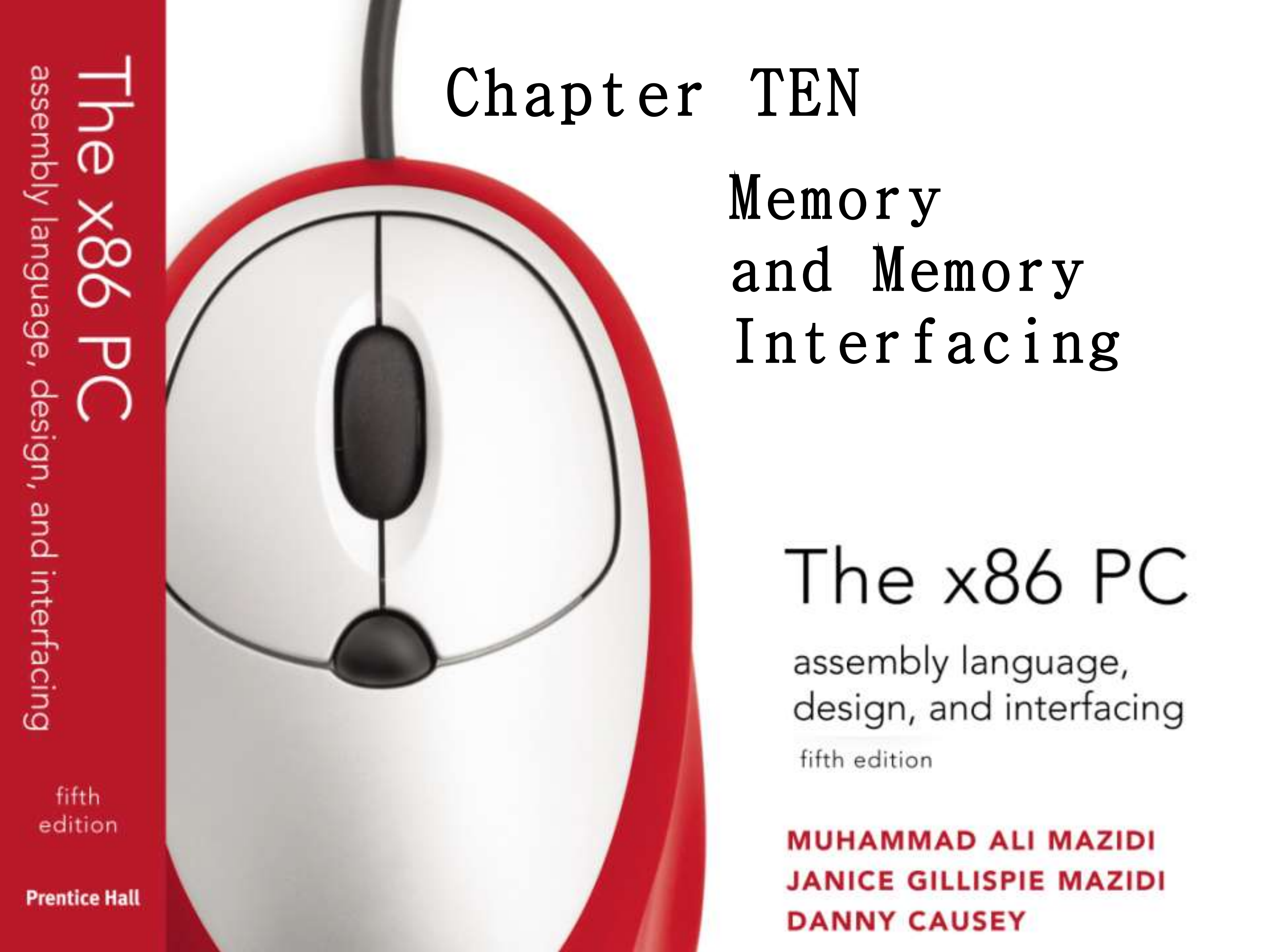
Chapter TEN

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

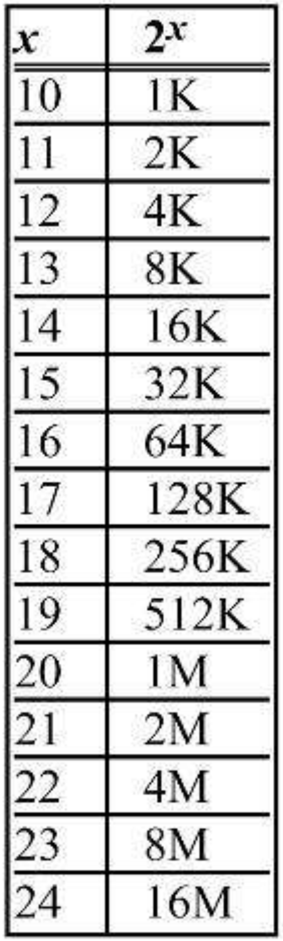
OBJECTIVES
this chapter enables the student to:

- 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.

10.1: SEMICONDUCTOR MEMORIES
memory organization summarized

- The entire chip contains 2^x x y bits, where
  - *x* is the number of address pins
  - *y* the number of data pins.
- 2^{10} = 1024 = 1K. (*Kilo* = 1000. 1 Kilobyte)

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,
    - 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,*
- For every bit of the PROM, there exists a fuse.
  - PROM is programmed by blowing the fuses.
  - If information burned into PROM is wrong, discard it,
  - 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
EPROM 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.

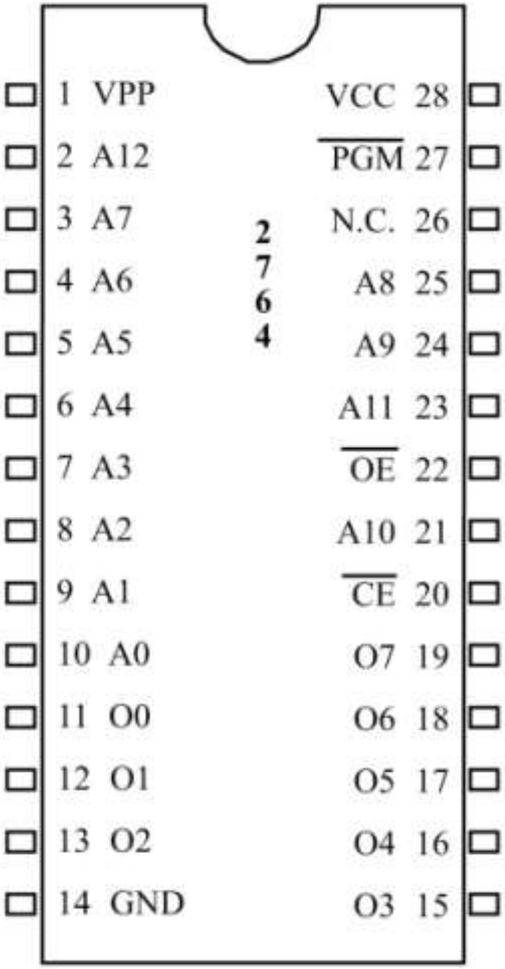


Fig. 10-1 UV-EPROM Chip

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.

