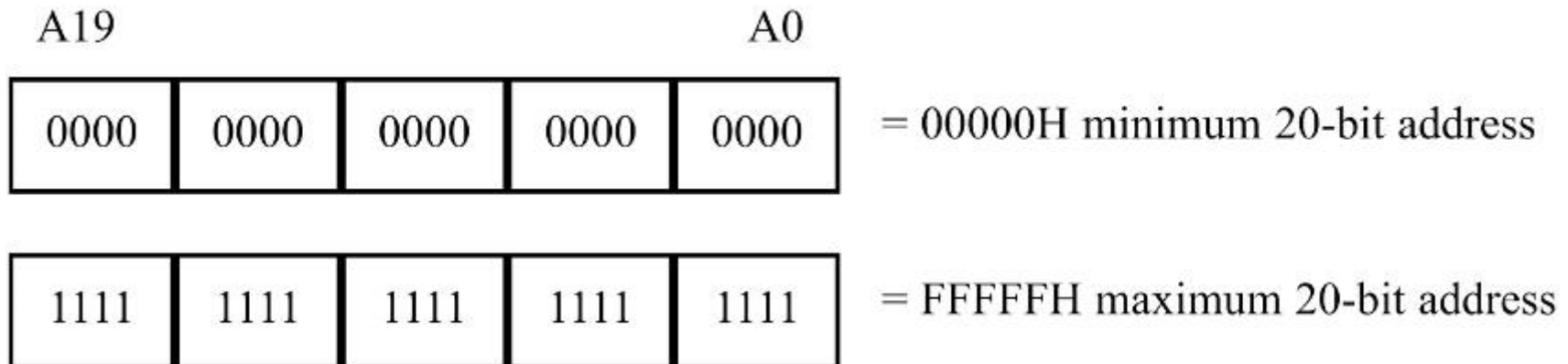


Fig. 10-14 20 Bit Address Range in Real Mode for x86 CPUs

# 10.3: IBM PC MEMORY MAP

## conventional memory – 640K of RAM

Any address assigned to any memory block in the 8088 PC must fall in this range

Of the addressable 1024K, PC designers set aside 640K for RAM, 128K for video display RAM, (VDR) & 256K for ROM.

In the x86 PC, addresses from 00000 to 9FFFFH, including location 9FFFFH, are set aside for RAM.

|       |          |
|-------|----------|
| FFFFF | ROM 256K |
| C0000 |          |
| BFFFF | VDR 128K |
| A0000 |          |
| 9FFFF | RAM 640K |
| 00000 |          |

**Fig. 10-15**  
Memory Map of the IBM PC

# 10.3: IBM PC MEMORY MAP

## BIOS data area

- The BIOS data area is used by BIOS to store some extremely important system information.

The operating system navigates the system hardware with the help of information stored in the BIOS data area.

**See the entire listing on page 271 of your textbook.**

Partial Listing of IBM PC RAM Memory Map for Interrupt, BIOS Data

| <u>Memory Location</u> | <u>Bytes</u> | <u>Description</u>             |
|------------------------|--------------|--------------------------------|
| 0000:0000 to 0000:03FF | 1024         | interrupt table                |
| 0000:0400 to 0000:0401 | 2            | port address of com1           |
| 0000:0402 to 0000:0403 | 2            | port address of com2           |
| 0000:0404 to 0000:0405 | 2            | port address of com3           |
| 0000:0406 to 0000:0407 | 2            | port address of com4           |
| 0000:0408 to 0000:0409 | 2            | port address of lpt1           |
| 0000:040A to 0000:040B | 2            | port address of lpt2           |
| 0000:040C to 0000:040D | 2            | port address of lpt3           |
| 0000:040E to 0000:040F | 2            | port address of lpt4           |
| 0000:0410 to 0000:0411 | 2            | list of installed hardware     |
| 0000:0412 to 0000:0412 | 1            | initialization flag            |
| 0000:0413 to 0000:0414 | 2            | memory size (K bytes)          |
| 0000:0415 to 0000:0416 | 2            | memory in I/O channel (if any) |
| 0000:0417 to 0000:0418 | 2            | keyboard status flag           |
| 0000:0419 to 0000:0419 | 1            | alternate key entry storage    |

