



## Chapter 2

# MEMORY DEVICES

Faculty of Computer Science and Engineering  
Department of Computer Engineering



Nguyen Quang Huy  
[huynguyen@cse.hcmut.edu.vn](mailto:huynguyen@cse.hcmut.edu.vn)

# LOGIC DESIGN 2

MSI logic circuits

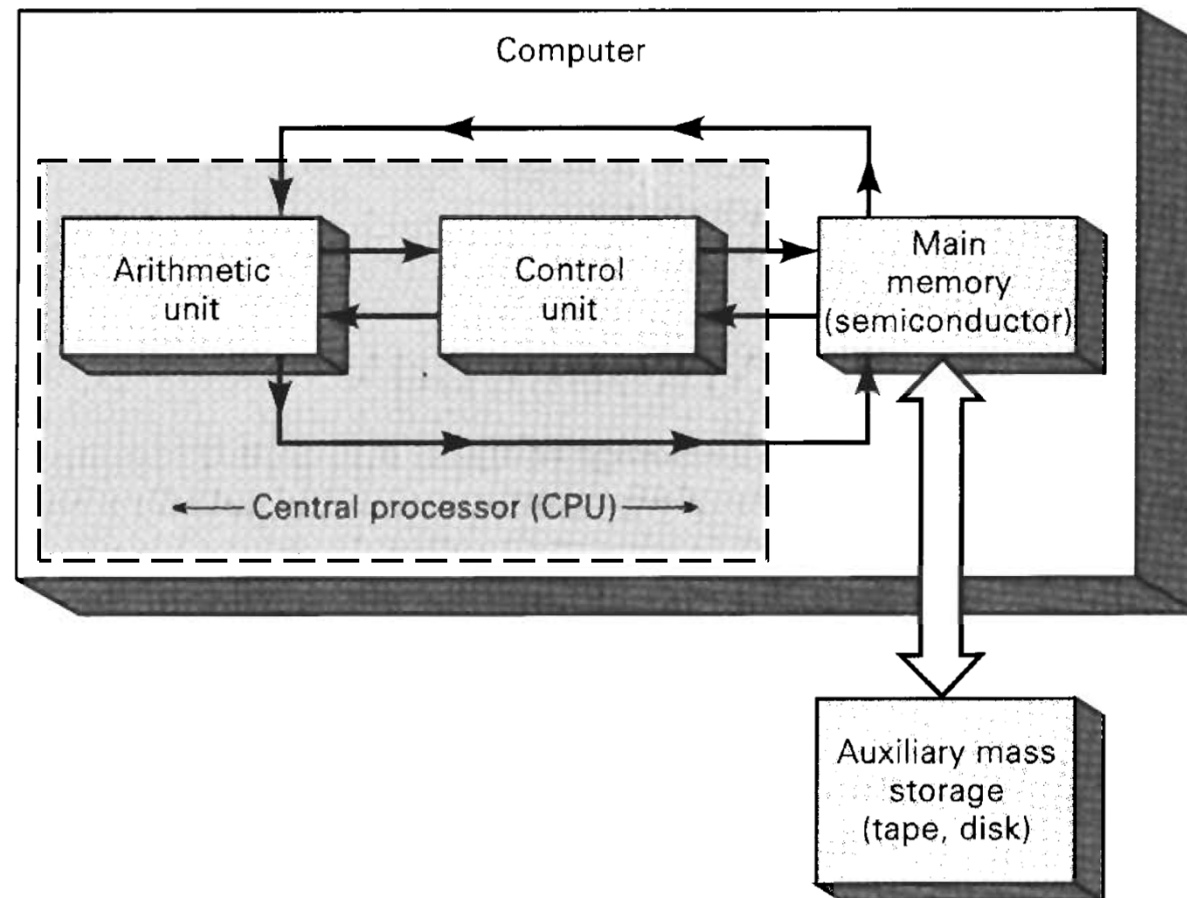
**Memory**

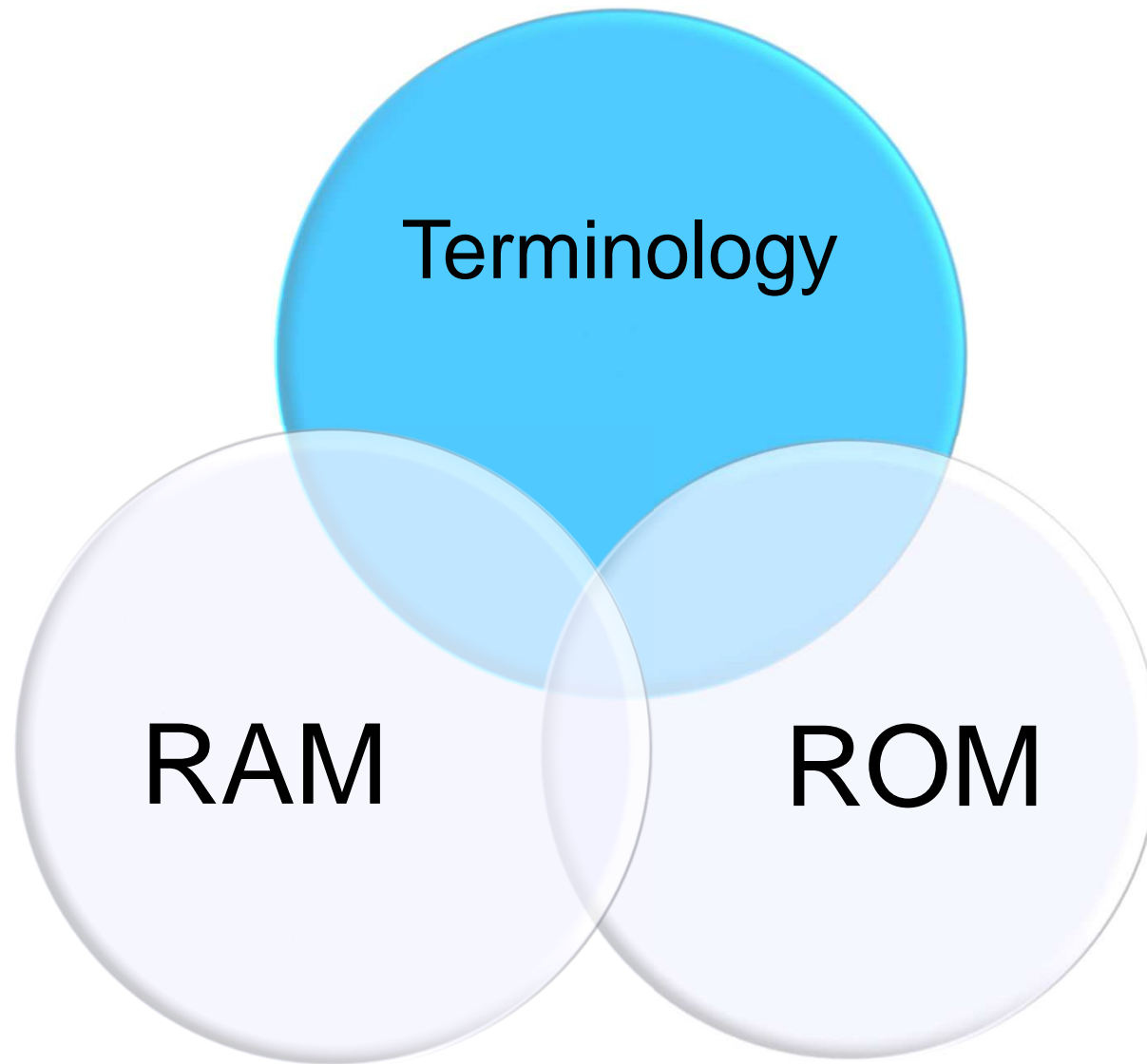
ADC / DAC

Logic family

# Introduction

- Main memory (working memory)
- Auxiliary memory (mass storage)





# Memory Cell

MEM A							

MEM B															

# Byte vs. Word

MEM A							

MEM B															



**Word size of MEM A ?**  
**Word size of MEM B ?**

# Memory Terminology (1)

- **Memory Cell:** device or an electrical circuit used to store a single bit (0 or 1)
- **Memory Word:** a group of bits (cells) in a memory that represents instructions or data of some type
  - Typically range from 4 – 64 bits, depend on the size of the computer
- **Byte:** a group of 8 bits
- **Capacity:** number of bits can be stored in a particular memory device / complete memory system
  - A memory that can store 4096 20-bit words
    - Capacity =  $4096 \times 20 = 4K \times 20 = 81920$  bits
  - $1K \rightarrow 2^{10} = 1024$ ,       $1M \rightarrow 2^{20}$ ,       $1G \rightarrow 2^{30}$

## Example (1)

- A certain semiconductor memory chip is specified as **2K x 8**
  - How many words can be stored on this chip?
  - What is the word size?
  - How many total bits can this chip store?
- **Solution**
  - Number of words:  $2K = 2 \times 1024 = 2048$  words
  - Each word is **8-bits (one byte)**
  - The total number of bits:  $2048 \times 8 = 16,384$  bits



## Example (2)

- Which memory stores the most bits ?
  - A 5M x 8 memory
  - A memory that stores 1M words at a word size of 16 bits
- **Solution**
  - $5M \times 8 = 5 \times 1,048,576 \times 8 = 41,943,040$  bits
  - $1M \times 16 = 1,048,576 \times 16 = 16,777,216$  bits
  - The 5M x 8 memory stores **more bits**.

# Memory Terminology (2)

- **Density**
- **Address:** location of a word in a memory (*unique*)

Addresses

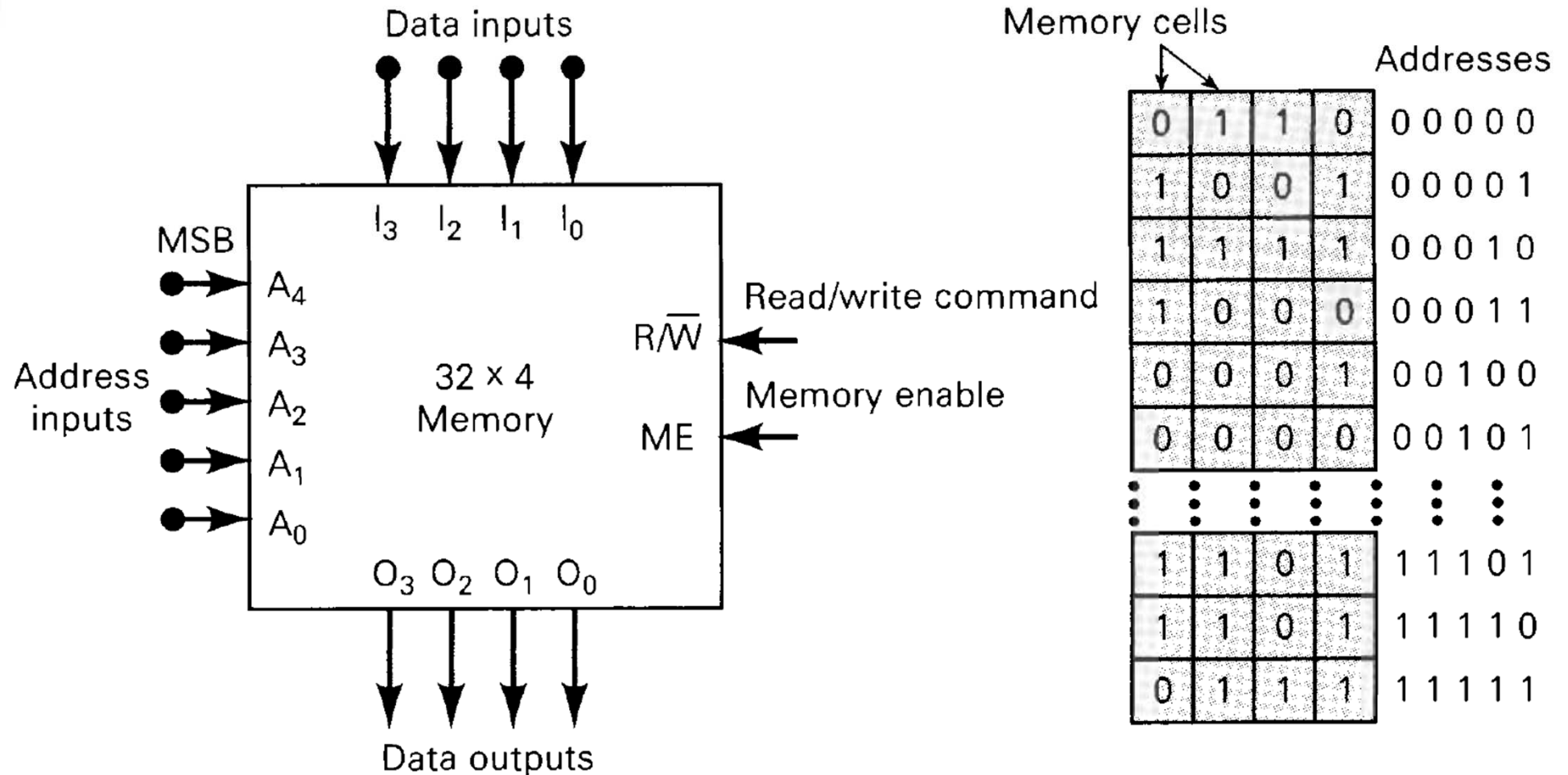
000	Word 0
001	Word 1
010	Word 2
011	Word 3
100	Word 4
101	Word 5
110	Word 6
111	Word 7