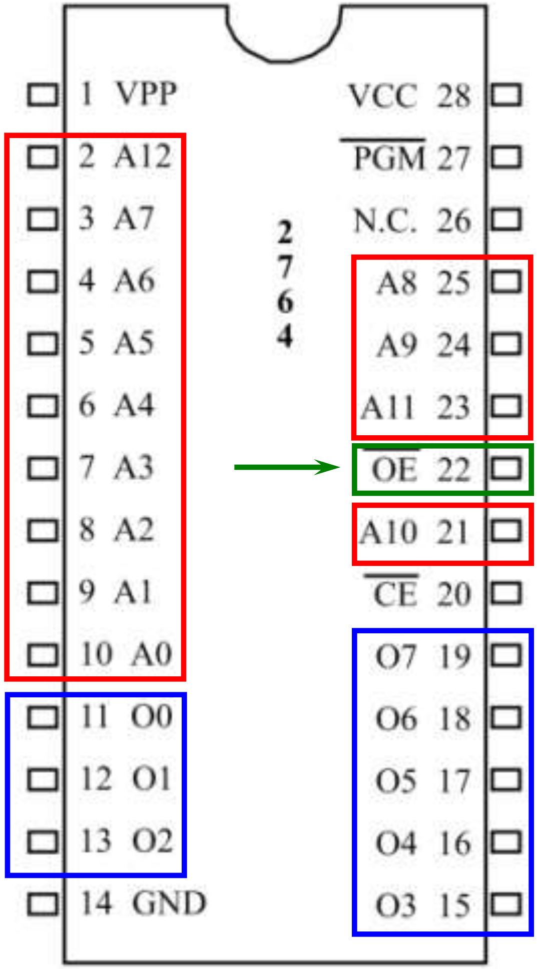
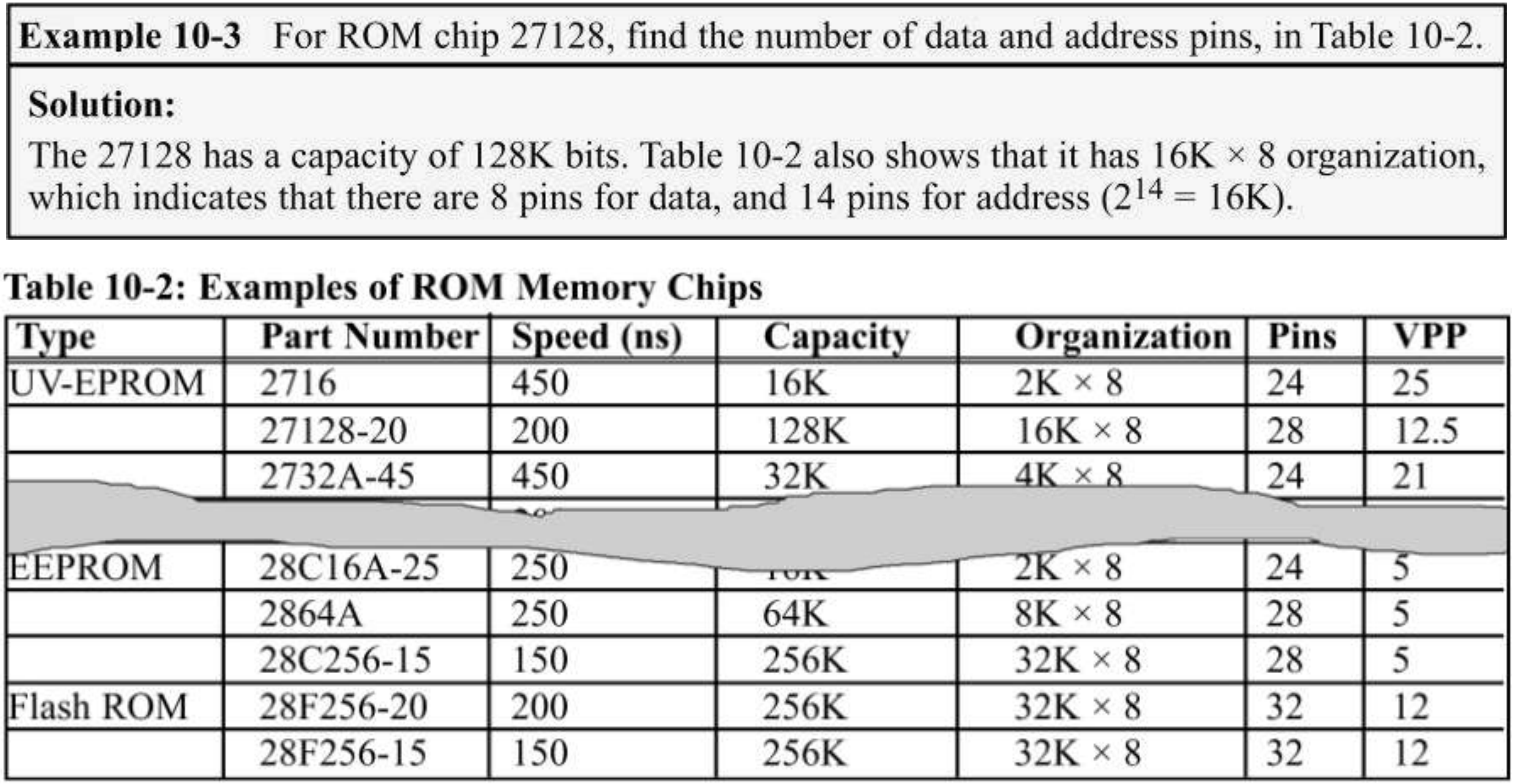


Fig. 10-1 UV-EPROM Chip

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
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
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 6116 pinouts / DRAM packaging issues

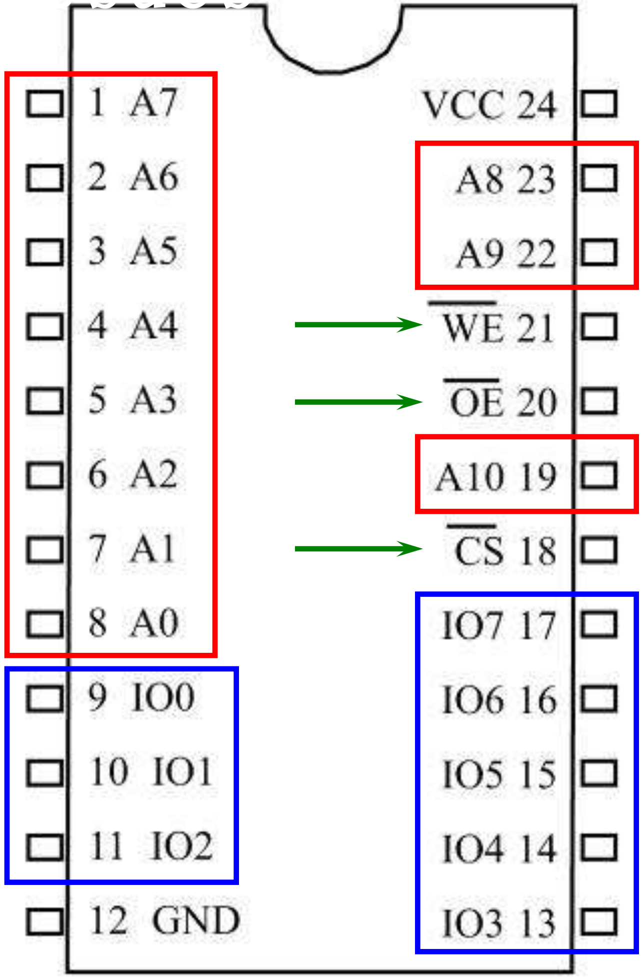


Fig. 10-3 6116 2K x 8 SRAM

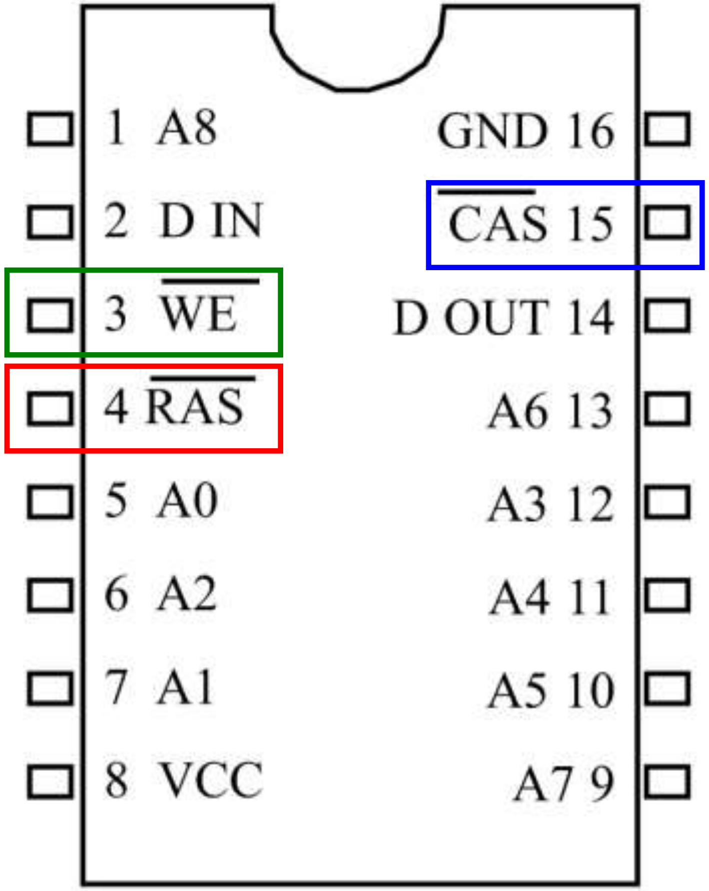


Fig. 10-7 256K × 1 DRAM

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.