## 10.3: IBM PC MEMORY MAP

### video display RAM (VDR) map

- To display information on the monitor of the PC, the CPU must first store it in video display RAM (VDR).
  - The video controller displays VDR contents on the screen.
    - Address of the VDR must be within the CPU address range.
- In the x86, from A0000 to BFFFFH, a total of 128K bytes of addressable memory is allocated for video.

**Table 10-4: Video Display RAM Memory Map**

| <b>Adapters</b> | <b>Number of Bytes Used</b> | <b>Starting Address</b> |
|-----------------|-----------------------------|-------------------------|
| CGA, EGA, VGA   | 16,384 (16 K)               | B8000H                  |
| MDA, EGA, VGA   | 4096 (4K)                   | B0000H                  |
| EGA, VGA        | 65,536 (64K)                | A0000H                  |

## 10.3: IBM PC MEMORY MAP

### ROM address and cold boot in the PC

- When power is applied to a CPU it must wake up at an address that belongs to ROM.
  - The first code executed by the CPU must be stored in nonvolatile memory.
  - On RESET, 8088 starts to fetch information from CS:IP of FFFF:0000.
    - Physical address FFFF0H.
  - As the microprocessor starts to fetch & execute instructions from FFFF0H, there must be an opcode in that ROM location.
    - The CPU finds the opcode for the FAR jump, and the target address of the JUMP.

**Table 10-5: 8088  
After RESET**

| CPU   | Contents |
|-------|----------|
| CS    | FFFFH    |
| DS    | 0000H    |
| SS    | 0000H    |
| ES    | 0000H    |
| IP    | 0000H    |
| Flags | Clear    |
| Queue | Empty    |



## 10.3: IBM PC MEMORY MAP

### ROM address and cold boot in the PC

- Example 10-9 shows a case using a simple DEBUG command.

#### Example 10-9

Using the DEBUG dump command, verify the JMP address for the cold boot and the BIOS date.

#### Solution:

From the directory containing DEBUG, enter the following:

```
C>DEBUG
-d ffff:0 LF
FFFF:0000 EA 5B E0 00 F0 30 34 2F-30 33 2F 30 37 00 FC .[...04/03/07..
-Q
C>
```

The first 5 bytes showed the jump command “EA” and the destination “F0000:E05B”. The next 8 bytes show the BIOS date, 04/03/07.

## 10.4: DATA INTEGRITY IN RAM AND ROM

- When storing or transferring data from one place to another, a major concern is data integrity.
- There are many ways to ensure data integrity depending on the type of storage.
  - The checksum method is used for ROM.
  - The parity bit method is used for DRAM.
  - The CRC (cyclic redundancy check) method is employed for mass storage devices such as hard disks and for Internet data transfer.

## 10.4: DATA INTEGRITY IN RAM AND ROM

### checksum byte

- To ensure the integrity of the contents of ROM, every PC must perform a checksum calculation.
  - Using a checksum byte, an extra byte tagged to the end of a series of bytes of data.
- To calculate the checksum byte of a series of bytes:
  - 1. Add the bytes together and drop the carries.
  - 2. The 2's complement of the total sum, the checksum byte, becomes the last byte of the stored information.
  - 3. Add all the bytes, including the checksum byte.
    - The result must be zero; If it is not zero, one or more bytes of data have been changed (corrupted).
    - See Examples 10-11 and 10-12.

## 10.4: DATA INTEGRITY IN RAM AND ROM

### checksum byte

#### Example 10-12

Assuming that the last byte of the following data is the checksum byte, show whether the data has been corrupted or not: 28H, C4H, BFH, 9EH, 87H, 65H, 83H, 50H, A7H, and 51H.