- **Read Operation:** *fetch* operation
- **Write Operation:** *store* operation

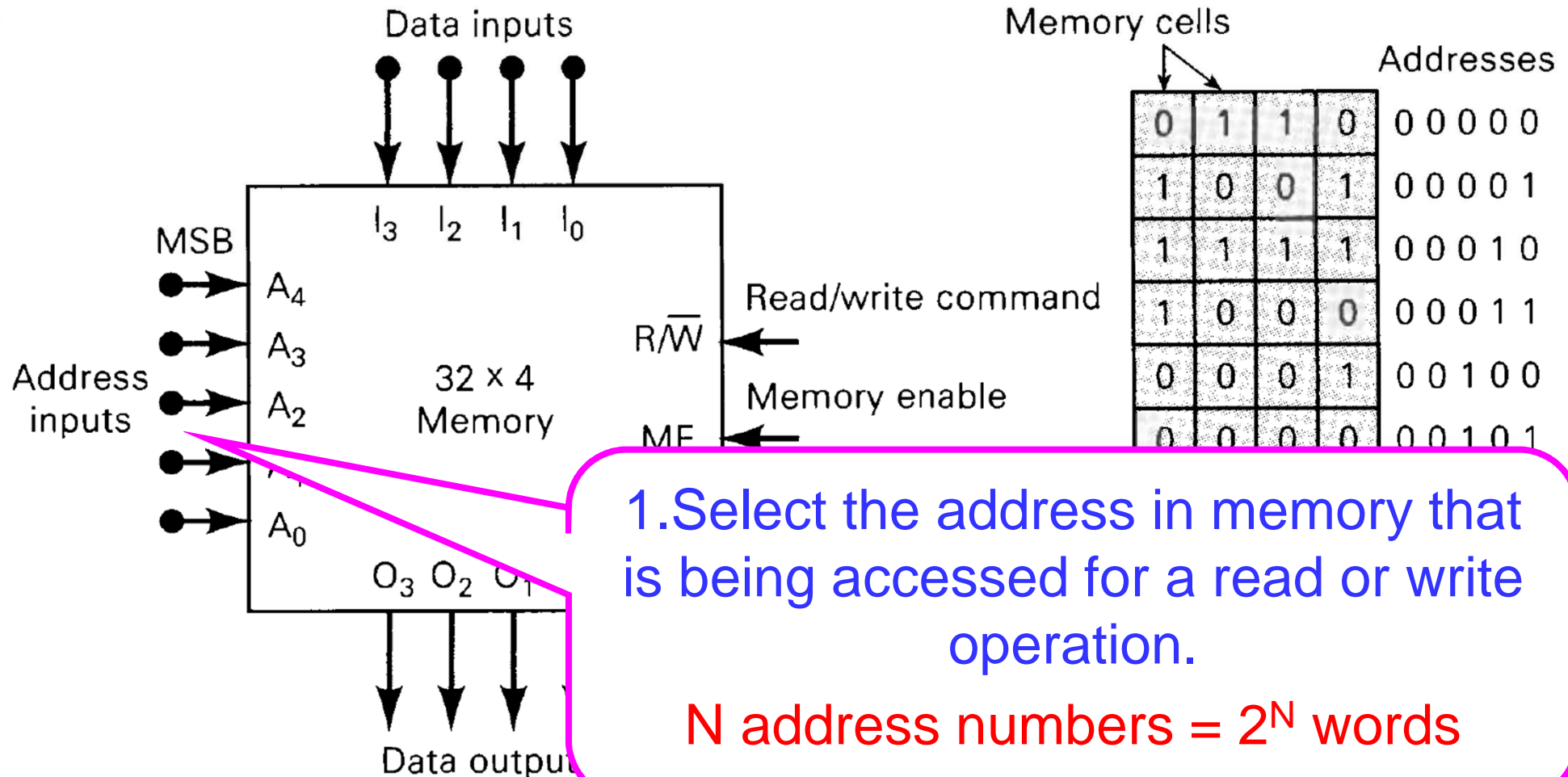
# Memory Terminology (3)

- Access Time (*for read operation*)
- Volatile, Nonvolatile Memory
- Random-Access Memory (RAM): *same access time*
- Sequential-Access Memory (SAM)
- Read-Only Memory (ROM)
- Static Memory Devices
- Dynamic Memory Devices: *refresh operation*
- Main Memory (*working memory*)
- Auxiliary Memory (*mass storage*)