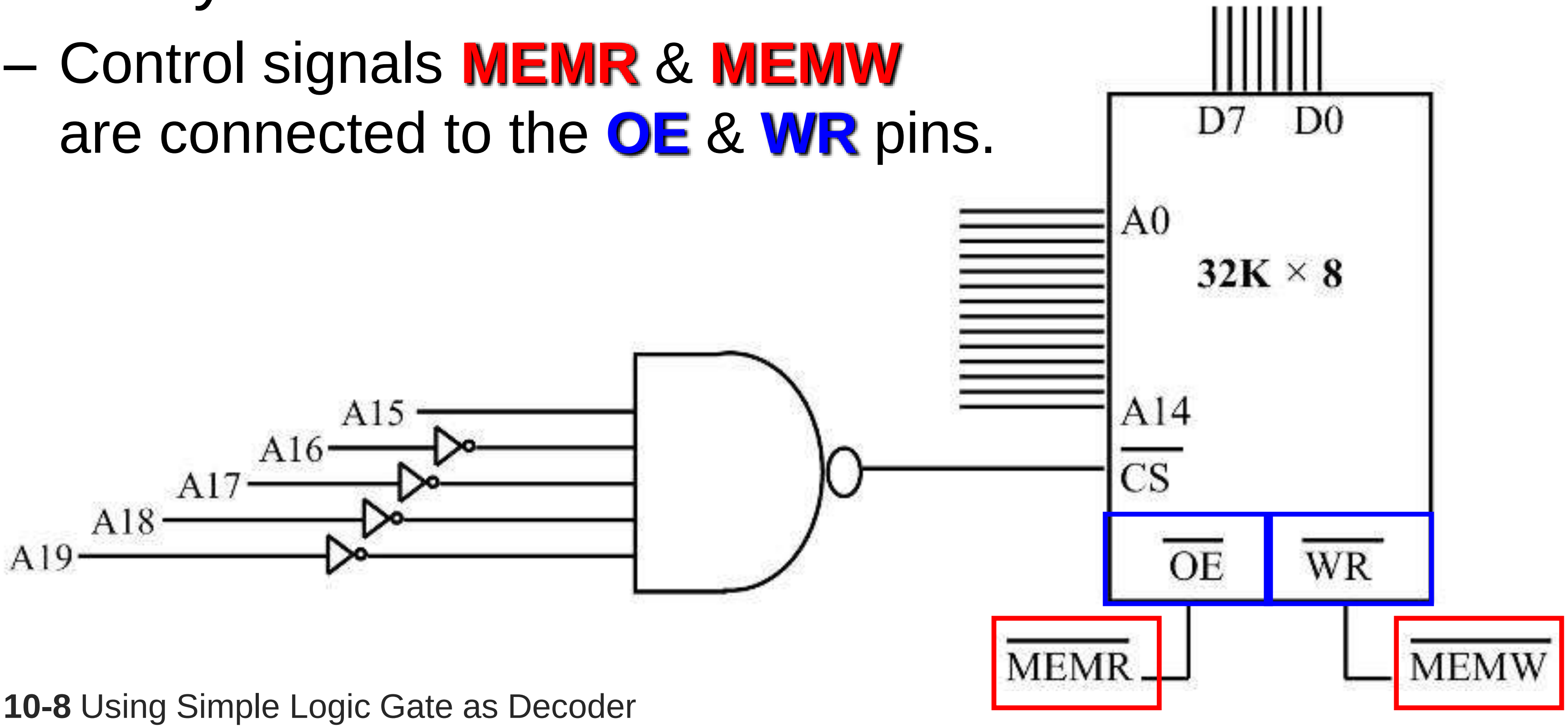


Fig. 10-8 Using Simple Logic Gate as Decoder

10.2: MEMORY ADDRESS DECODING
3x8 Decoder 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.