#### **Solution:**

The sum of the bytes plus the checksum byte must be zero; otherwise, the data is corrupted

$$28H + C4H + BFH + 9EH + 87H + 65H + 83H + 50H + A7H + 51H = 500H$$

By dropping the accumulated carries (the 5), we get 00. The data is not corrupted. See Figure 10-17 for a program that performs this verification.

***See also Example 11 on page 274 of your textbook.***

## 10.4: DATA INTEGRITY IN RAM AND ROM

### checksum program

- When the PC is turned on, one of the first things the BIOS does is to test the system ROM.
  - The code for such a test is stored in the BIOS ROM.

```

                2411 ;-----
                2412 ;       ROS CHECKSUM SUBROUTINE           :
                2413 ;-----
EC4C            2414 ROS_CHECKSUM PROC NEAR    ;NEXT_ROS_MODULE
EC4C B90020     2415             MOV    CX,8192 ;NUMBER OF BYTES TO ADD
EC4F           2416 ROS_CHECKSUM_CNT:  ;ENTRY PT. FOR OPTIONAL ROS TEST
EC4F 32C0      2417             XOR    AL,AL
EC51           2418 C26:
EC51 0207     2419             ADD    AL,DS:[ BX]
EC53 43       2420             INC    BX    ;POINT TO NEXT BYTE
EC54 E2FB     2421             LOOP  C26  ;ADD ALL BYTES IN ROS MODULE
EC56 0AC0     2422             OR     AL,AL ; SUM = 0?
EC58 C3       2423             RET
                2424 ROS_CHECKSUM ENDP
```

## 10.4: DATA INTEGRITY IN RAM AND ROM checksum program

- Note in the code how all the bytes are added together without keeping the track of carries.
  - The total sum is ORed with itself to see if it is zero.
  - The zero flag is expected to be high on return from this subroutine; If it is not, the ROM is corrupted.

```
                2411 ;-----  
                2412 ;       ROS CHECKSUM SUBROUTINE       :  
                2413 ;-----  
EC4C             2414 ROS_CHECKSUM PROC NEAR ;NEXT_ROS_MODULE  
EC4C B90020      2415             MOV  CX,8192 ;NUMBER OF BYTES TO ADD  
EC4F             2416 ROS_CHECKSUM_CNT: ;ENTRY PT. FOR OPTIONAL ROS TEST  
EC4F 32C0        2417             XOR   AL,AL  
EC51             2418 C26:  
EC51 0207        2419             ADD   AL,DS:[ BX]  
EC53 43          2420             INC   BX   ;POINT TO NEXT BYTE  
EC54 E2FB        2421             LOOP  C26 ;ADD ALL BYTES IN ROS MODULE  
EC56 0AC0        2422             OR    AL,AL ; SUM = 0?  
EC58 C3          2423             RET  
                2424 ROS_CHECKSUM ENDP
```