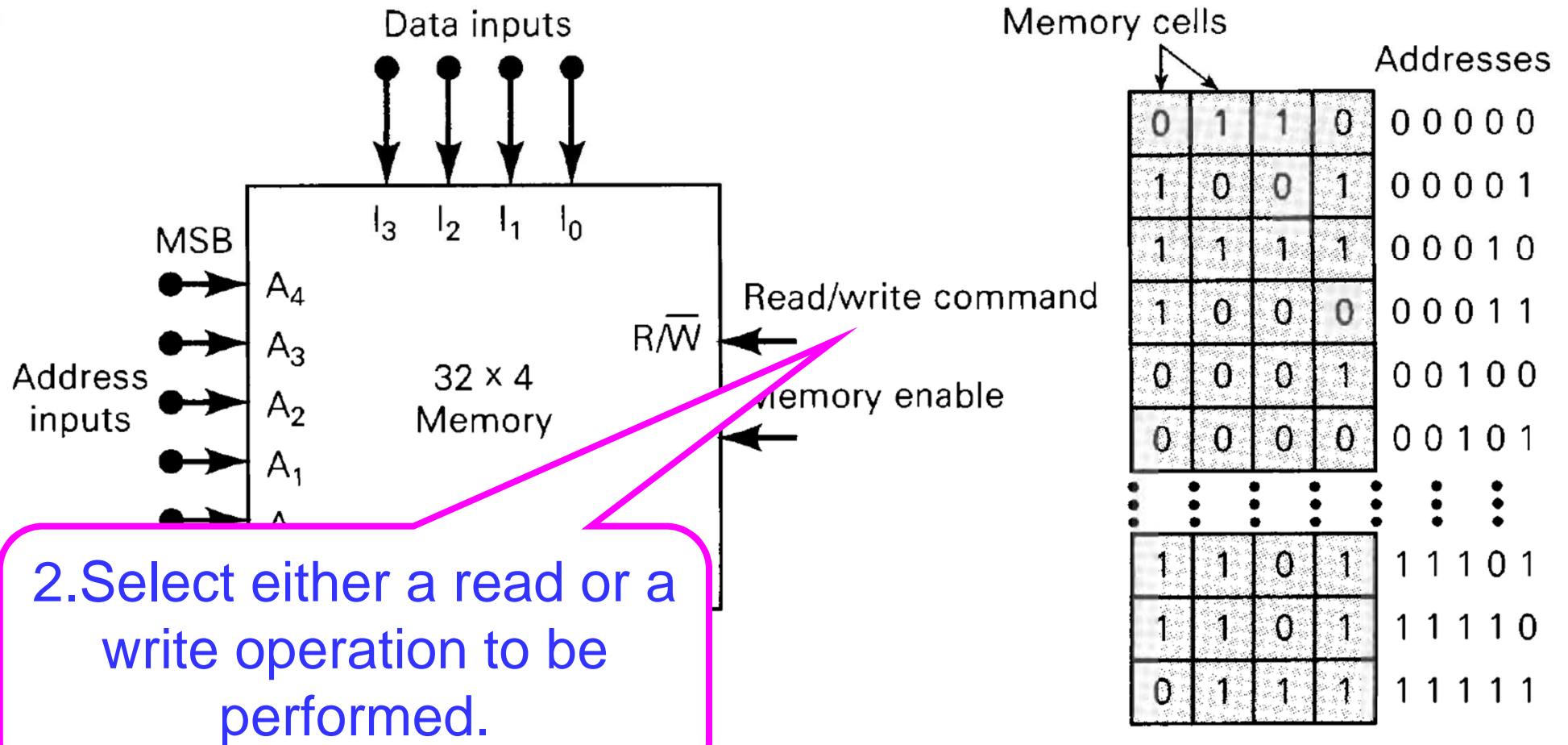
# General Memory Operation



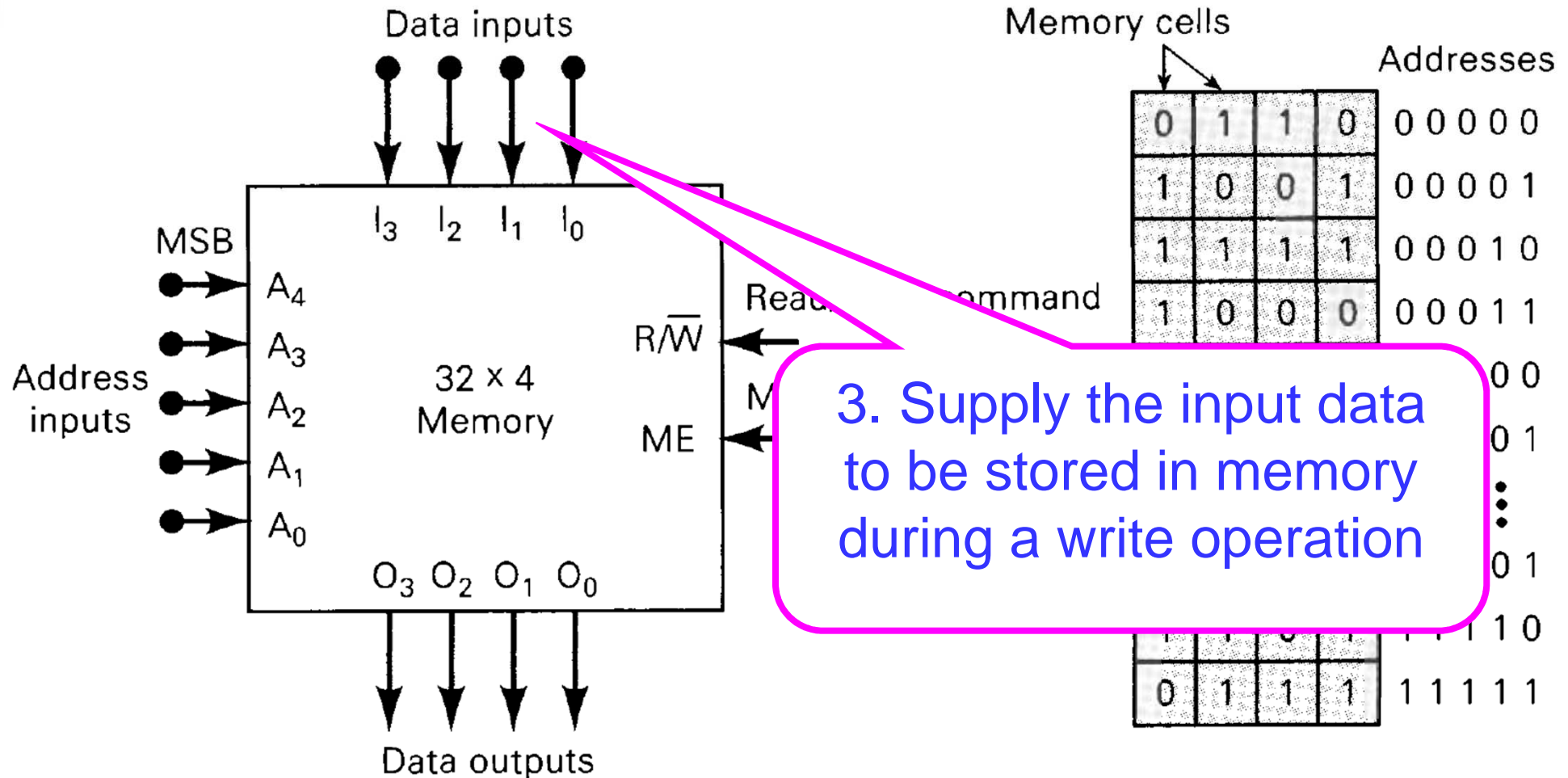
# General Memory Operation



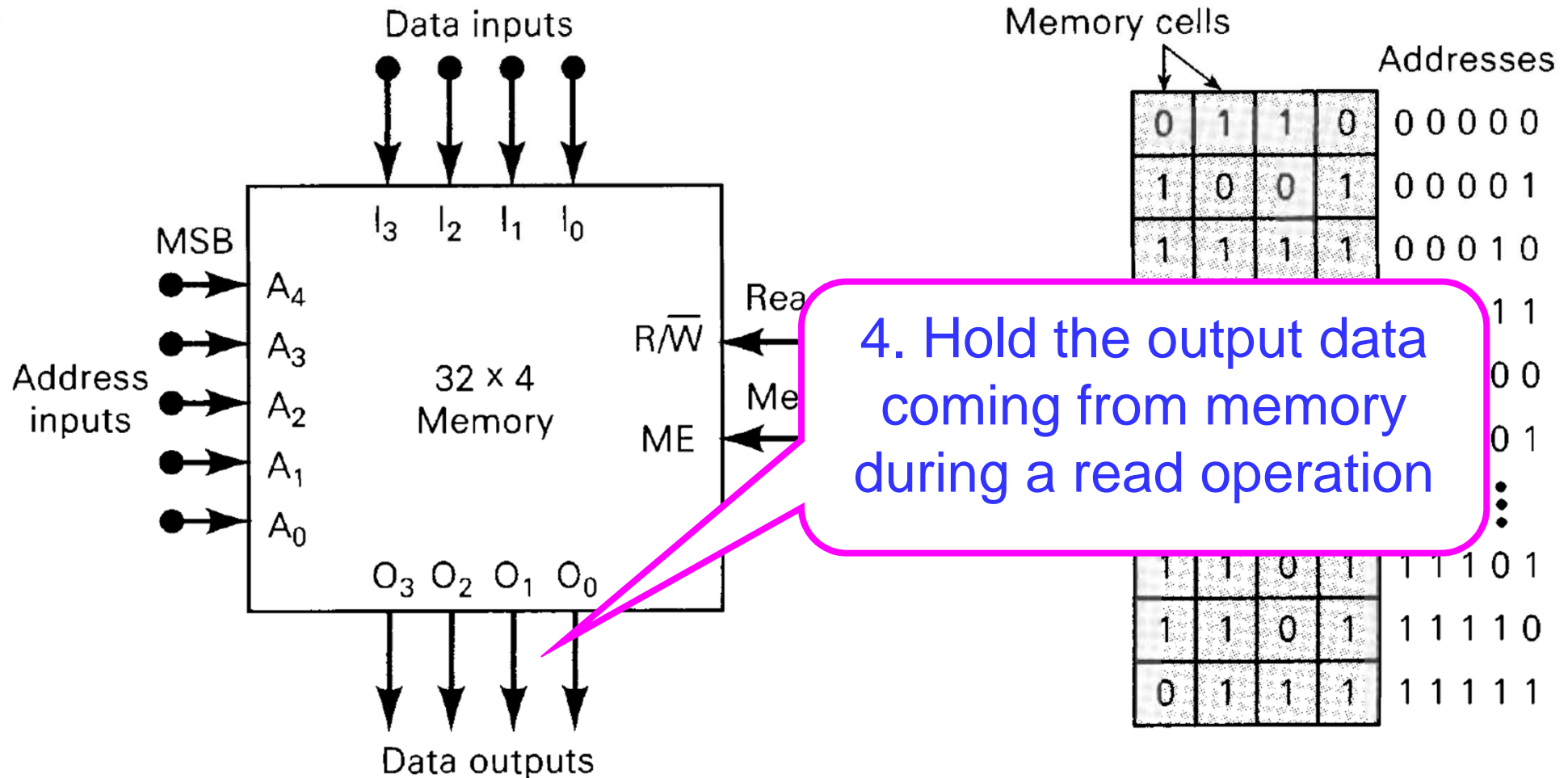
# General Memory Operation



# General Memory Operation

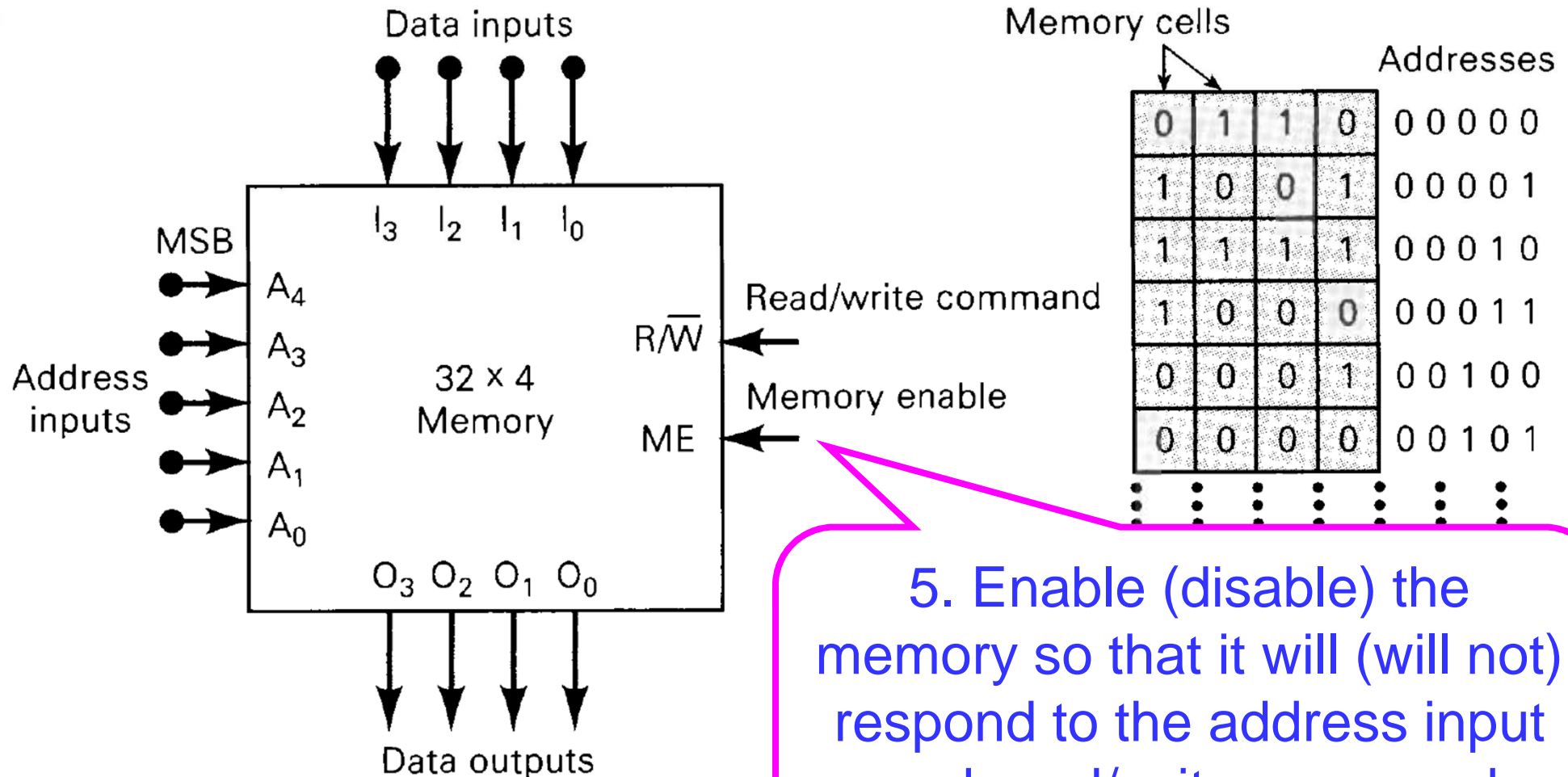


# General Memory Operation





# General Memory Operation



5. Enable (disable) the memory so that it will (will not) respond to the address input and read/write command

# Example

- Describe the conditions at each **input and output** when the contents of address location **00100** are to be **read**

- Solution**

- Address inputs: 00100
- Data inputs: xxxx, not used
- R/W: HIGH
- Memory Enable: HIGH
- Data outputs: 0001

Memory cells

0	1	1	0	0 0 0 0 0
1	0	0	1	0 0 0 0 1
1	1	1	1	0 0 0 1 0
1	0	0	0	0 0 0 1 1
0	0	0	1	0 0 1 0 0
0	0	0	0	0 0 1 0 1
⋮	⋮	⋮	⋮	⋮
1	1	0	1	1 1 1 0 1
1	1	0	1	1 1 1 1 0
0	1	1	1	1 1 1 1 1

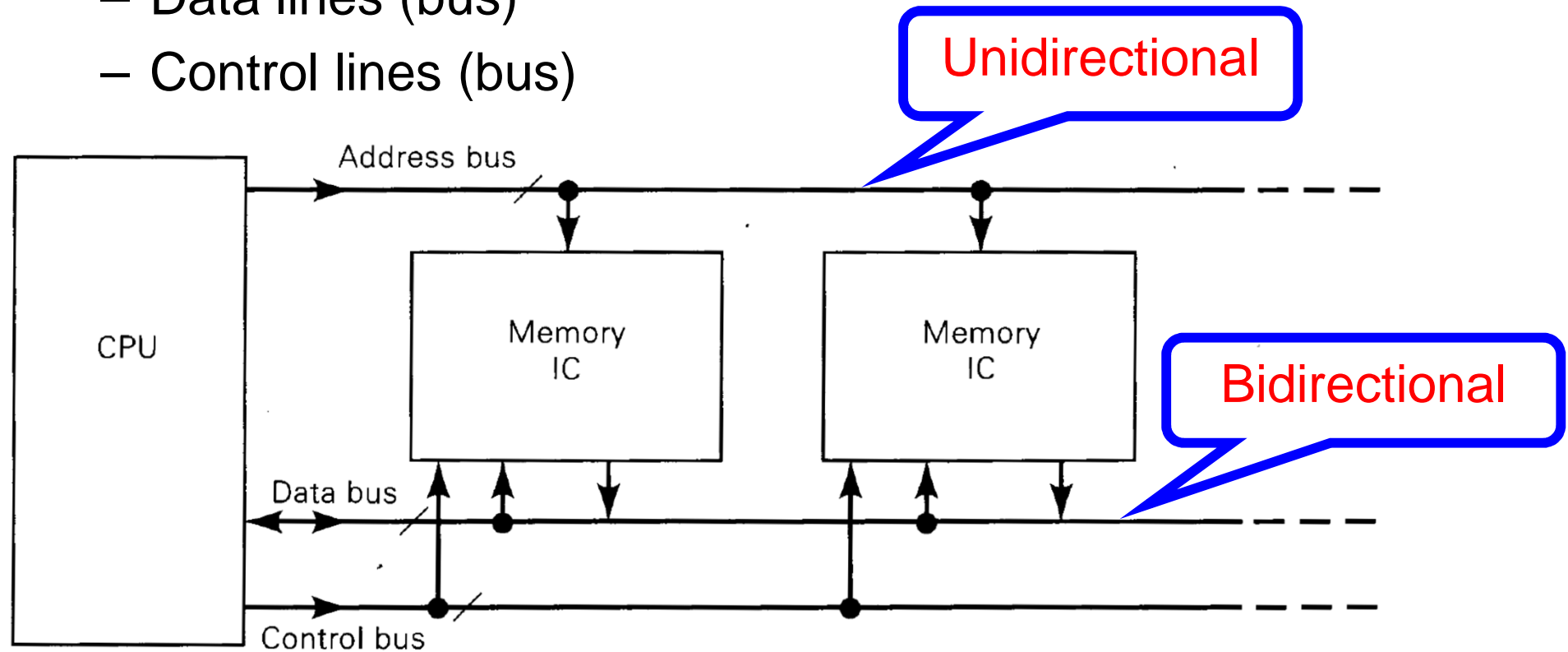
Adresse:

# Example

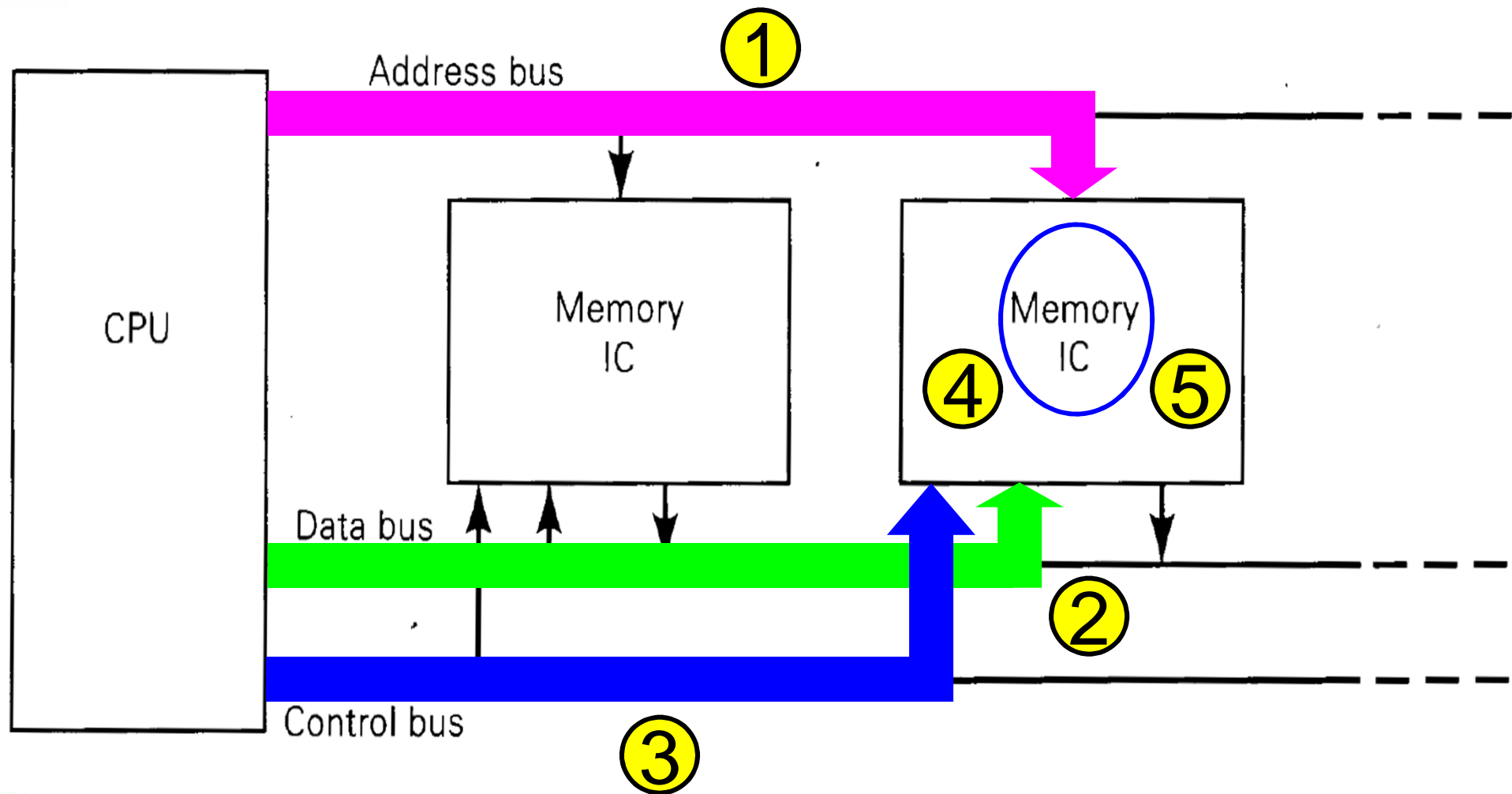
- A certain memory has a capacity of **4K x 8**
  - How many data input and output lines does it have?
  - How many address lines does it have?
  - What is its capacity in bytes?
- **Solution**
  - 8-bit data input/output (**word size** = 8)
  - Total number of words = 4K =  $4 \times 2^{10} = 4096$  words
    - Number of address =  $4096 = 2^{12}$
    - Require a 12-bit address code
  - 1 byte = 8 bit → capacity = 4096 bytes

# CPU-Memory Connections

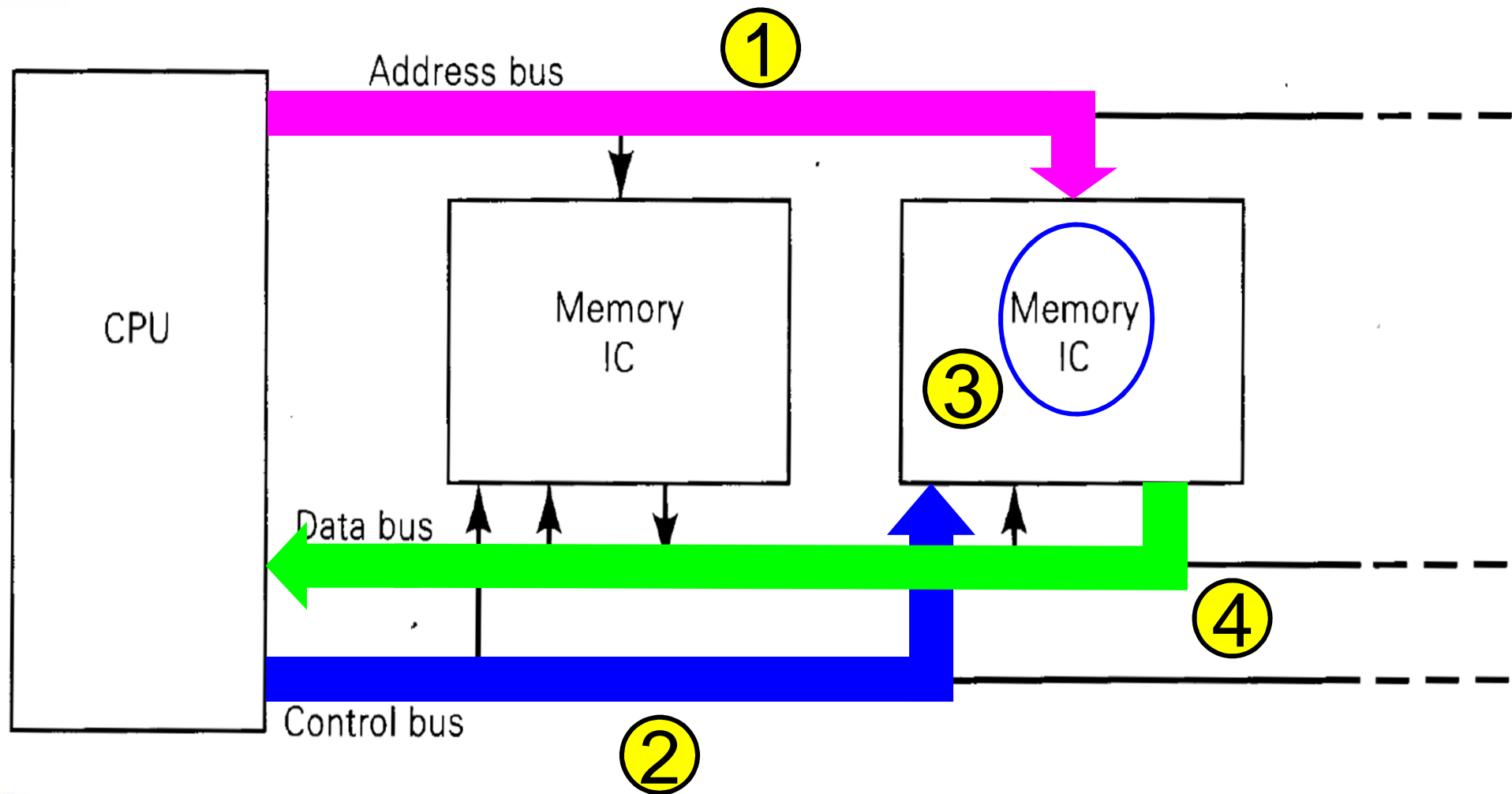
- Memory ICs (RAM, ROM) are interfaced to the CPU over three *groups of signal lines or buses*
  - Address lines (bus)
  - Data lines (bus)
  - Control lines (bus)

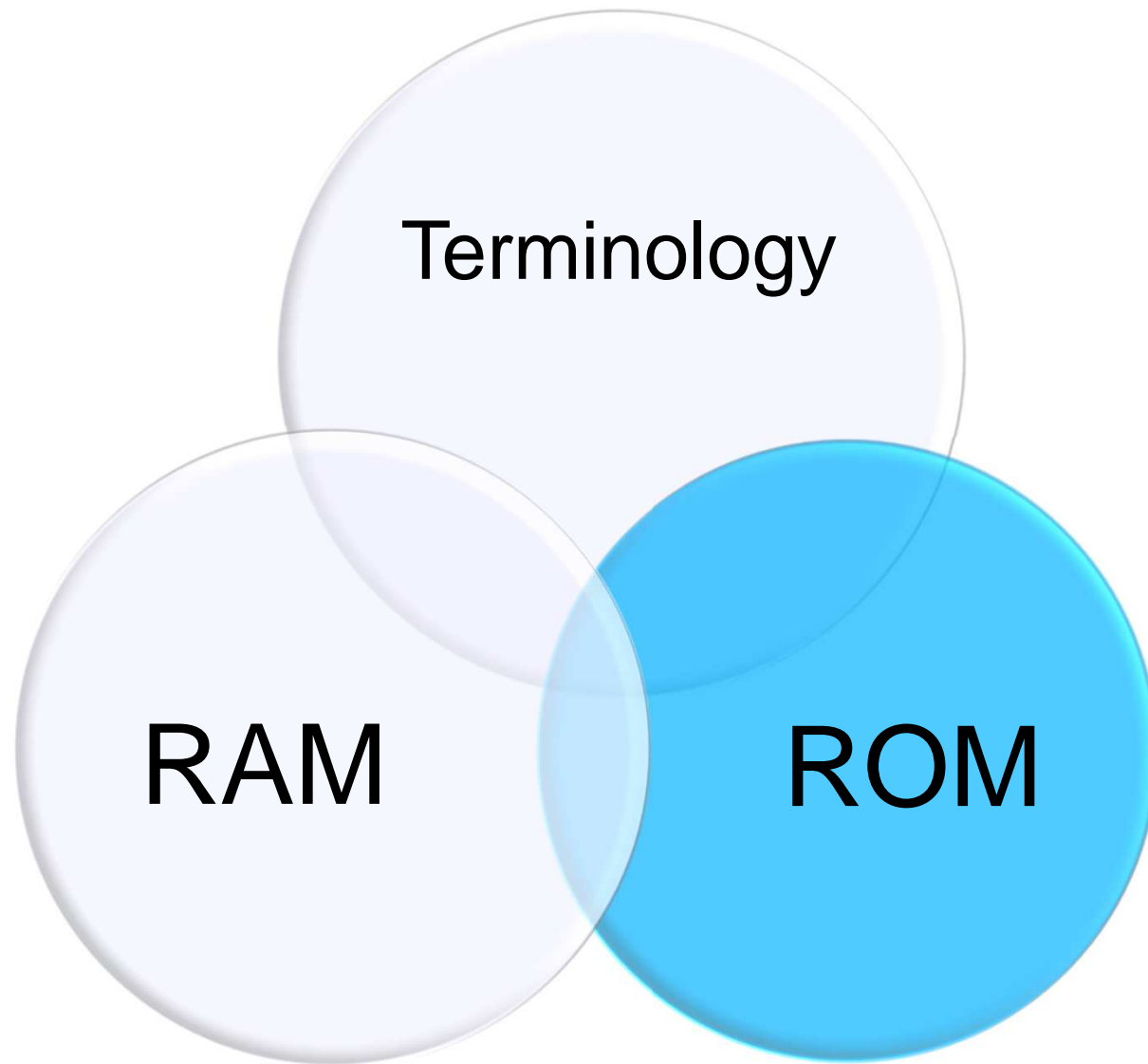


# Write Operation



# Read Operation





# Read-Only Memories

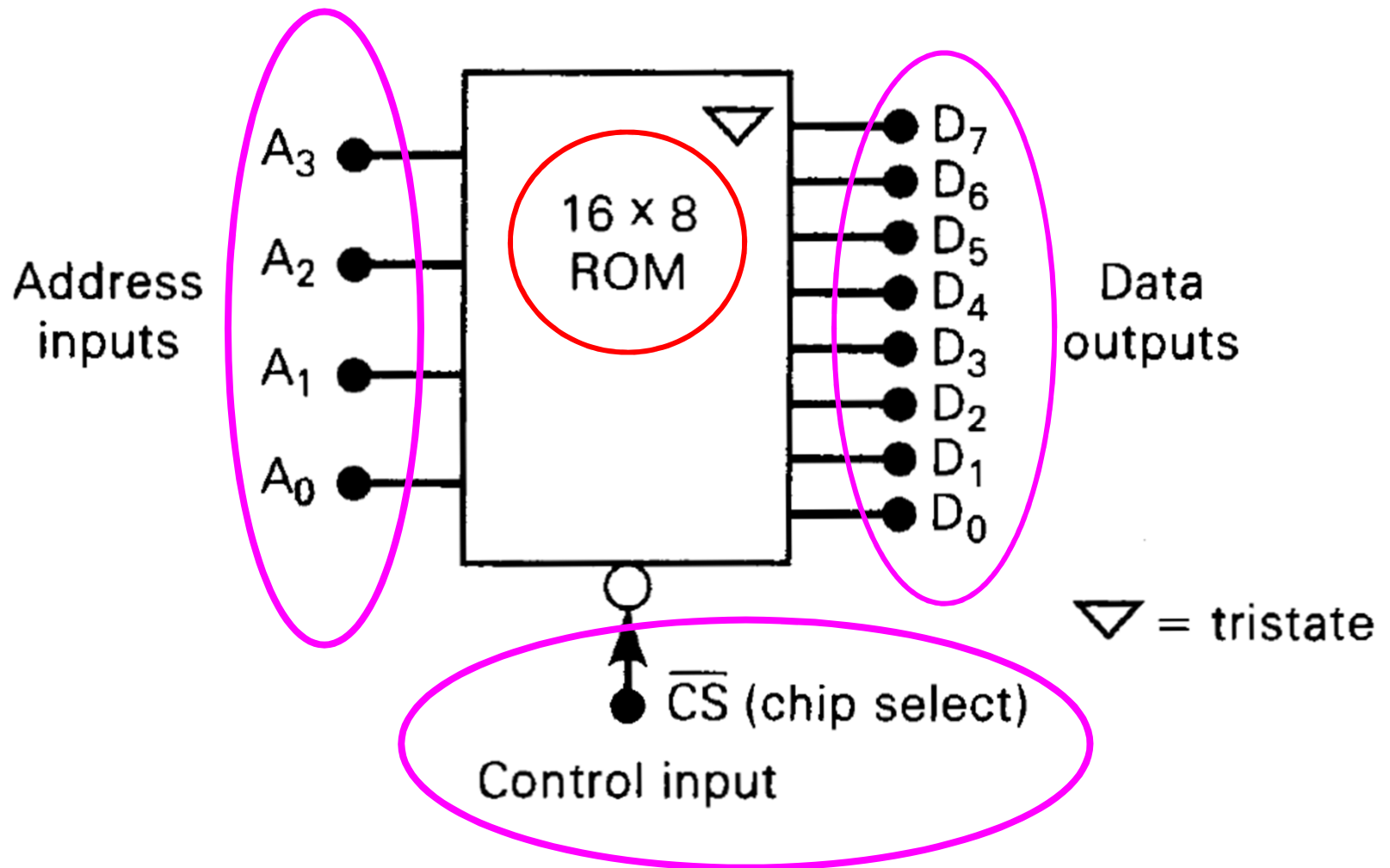
- ROM holds data that either are **permanent** or will **not change frequently**
- During normal operation
  - Data can be read from ROM
  - No new data can be written
- The process of entering data (*programming* or *burning-in* the ROM)
  - Built-in during the manufacturing process
  - Entered electrically
- Data
  - Not to change once **programmed**
  - **Reprogrammed** as often as desired