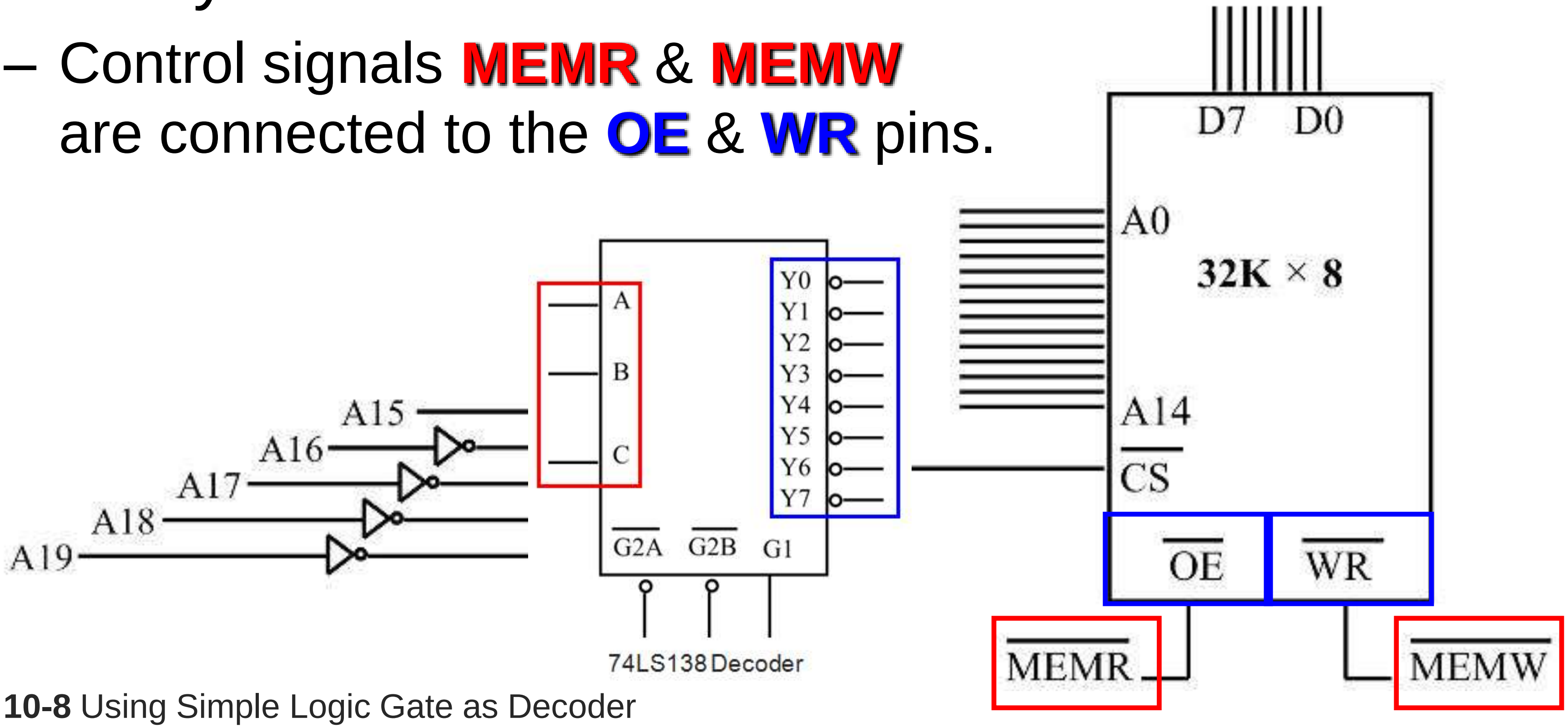


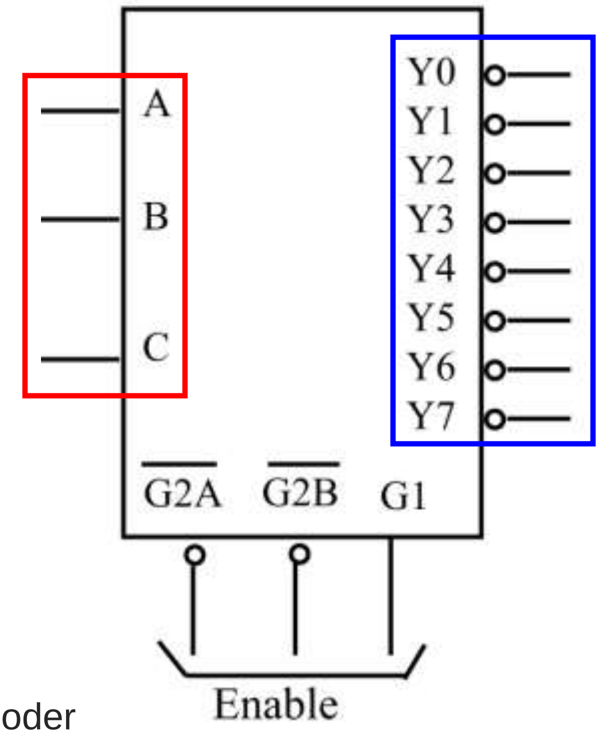
Fig. 10-8 Using Simple Logic Gate as Decoder

10.2: MEMORY ADDRESS DECODING 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**.

Block Diagram



Function Table

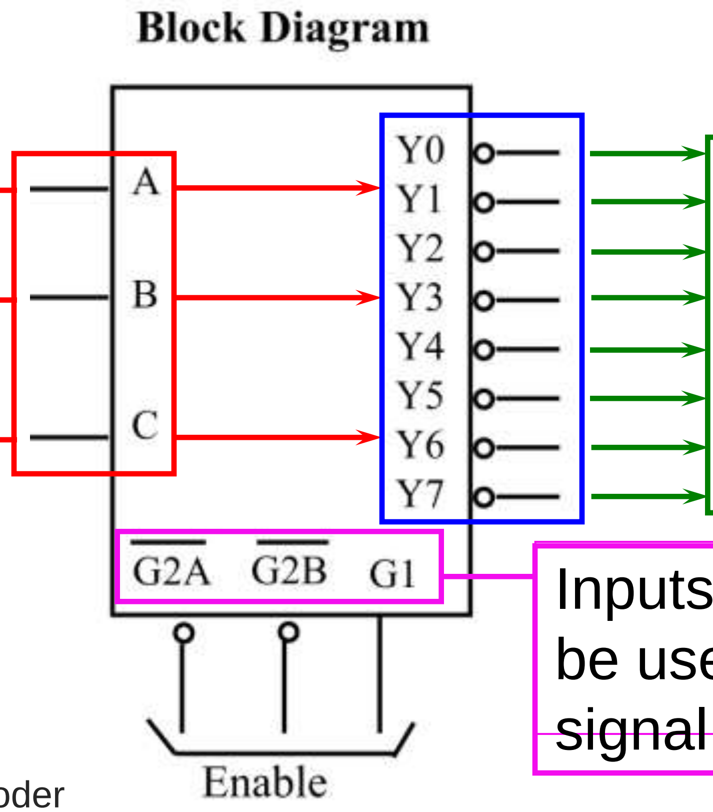
Inputs													
Enable		Select			Outputs								
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L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

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.

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



Each **Y** output connects to the **CS** of a memory chip, allowing control of 8 memory blocks by a single 74LS138.

Inputs **G2A**, **G2B** & **G1**, can be used for address or control signal selection

Fig. 10-11 74LS138 Decoder

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.

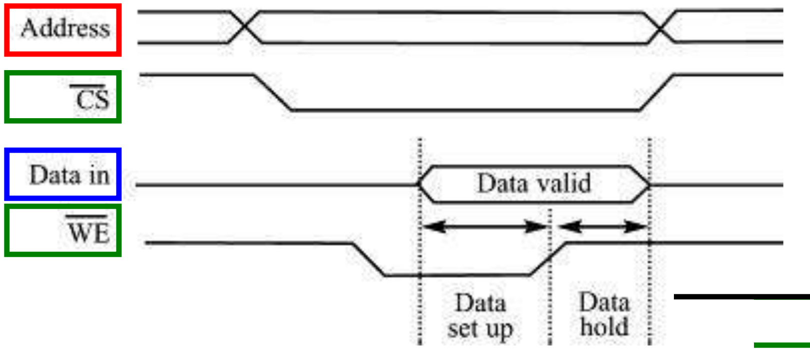


Fig. 10-5 Memory Write Timing for SRAM

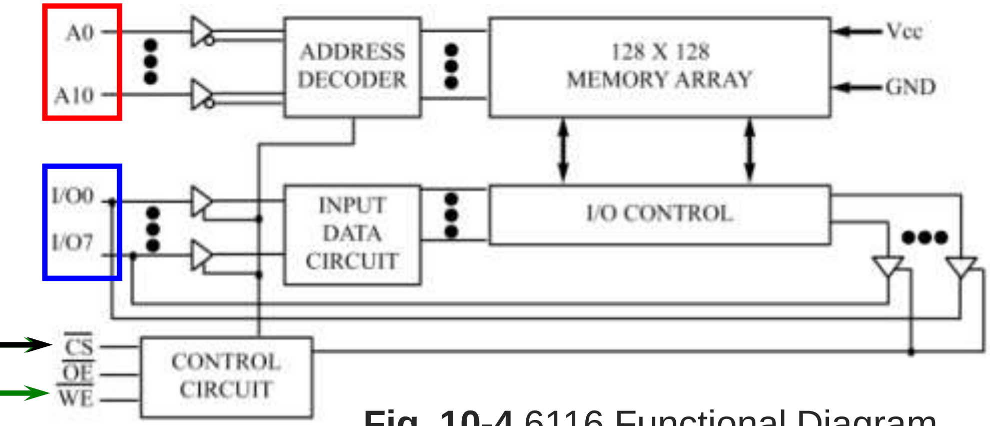


Fig. 10-4 6116 Functional Diagram

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, a high-to-low pulse on the **OE** pin will read the data out of the chip.