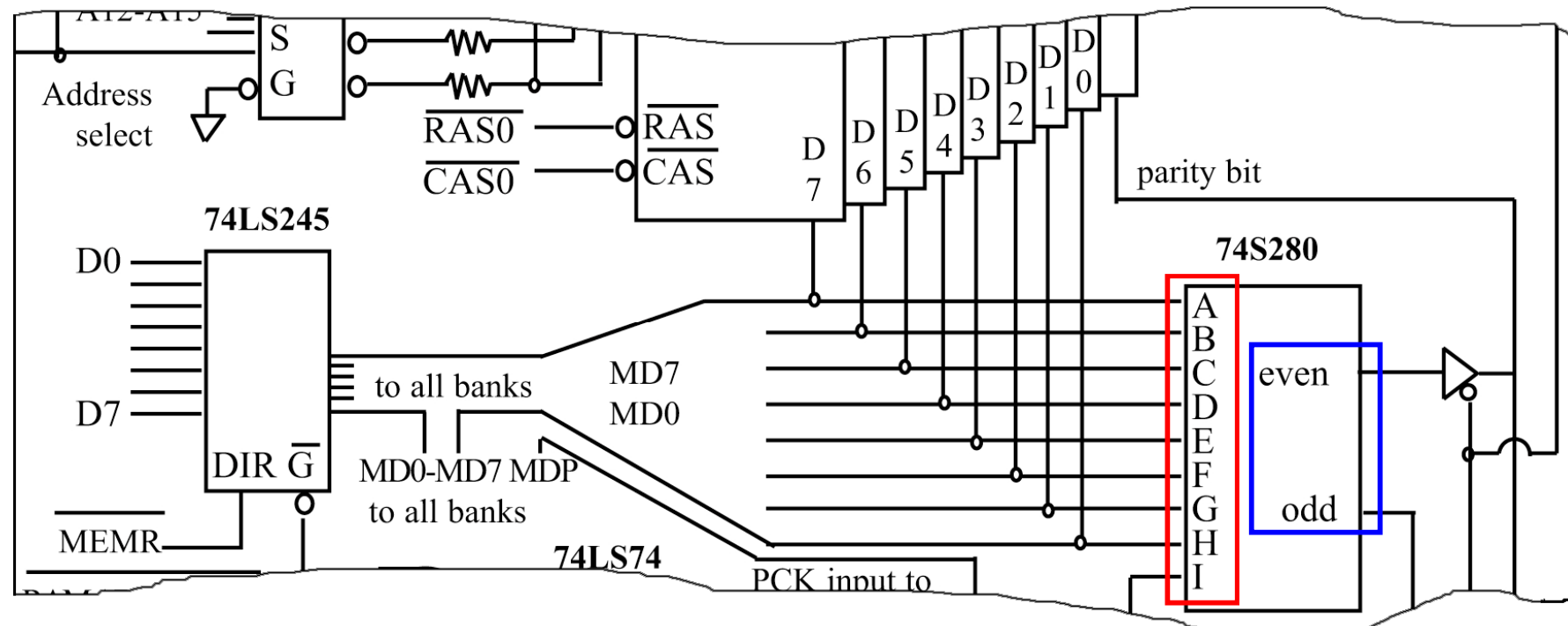
## 10.4: DATA INTEGRITY IN RAM AND ROM parity bit generator/checker in the PC

- Parity is used to detect two types of DRAM errors:
  - **Hard error** - some bits, or an entire row of memory cells in the memory chip get permanently stuck high or low.
    - Thereafter *always* producing 1 or 0, regardless of what is written into the cell(s).
  - **Soft error** - a single bit is changed from 1 to 0 or 0 to 1.
    - Due to current surge or certain particle radiation in the air.
- Including a parity bit to ensure RAM data integrity is the most widely used, simplest & cheapest method.
  - This method can only indicate if there is a difference between the data written to memory, and data read.
    - It cannot correct the error, as some high-performance systems.

## 10.4: DATA INTEGRITY IN RAM AND ROM

### 74S280 parity bit generator and checker

- To understand parity bit circuitry, it is necessary to understand the 74LS280 parity bit generator & checker chip, which has 9 **inputs** & 2 **outputs**.



**See the entire diagram on page 276 of your textbook.**



## 10.4: DATA INTEGRITY IN RAM AND ROM

### 74S280 parity bit generator and checker

- Even or odd output is activated as in Table 10-6.
  - If all 9 inputs have an *even* number of 1 bits, the **even** output goes *high*, as in cases 1 and 4.
  - If the 9 inputs have an *odd* number of high bits, the **odd** output goes *high*, as in cases 2 and 3.