# Read-Only Memories

- A major use for ROMs is in the storage of programs in microcomputers.
  - ROMs are nonvolatile → the program are not lost when electrical power is turned off
  - When the microcomputer is turned on, it can immediately begin executing the program stored in ROM

# ROM Block Diagram



# Example – ROM's Data

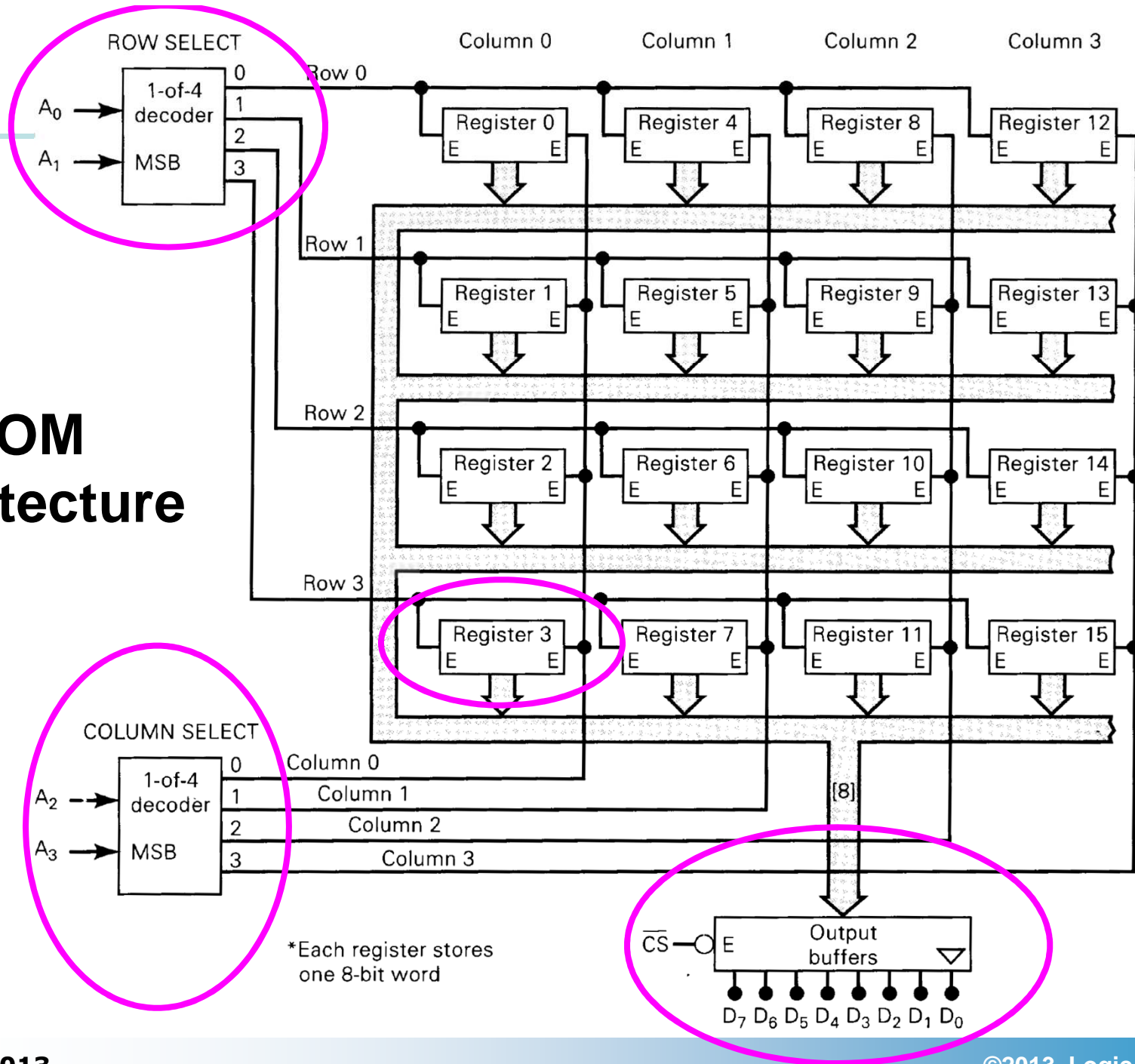
Word	Address				Data							
	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	1	0	1	1	1	1	0
1	0	0	0	1	0	0	1	1	1	0	1	0
2	0	0	1	0	1	0	0	0	0	1	0	1
3	0	0	1	1	1	0	1	0	1	1	1	1
4	0	1	0	0	0	0	0	1	1	0	0	1
5	0	1	0	1	0	1	1	1	1	0	1	1
6	0	1	1	0	0	0	0	0	0	0	0	0
7	0	1	1	1	1	1	1	0	1	1	0	1
8	1	0	0	0	0	0	1	1	1	1	0	0
9	1	0	0	1	1	1	1	1	1	1	1	1
10	1	0	1	0	1	0	1	1	1	0	0	0
11	1	0	1	1	1	1	0	0	0	1	1	1
12	1	1	0	0	0	0	1	0	0	1	1	1
13	1	1	0	1	0	1	1	0	1	0	1	0
14	1	1	1	0	1	1	0	1	0	0	1	0
15	1	1	1	1	0	1	0	1	1	0	1	1

Word	Address				Data							
	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub> –D <sub>0</sub>							
0		0			DE							
1		1			3A							
2		2			85							
3		3			AF							
4		4			19							
5		5			7B							
6		6			00							
7		7			ED							
8		8			3C							
9		9			FF							
10		A			B8							
11		B			C7							
12		C			27							
13		D			6A							
14		E			D2							
15		F			5B							

# Read Operation

- In order to read a data word from ROM, we need to do two things
  - Apply the appropriate address inputs
  - Activate the control inputs

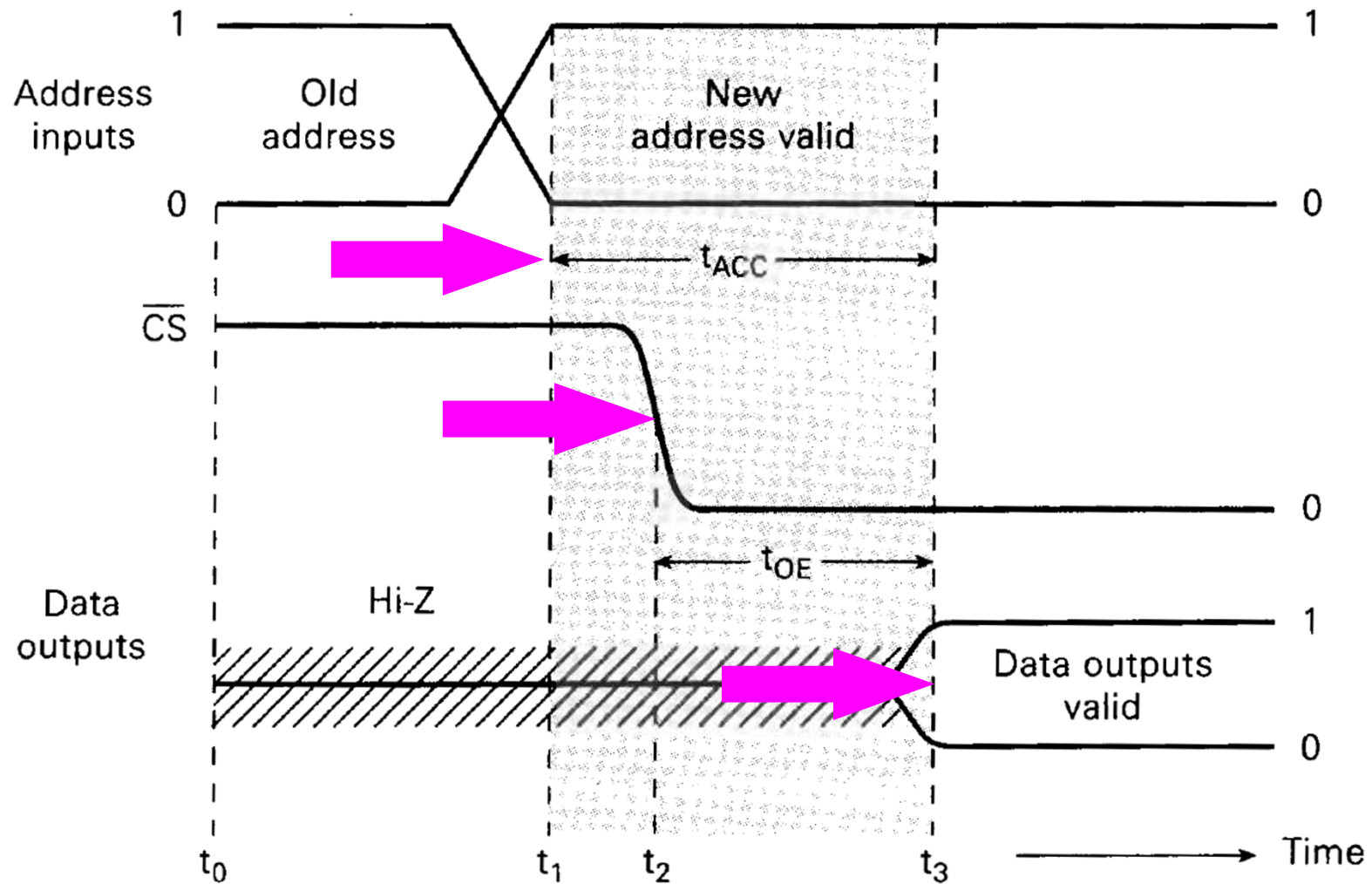
# ROM Architecture



# ROM Architecture

- Register Array
  - Store data that have been programmed into the ROM
  - Contain a number of memory cells equal to the word size
  - Data outputs are connected to an internal data bus
  - 2 enable inputs (E)
- Address Decoders
  - Determine which register in the array will be enabled
  - Row-select lines, column-select lines
- Output Buffers
  - Pass data from enabled register to the external data outputs

# ROM Timing



# Type of ROMs

- Mask-Programmed ROM (MROM)
  - Storage locations are written into (programmed) by the manufacturer according to the customer's specification
  - A *mask* is used to control the electrical interconnections on the chip
    - A special mask is required for each different set of information to be stored in the ROM
    - These masks are expensive
  - **Disadvantage:** cannot be reprogrammed in the event of a design change requiring a modification of the stored data
  - The most economical approach when a large quantity of identically programmed ROMs are needed

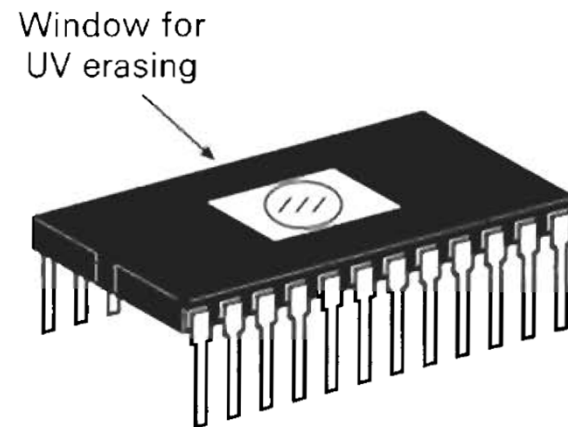


# Type of ROMs

- Programmable ROM (PROM)
  - Use in lower-volume applications, manufacturers have developed **fusible-link PROMs** that are user-programmable
  - *One-time programmable* (OTP ROM), **cannot be erased and reprogrammed**
- Erasable Programmable ROM (EPROM)
  - Programmed by user, erased and reprogrammed as often as desired
  - The programming process is usually performed by a **special programming circuit** that is separate from the circuit in which the EPROM will eventually be working
  - Erased by exposing it to *ultraviolet* (UV)

# Type of ROMs

- Erasable Programmable ROM (EPROM)
  - Disadvantages
    - They must be removed from their circuit to be erased and reprogrammed
    - The erase operation **erases the entire chip** - there is no way to select only certain addresses to be erased
    - The erase and reprogramming process can typically take 20 minutes or more.