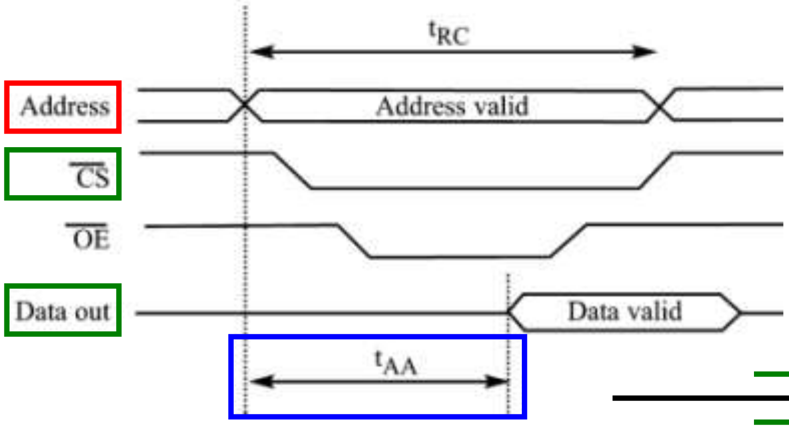


Fig. 10-6 Memory Read Timing for SRAM

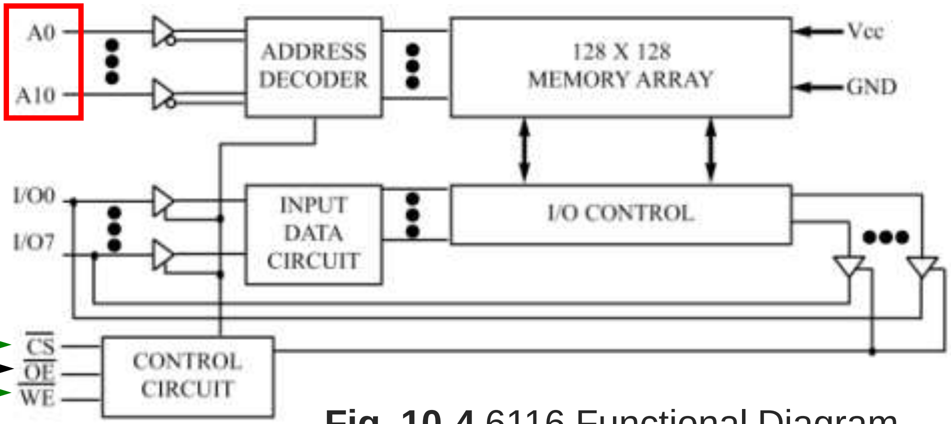
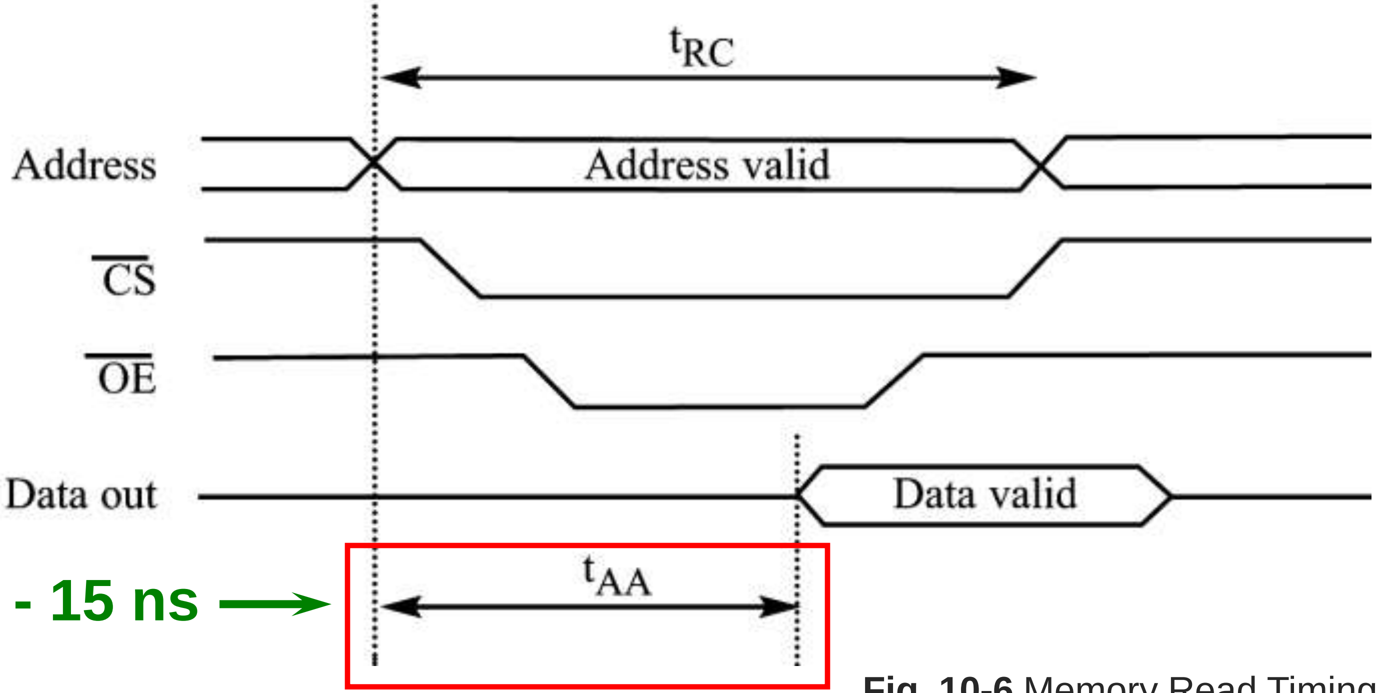


Fig. 10-4 6116 Functional Diagram

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. Speed for the 6116 chip can vary from **100 ns** to **15 ns**.



100 ns - 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.