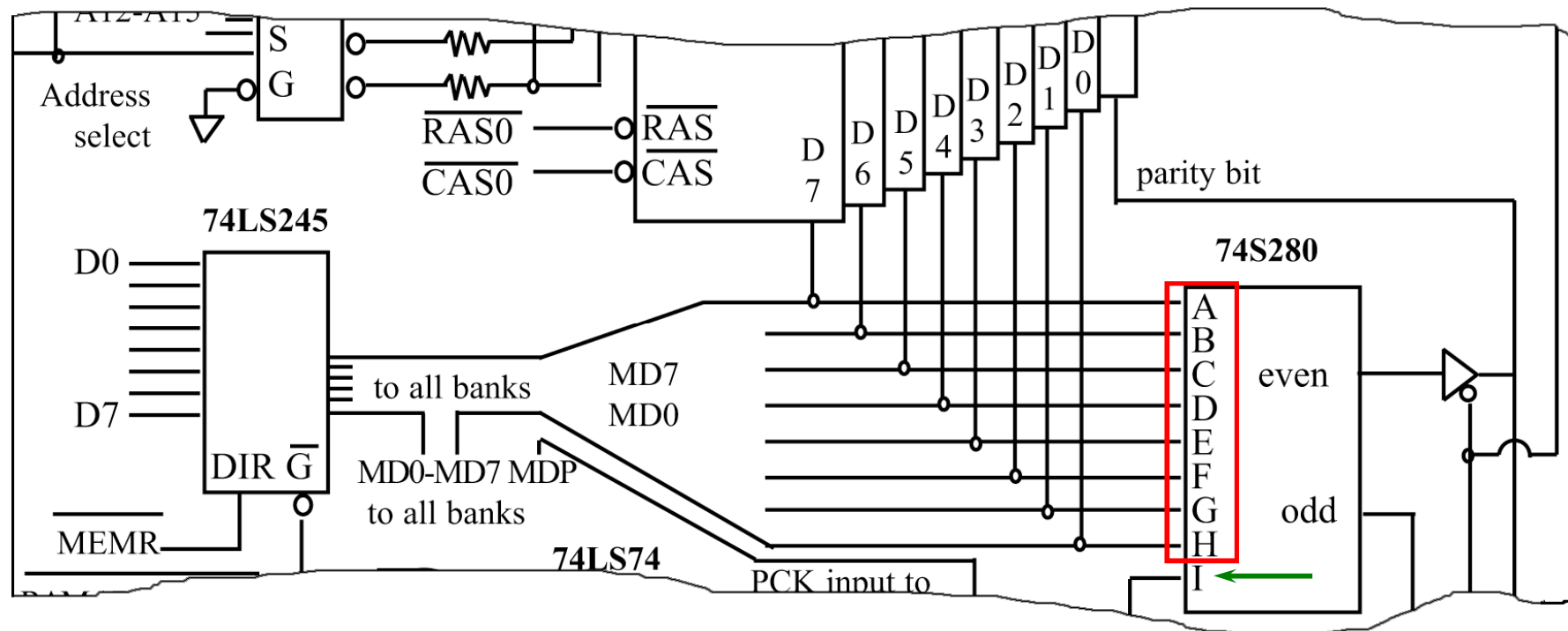
**Table 10-6: 74280 Parity Check**

| Case | Inputs |   | Outputs |     |
|------|--------|---|---------|-----|
|      | A–H    | I | Even    | Odd |
| 1    | Even   | 0 | 1       | 0   |
| 2    | Even   | 1 | 0       | 1   |
| 3    | Odd    | 0 | 0       | 1   |
| 4    | Odd    | 1 | 1       | 0   |

# 10.4: DATA INTEGRITY IN RAM AND ROM

## 74S280 parity bit generator and checker

- Inputs **A–H** are connected to the data bus, 8 bits.
  - The **I** input is used as a parity bit to check the data byte read from memory.