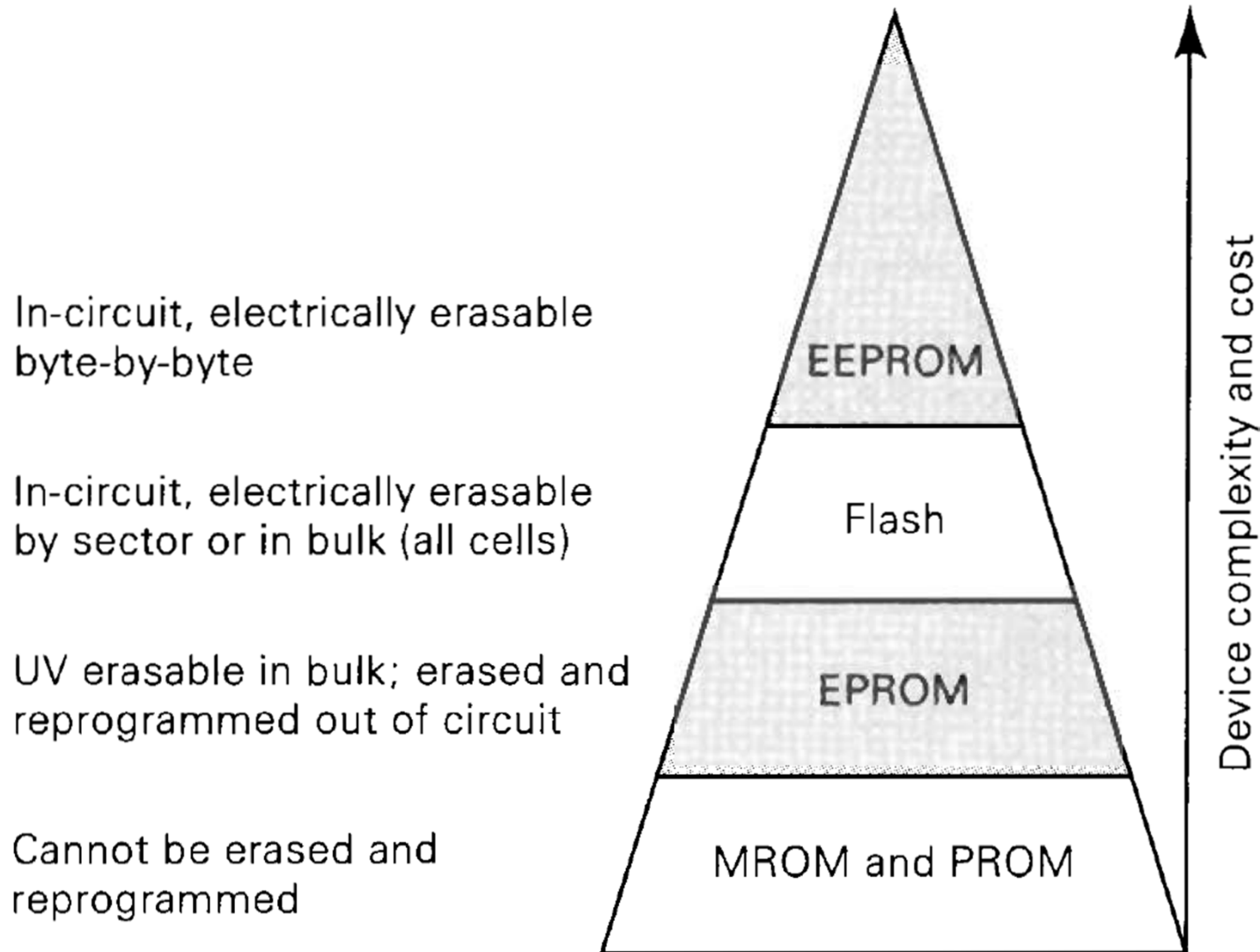
# Type of ROMs

- **Electrical Erasable PROM (EEPROM)**
  - The erasing and programming of an EEPROM can be done in circuit
  - The ability to erase and rewrite individual bytes
  - During a write operation, internal circuitry automatically erases all of the cells at an address location prior to writing in the new data
- **CD ROM – Compact Disk**
  - To store data on the disks, a very intense laser beam is focused on a very small point on the disk
  - Digital data (1 or 0) are stored on the disk one bit at a time by burning or not burning a pit into the reflective coating

# Flash Memory

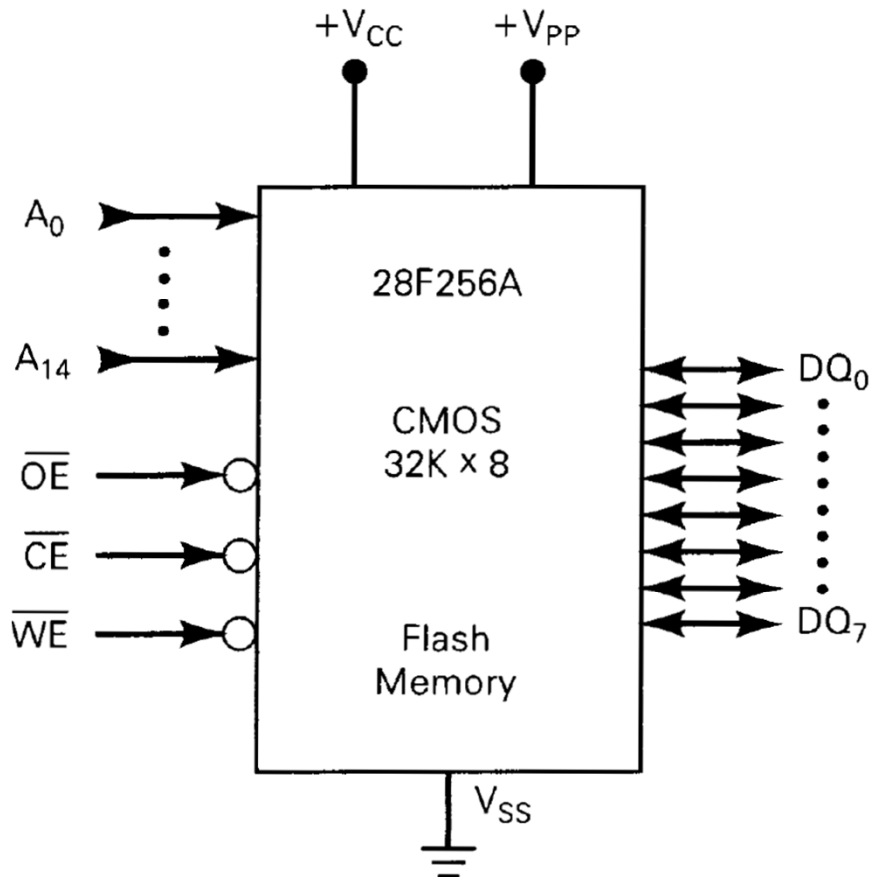
- *Challenge*: fabricate a nonvolatile memory with the EEPROM's in-circuit electrical erasability, but with high density and low cost per bits (EPROM)
- Solution: **Flash memory**
  - Flash memory cell is like the EPROM cell, being only slightly larger
  - Allows electrical erasability but can be built with much higher densities than EEPROMs
  - The cost is considerably less than for EEPROM
  - Rapid erase and write times
  - Use *bulk erase* operation in which all cells on the chip are erase simultaneously, requires hundreds of milliseconds compares to 20 minutes for UV EPROMs

# Trade-Offs for Nonvolatile Memory



# Example

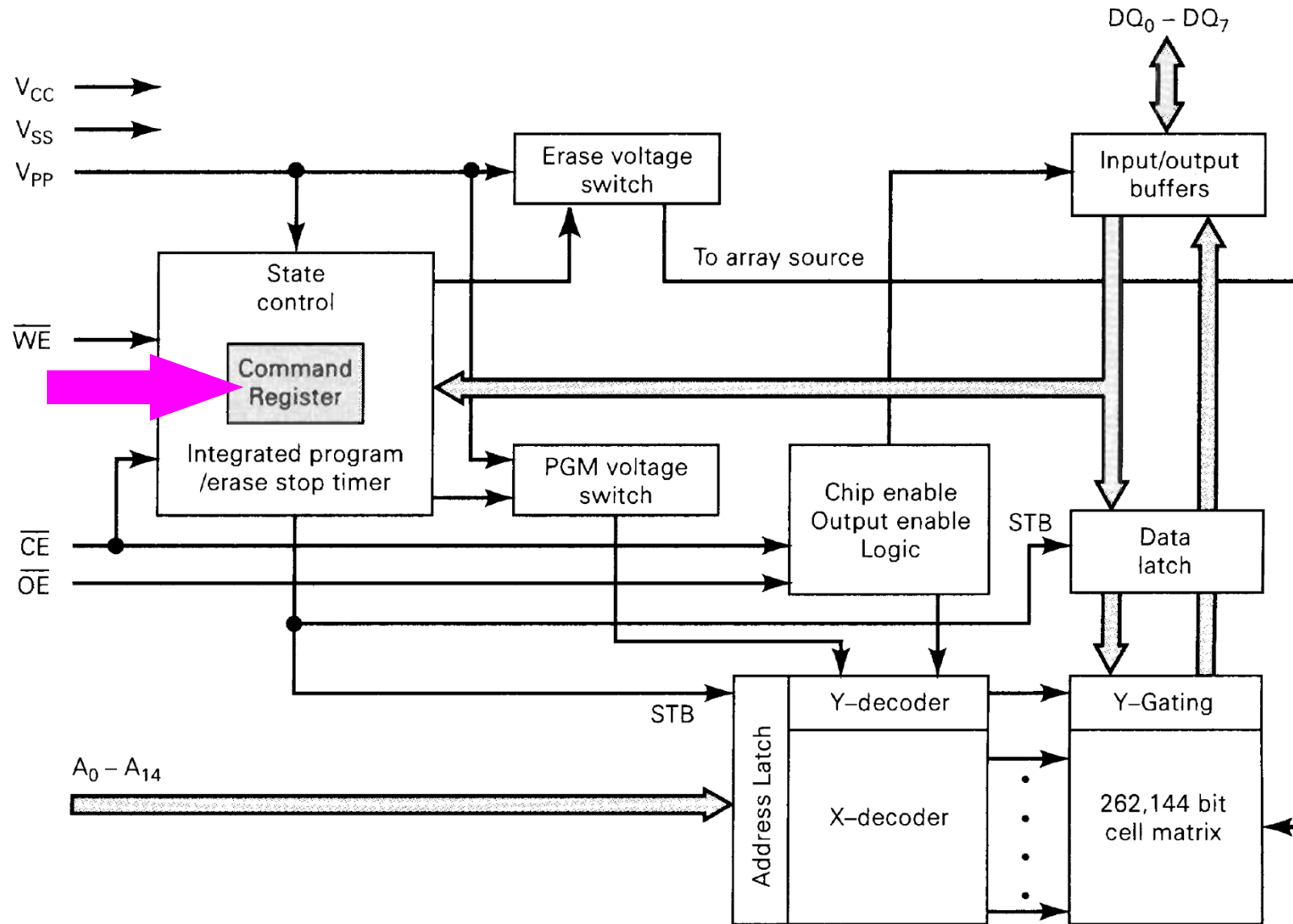
- The 28F256A CMOS Flash Memory IC



Mode	Inputs			Data pins
	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	
READ	LOW	LOW	HIGH	$DATA_{OUT}$
STANDBY	HIGH	X	X	High Z
WRITE*	LOW	HIGH	LOW	$DATA_{IN}$