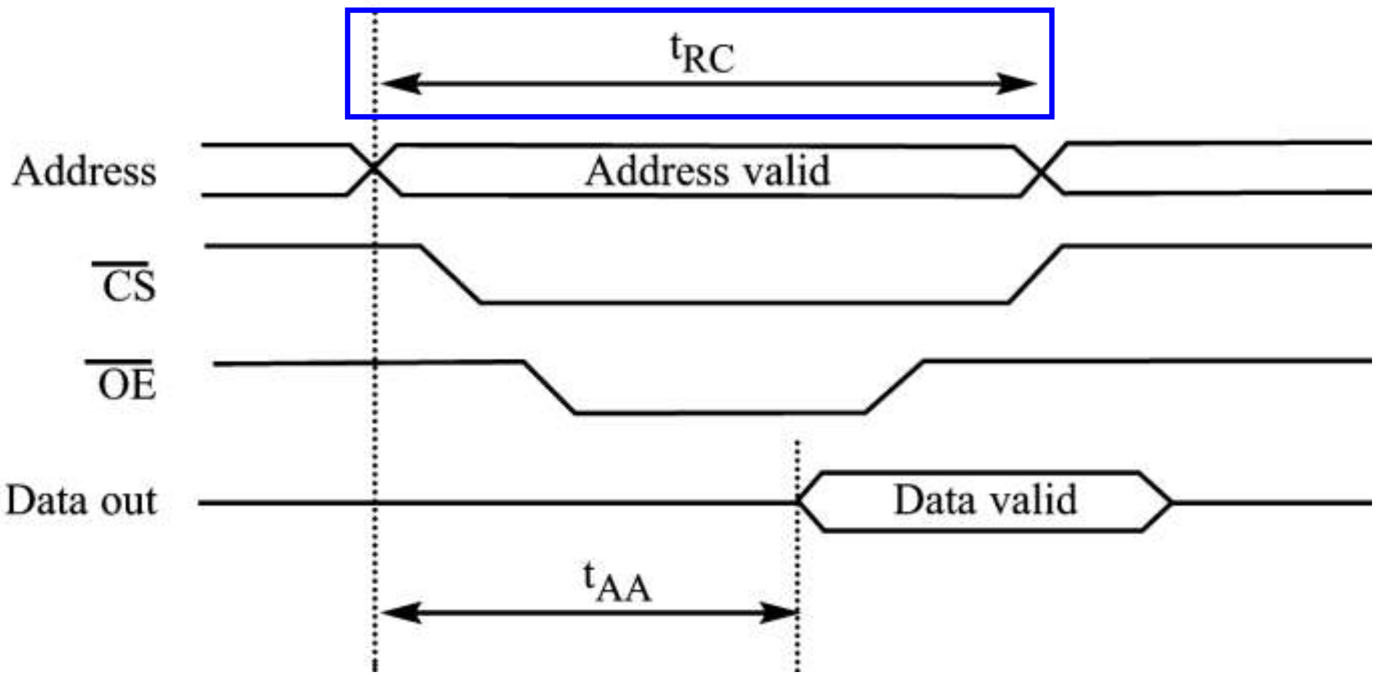


Fig. 10-6 Memory Read Timing for SRAM

10.1: SEMICONDUCTOR MEMORIES
SRAM 6116 access & read time

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

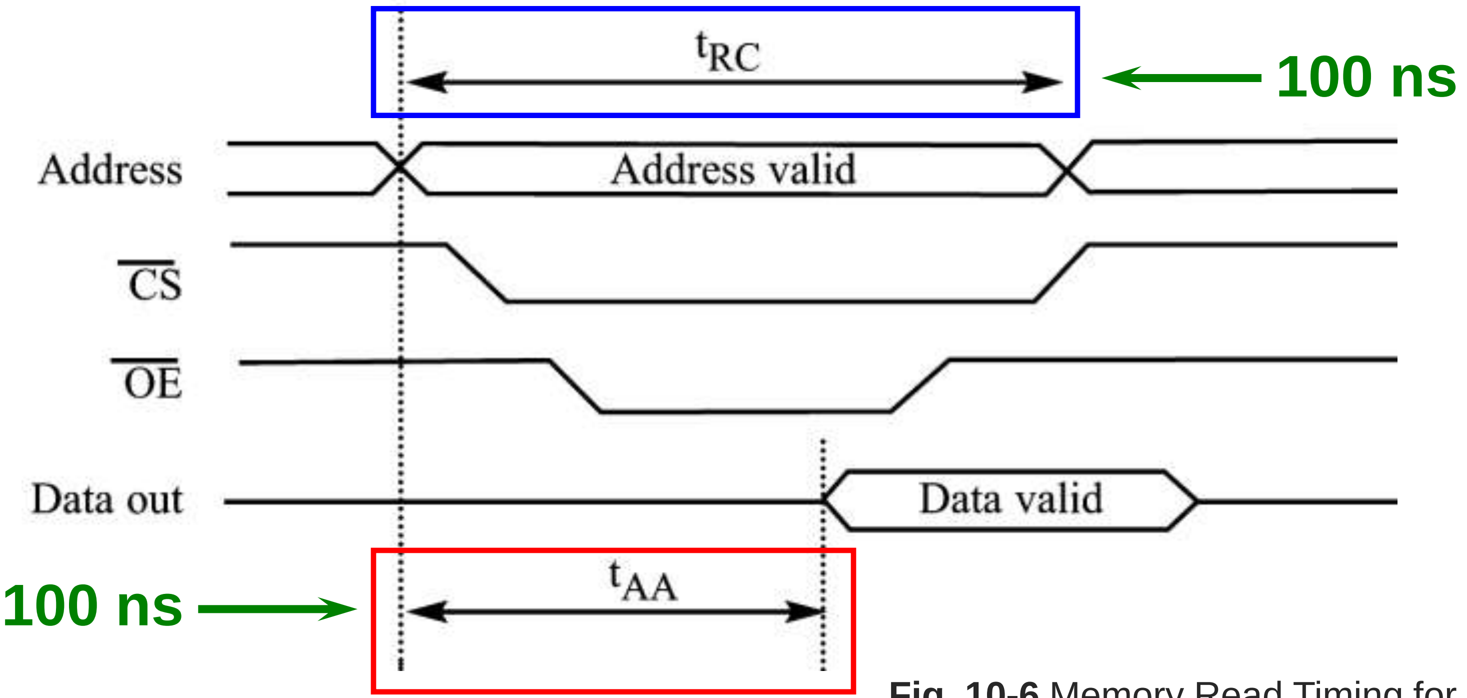


Fig. 10-6 Memory Read Timing for SRAM

10.1: SEMICONDUCTOR MEMORIES
DRAM 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.

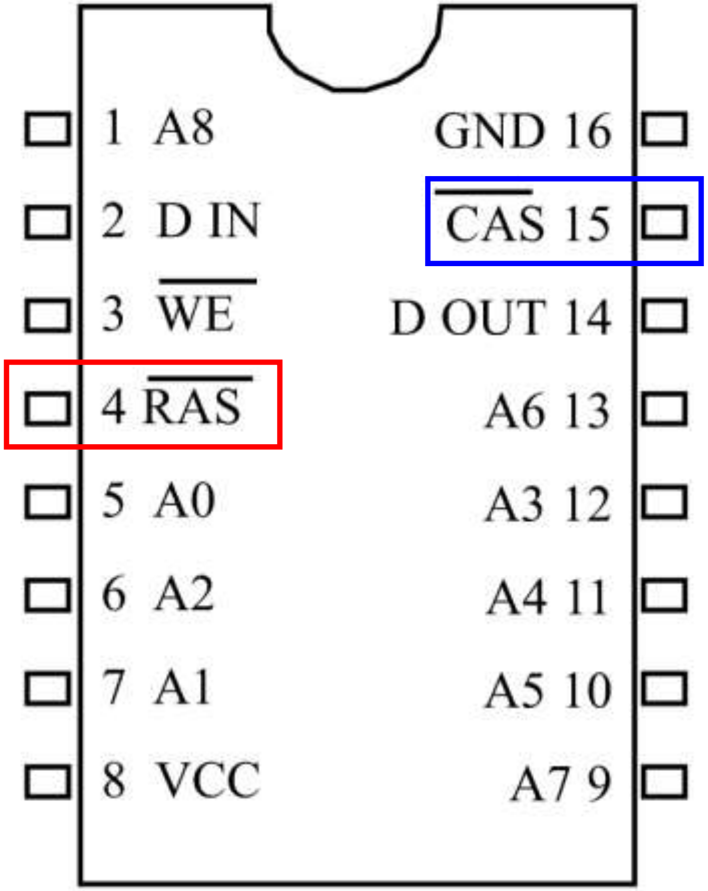


Fig. 10-7 256K × 1 DRAM

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.

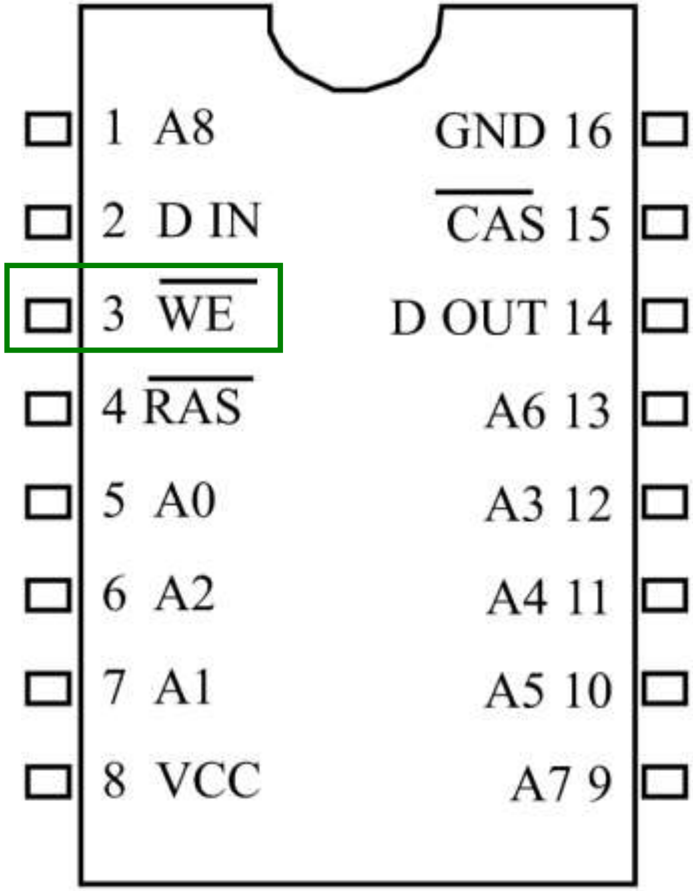
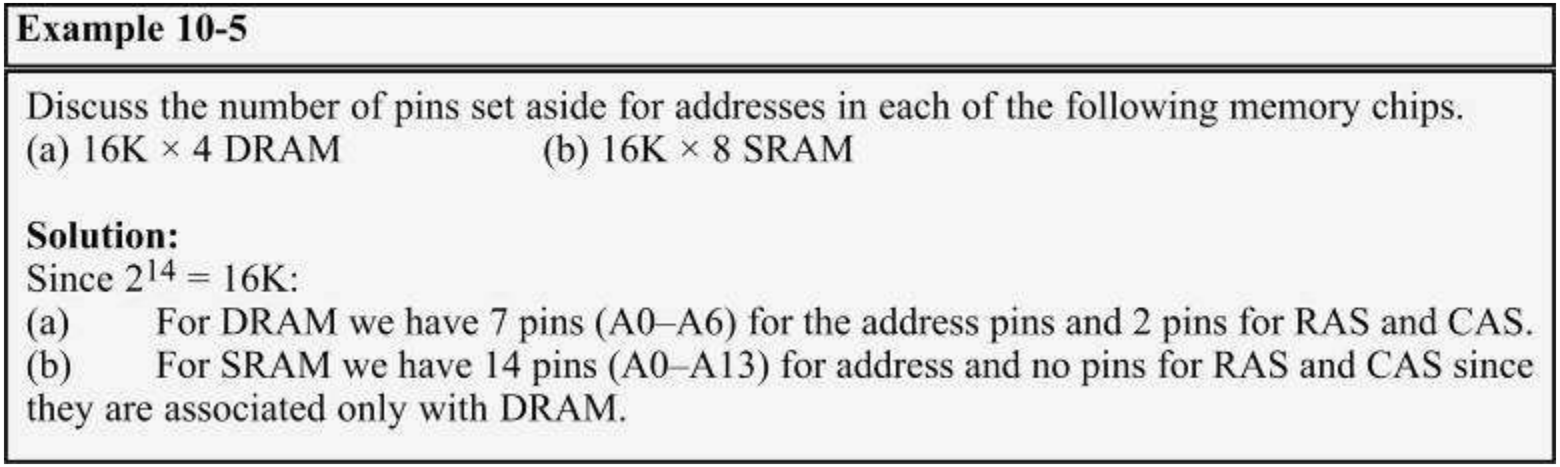