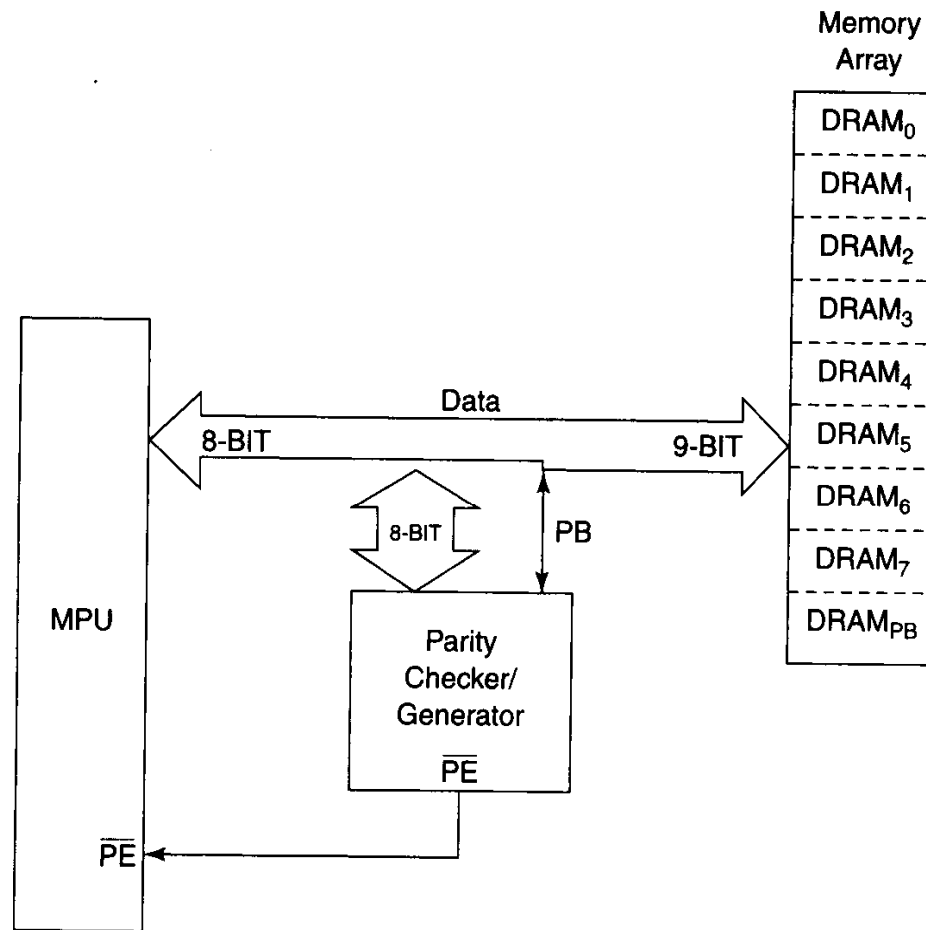
**See the entire diagram on page 276 of your textbook.**

## 10.4: DATA INTEGRITY IN RAM AND ROM

### 74S280 parity bit generator and checker

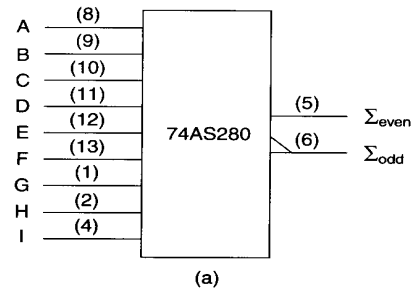
- When a byte is written to a memory location, the even-parity bit is generated and saved on the ninth DRAM chip as a parity bit.
  - With use of control signal  $\overline{\text{MEMW}}$ .
- When a byte of data is read from the same location:
  - The parity bit is gated into the I input through  $\overline{\text{MEMR}}$ .
- If there is a difference between data written & read, the **Q** output (**PCK**, parity bit check) of the 74LS74 is activated.
  - **Q** activates NMI, indicating a parity bit error, and will display a parity bit error message.