\*Note: If  $V_{PP} \leq 6.5V$  a write operation cannot be performed

# Functional Diagram



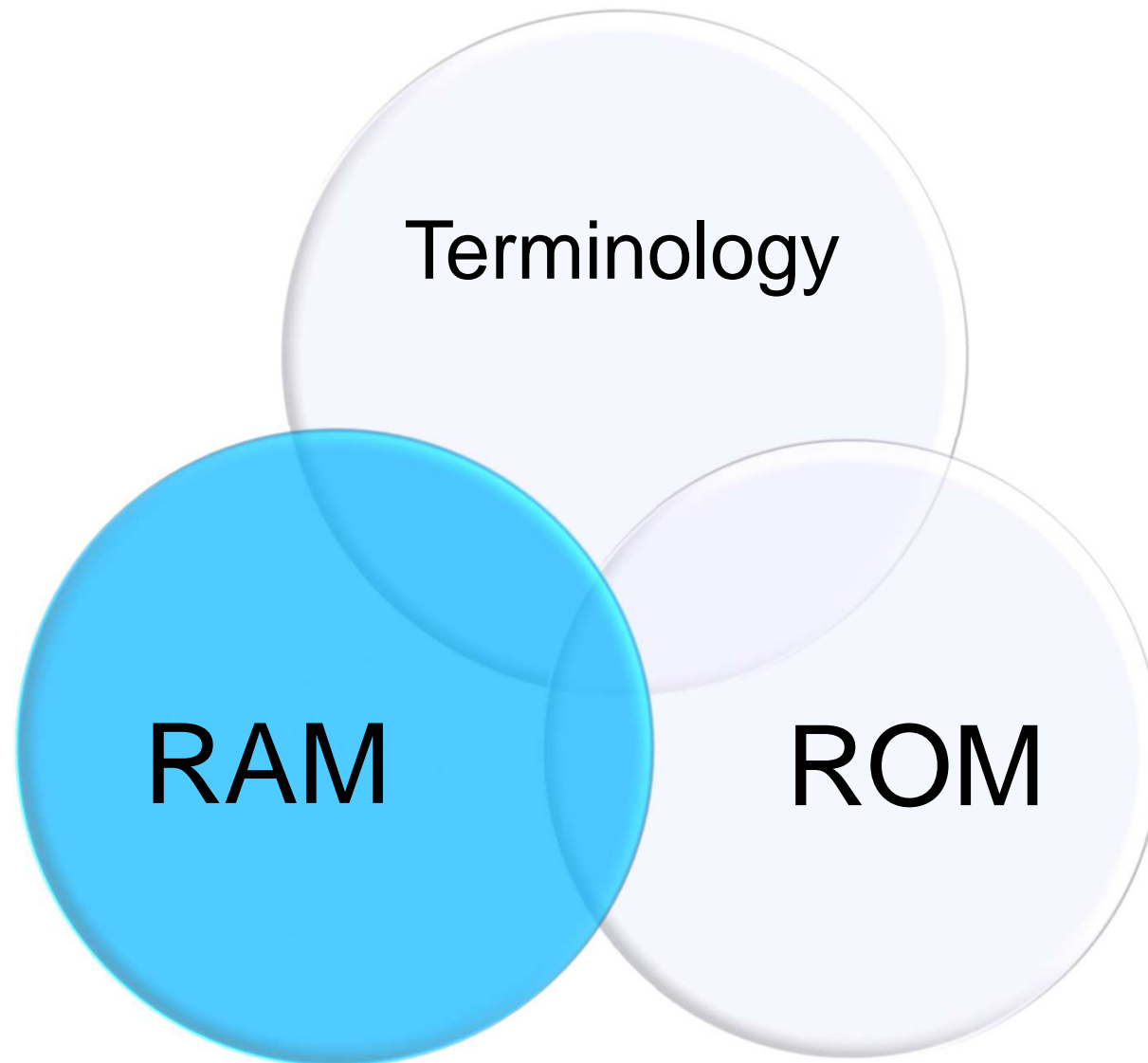
# ROM - Operations

- Read Command
- Set-Up Erase/Erase Commands
- Erase-Verify Command
- Set-Up Program/Program Commands
- Program-Verify Command



# ROM Applications

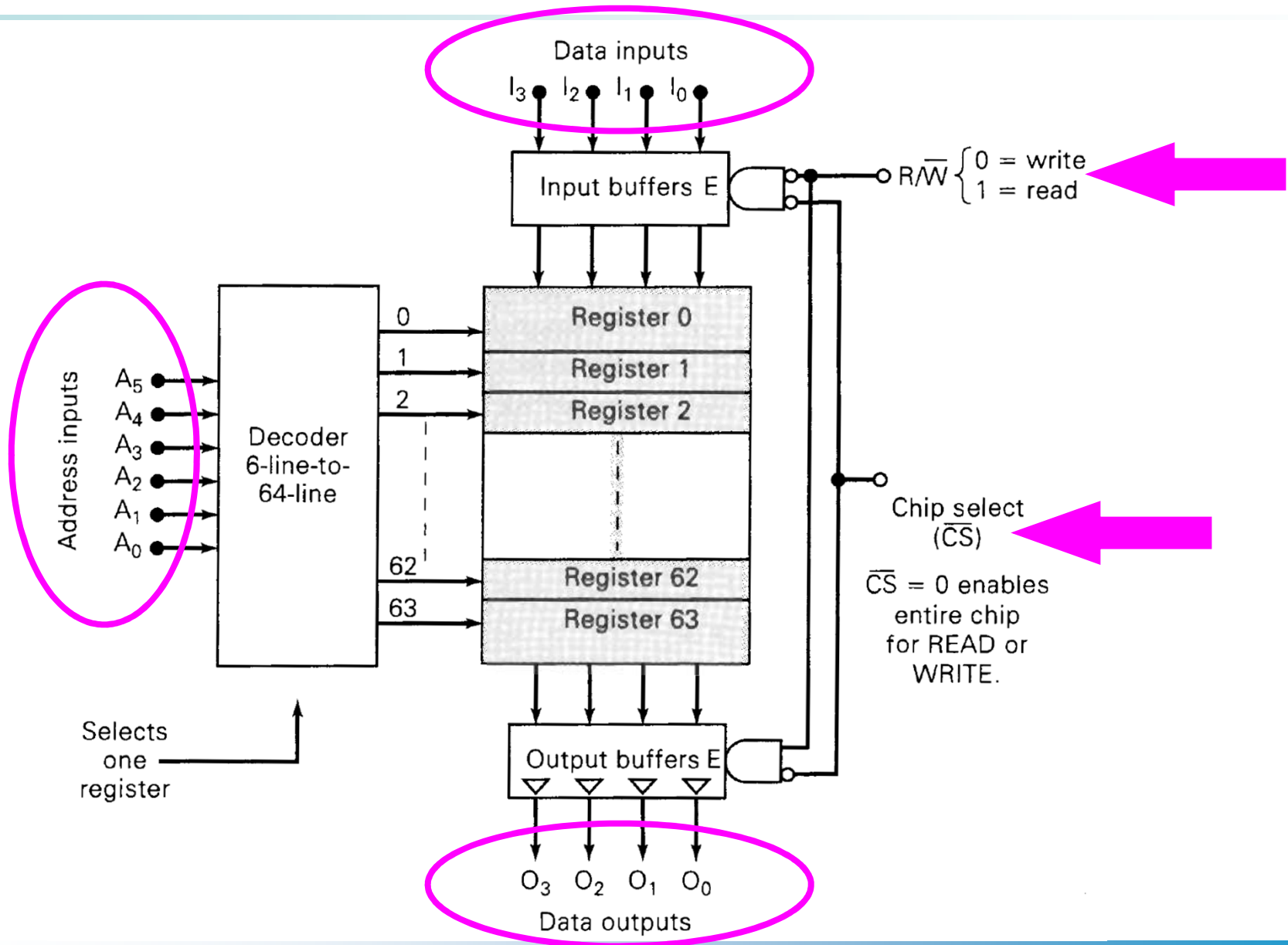
- Firmware
  - The storage of data and program codes that must be available on power-up
- Bootstrap Memory
  - Initialize the system hardware, loads the OS
- Data Tables
  - Store tables of data that do not change
- Data Converter
- Function Generator
- Auxiliary Storage



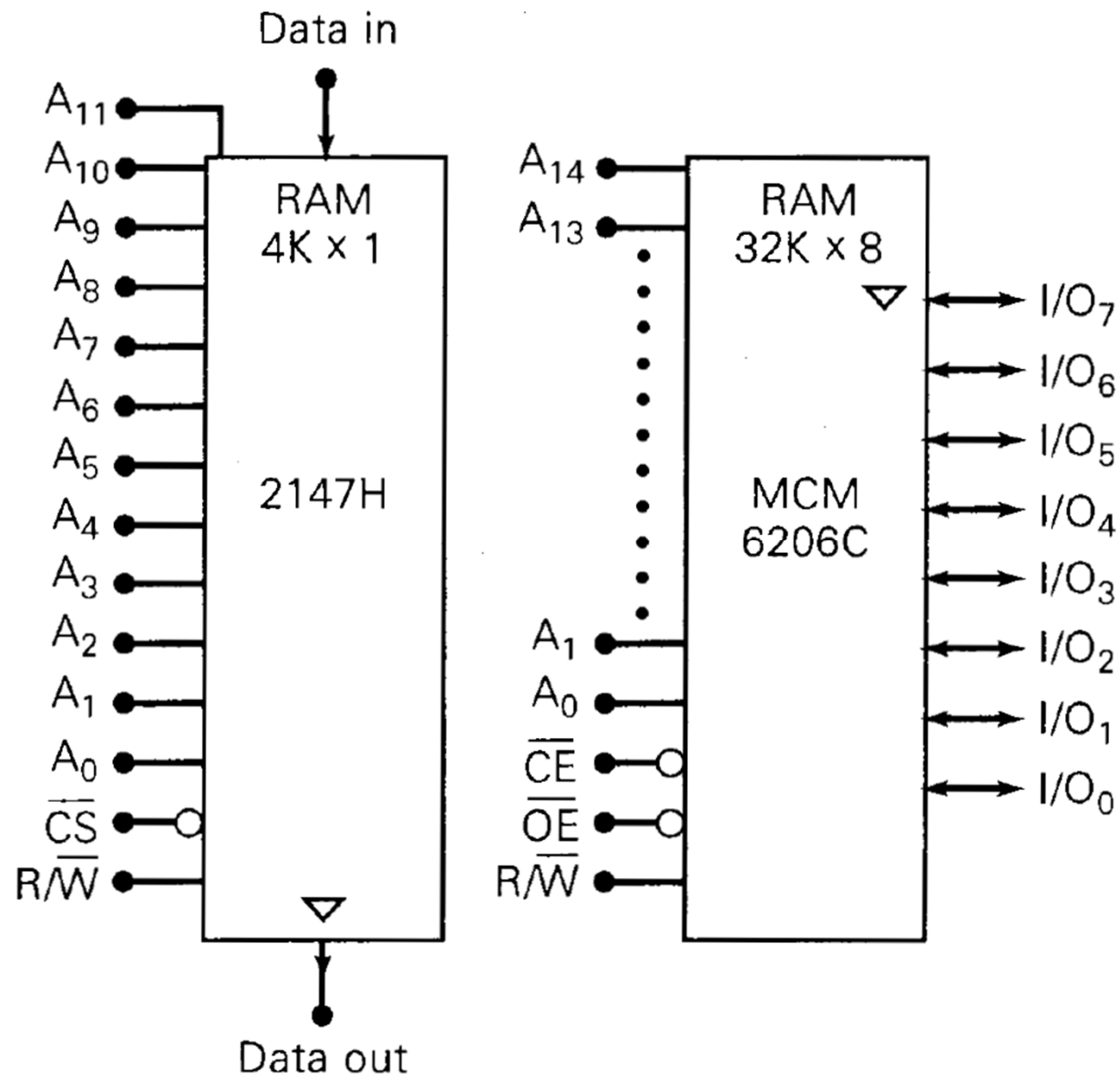
# Semiconductor RAM

- RAM: random-access memory
- Any memory address location is as easily accessible as any other
- Temporary storage of programs and data
- Fast read and write cycle times
- Disadvantage:
  - Volatile memory: lose all stored information if power is interrupted or turned off