Fig. 10-7 256K × 1 DRAM

10.1: 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

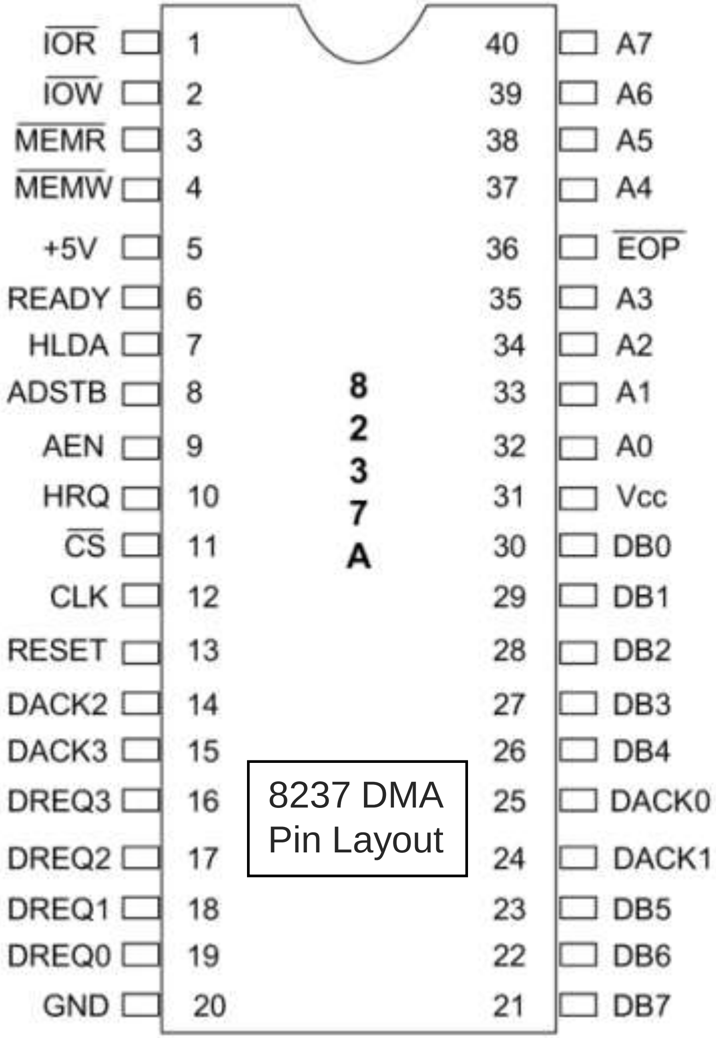
Solution:
Since $2^{14} = 16K$:
(a) For DRAM we have 7 pins (A0–A6) for the address pins and 2 pins for RAS and CAS.
(b) For SRAM we have 14 pins (A0–A13) for address and no pins for RAS and CAS since they are associated only with DRAM.

CONCEPT OF DMA

- There is often need to transfer a many bytes between memory & peripherals like disk drives.
- The Intel 8237 DMAC (direct memory access Controller) chip functions to provide a direct connection between peripherals and memory,
- When DMA needs the buses, it sends a HOLD signal to the CPU, and the CPU responds with a HLDA (hold acknowledge) signal.
  - Indicating the DMA can use the buses.

8237 DMA INTERFACING IN THE IBM PC

- The 8237 DMA.



8237 DMA
Pin Layout

9.3: 8-BIT SECTION OF ISA BUS
one bus, two masters

- 8088 is unacceptably slow for transferring large numbers of bytes of data, as in hard disk transfers.
  - The 8237 chip is used for large data transfers.
- The 8237 must have access to all three buses.
  - *Bus arbitration*, achieved by the **AEN** (address enable) generation circuitry allows either the 8088 processor or the 8237 DMA to bus gain control.