# Parity circuits



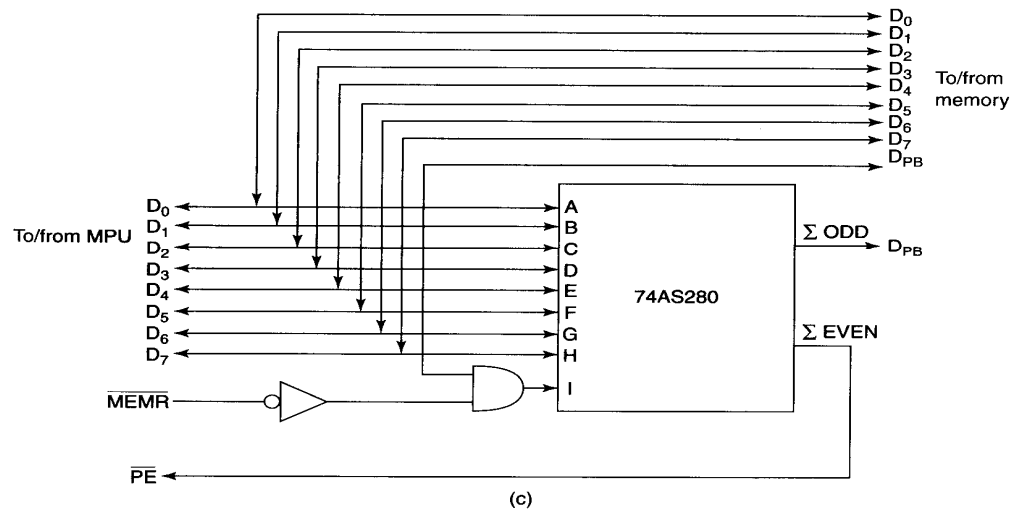
**Figure 9-23** Data-storage memory interface with parity-checker generator.

# Parity circuits

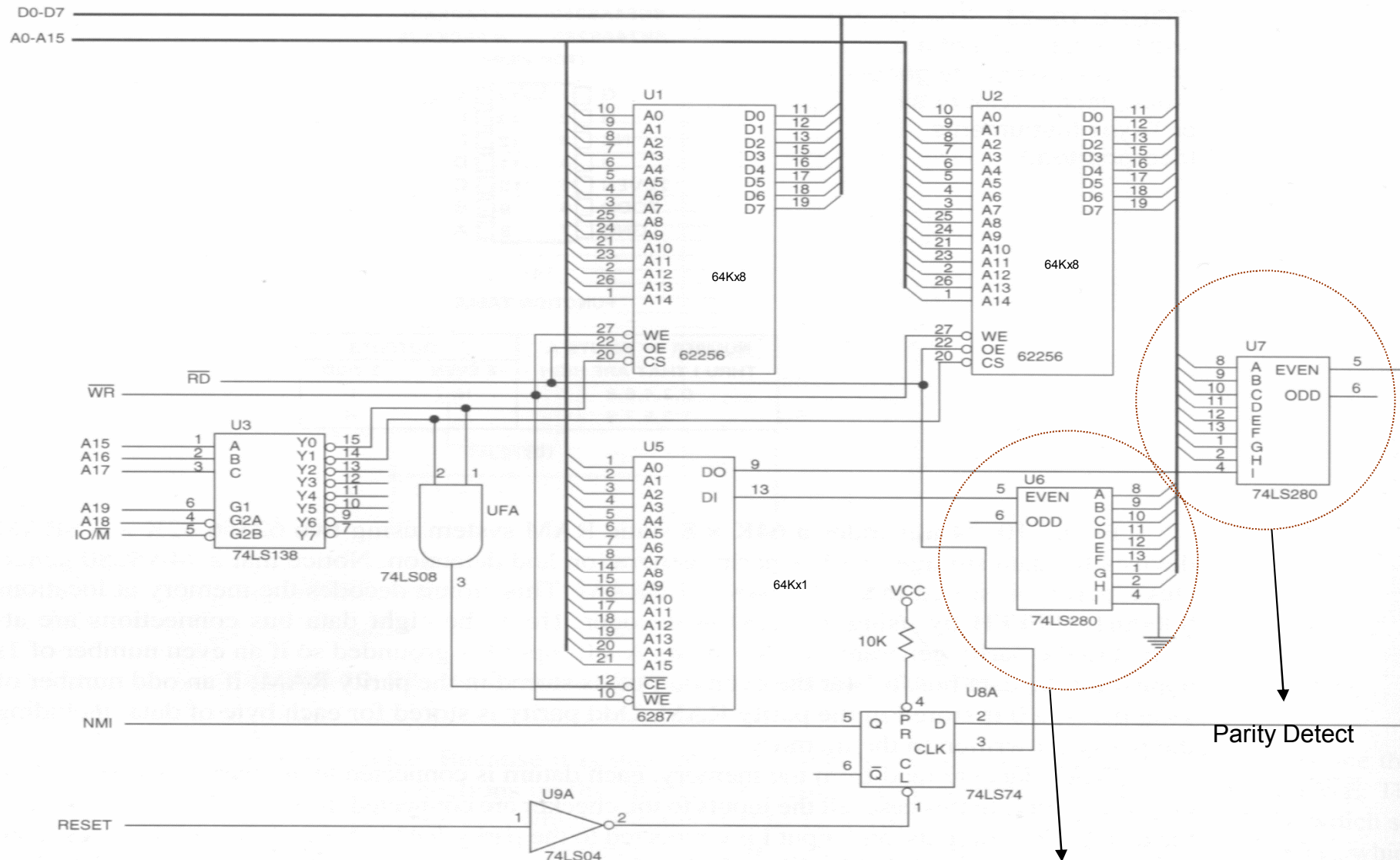


**Figure 9-24** (a) Block diagram of the 74AS280. (Texas Instruments Incorporated) (b) Function table. (Texas Instruments Incorporated) (c) Even-parity checker/generator connection.

| NUMBER OF INPUTS A THRU I THAT ARE HIGH | OUTPUTS |       |
|---|---------|-------|
|   | Σ EVEN  | Σ ODD |
| 0,2,4,6,8                               | H       | L     |
| 1,3,5,7,9                               | L       | H     |



# Parity Error Detection Circuit




# Checksum byte (used for ROM)

- ✓ Add the bytes together and drop the carries
- ✓ Take the 2's complement of the total sum, and that is the checksum byte, which becomes the last byte of the stored information.
- ✓ To perform the checksum operation add all the bytes, including the checksum byte. The result must be zero. If it is not zero, one or more bytes of data have been changed (corrupted)

Example: Assume that we have 5 bytes of hexadecimal data: 1A, 14, 82, FC, 3E.

- a) Find the checksum byte
- b) Perform the checksum operation to ensure integrity
- c) If the 3<sup>rd</sup> byte is changed to 44 show how the error is detected?

- a) The checksum is:  $1A+14+82+FC+3E = 1EA$  drop 1  $\rightarrow EA$ , take 2's comp  $\Rightarrow 16$
- b)  $1A+14+82+FC+3E+16 = 00$
- c)  $1A+14+44+FC+3E+16 = 1C2 \rightarrow$  Error!



| Dec | Hex | Bin      |
|-----|-----|----------|
| 10  | A   | 00001010 |

**ENDS ; Week8**

# The x86 PC

assembly language, design, and interfacing

fifth  
edition

Prentice Hall

# The x86 PC

assembly language,  
design, and interfacing

fifth edition

**MUHAMMAD ALI MAZIDI**  
**JANICE GILLISPIE MAZIDI**  
**DANNY CAUSEY**