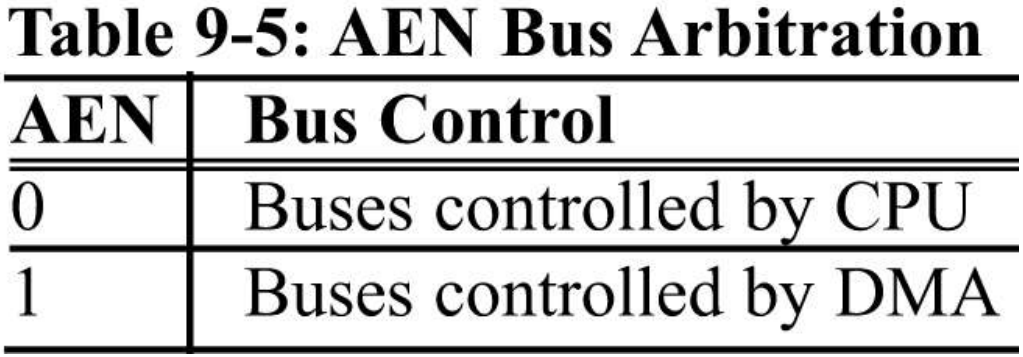


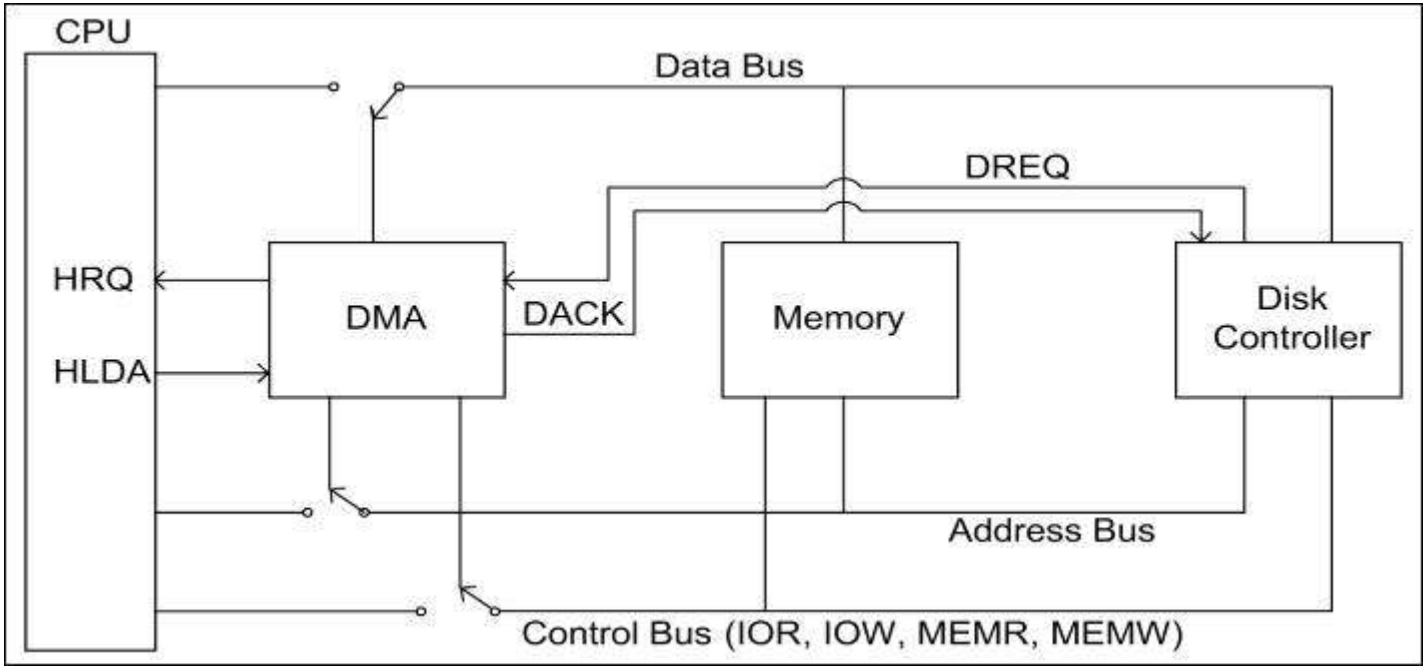
Table 9-5: AEN Bus Arbitration

AEN	Bus Control
0	Buses controlled by CPU
1	Buses controlled by DMA

CONCEPT OF DMA
bus sharing

- While DMA uses the buses, the CPU is idle, and when the CPU uses the bus, DMA is sitting idle.
  - After DMA finishes, it makes HOLD go *low* & the CPU will regain control over the buses

Fig. 15-1
DMA Usage
of System Bus



CONCEPT OF DMA
steps involved in a DMA transfer

- DMA can only *transfer* information.
  - It cannot decode and execute instructions.
- When the CPU receives a HOLD request from DMA, it finishes the present bus cycle (but not necessarily the present instruction) before it hands over control of the buses to the DMA.
- To transfer a block of data from memory to I/O, DMA must know:
  - The address of the beginning of the data block. (address of the first byte of data)
  - The number of bytes (count) it needs to transfer.

CONCEPT OF DMA
steps involved in a DMA transfer

- DMA Transfer Steps:
  - 1. A peripheral device (like the disk controller) will request DMA service by pulling DREQ (DMA request) *high*.
  - 2. DMA puts a *high* on its HRQ (hold request), signaling the CPU through its HOLD pin that it needs to the buses.
  - 3. The CPU finishes the present bus cycle & responds to DMA by putting *high* on HLDA (hold acknowledge).
    - Telling the 8237 DMA it can use the buses to perform its task.
    - HOLD must remain *active-high* while DMA performs its task.
  - 4. DMA will activate DACK (DMA acknowledge), which tells the peripheral device it will start to transfer the data.

CONCEPT OF DMA
steps involved in a DMA transfer

- DMA Transfer Steps:
  - 5. DMA starts to transfer data from memory to the I/O peripheral by putting the address of the first byte of the block on the address bus and activating MEMR.
    - Reading the byte from memory into the data bus; it then activates IOW to write the data to the peripheral.
    - DMA decrements the counter, increments the address pointer & repeats the process until the count reaches zero.
  - 6. After the DMA has finished, it will deactivate HRQ, signaling the CPU that it can regain control over its buses.

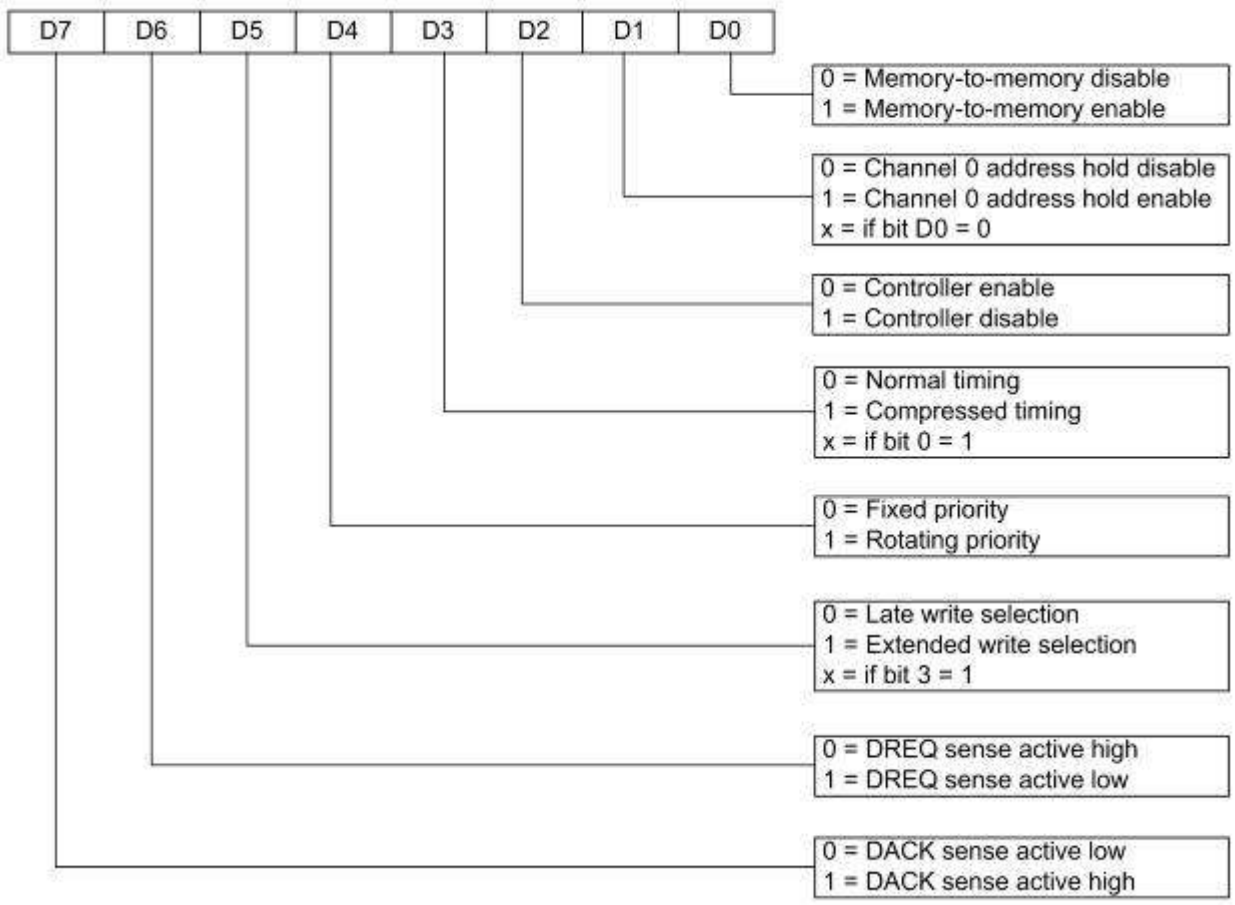
8237 DMA CHIP PROGRAMMING
command register

8237 is capable of transferring data…

From a peripheral device to memory. (reading from disk)

From memory to a peripheral device (writing a file to disk)

From memory to memory. (Shadow RAM)



D7 D6 D5 D4 D3 D2 D1 D0

D0:
0 = Memory-to-memory disable
1 = Memory-to-memory enable

D1:
0 = Channel 0 address hold disable
1 = Channel 0 address hold enable
x = if bit D0 = 0

D2:
0 = Controller enable
1 = Controller disable

D3:
0 = Normal timing
1 = Compressed timing
x = if bit 0 = 1

D4:
0 = Fixed priority
1 = Rotating priority

D5:
0 = Late write selection
1 = Extended write selection
x = if bit 3 = 1

D6:
0 = DREQ sense active high
1 = DREQ sense active low

D7:
0 = DACK sense active low
1 = DACK sense active high

8237 Command Register Format