# RAM Architecture

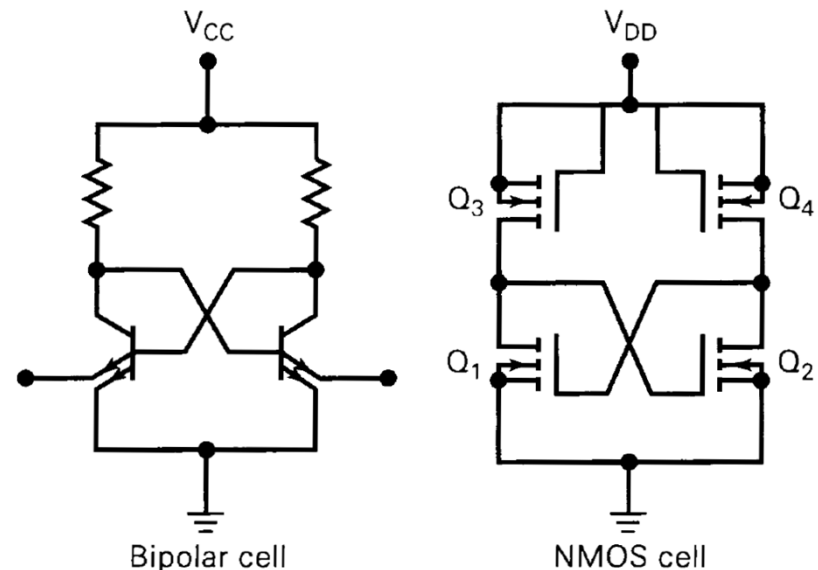


# Examples



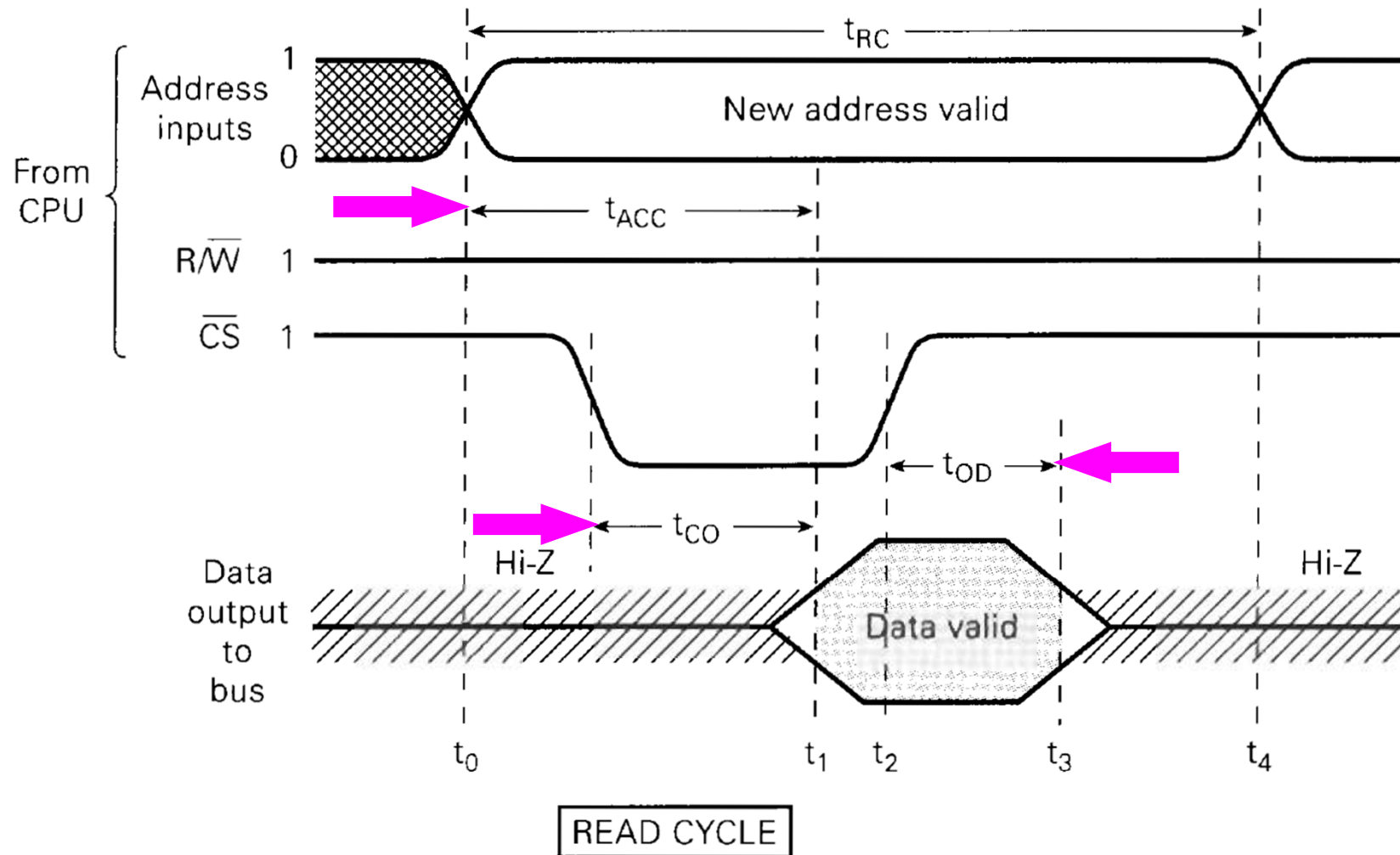
# Static RAM

- Store data as long as power is applied to the chip
- Static RAM memory cells are essentially flip-flops that will stay in a given state (store a bit) indefinitely
- SRAMs are available in bipolar, MOS, and BiCMOS technologies
  - the majority of applications use NMOS or CMOS RAMs
- Bipolar
  - Advantage in speed
  - More chip area
- MOS
  - Greater capacities
  - Low power consumption



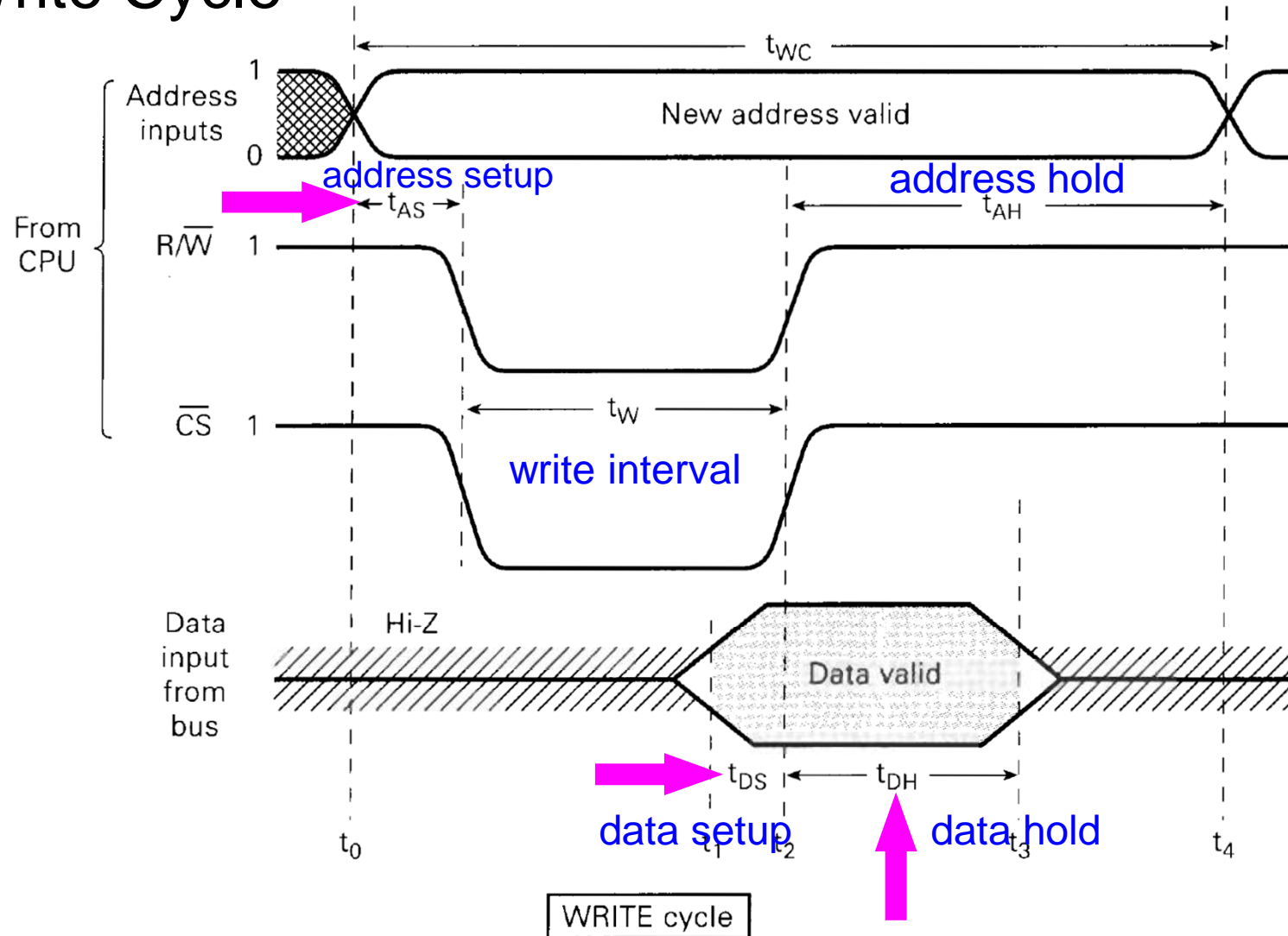
# Static-RAM Timing

- Read Cycle



# Static-RAM Timing

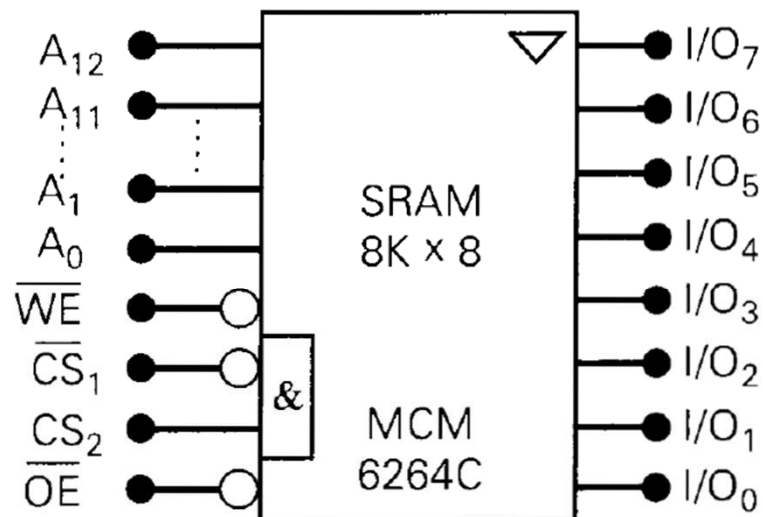
- Write Cycle





# Actual SRAM Chip

- 2764
- 6264
- 2864
- 27256



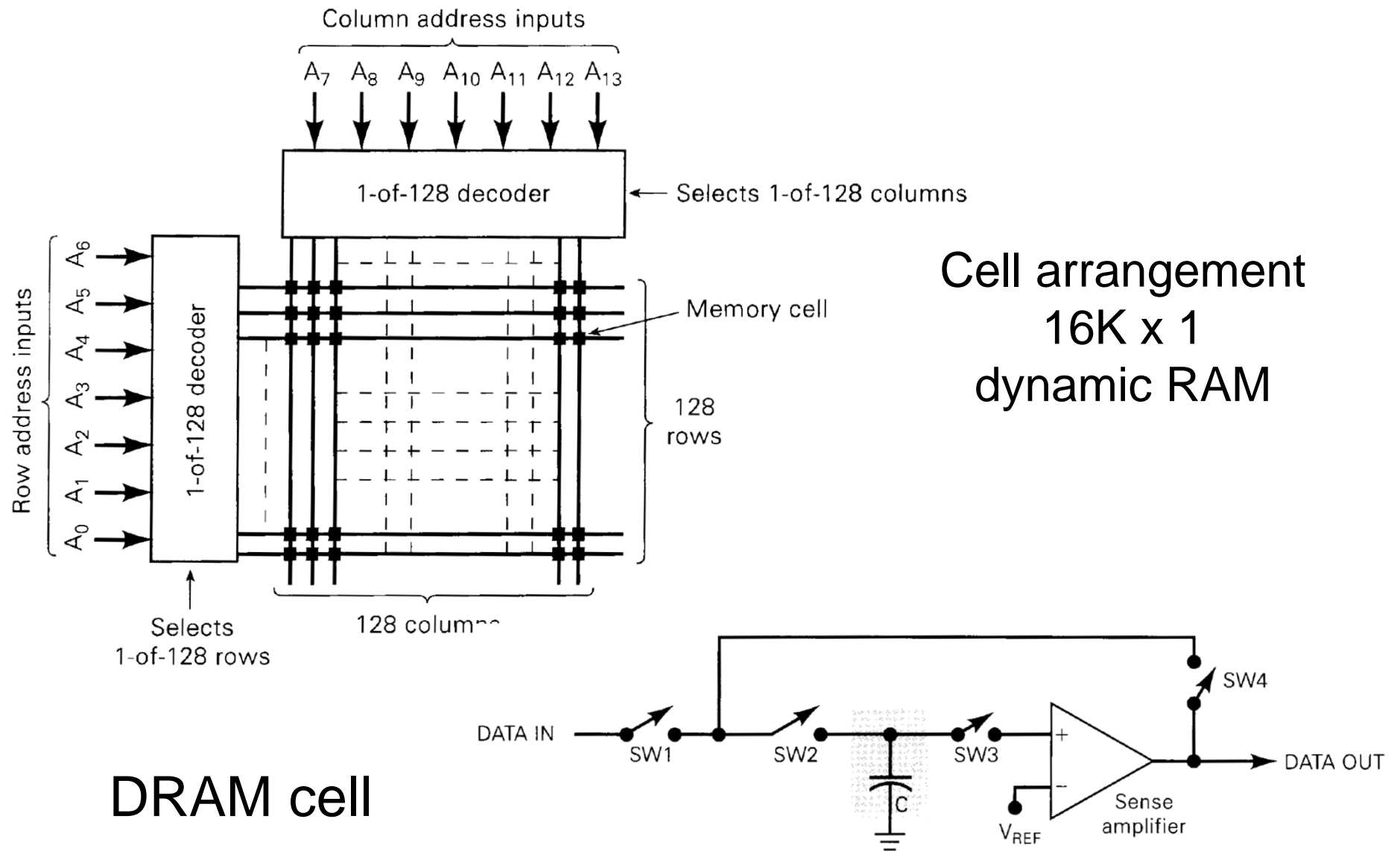
Mode	Inputs				I/O pins
	$\overline{WE}$	$\overline{CS}_1$	$CS_2$	$\overline{OE}$	
READ	1	0	1	0	DATA <sub>OUT</sub>
WRITE	0	0	1	X	DATA <sub>IN</sub>
Output disable	1	X	X	1	High Z
Not selected (power down)	X	1	X	X	High Z

X = don't care

# Dynamic RAM (DRAM)

- Fabricated using MOS technology, high capacity, low power, moderate operating speed
- Store 1s and 0s as charges on a small MOS capacitor (a few pF)
- Require periodic recharging of the memory cells – *refreshing*
- DRAMs typically have four times the density of SRAMs

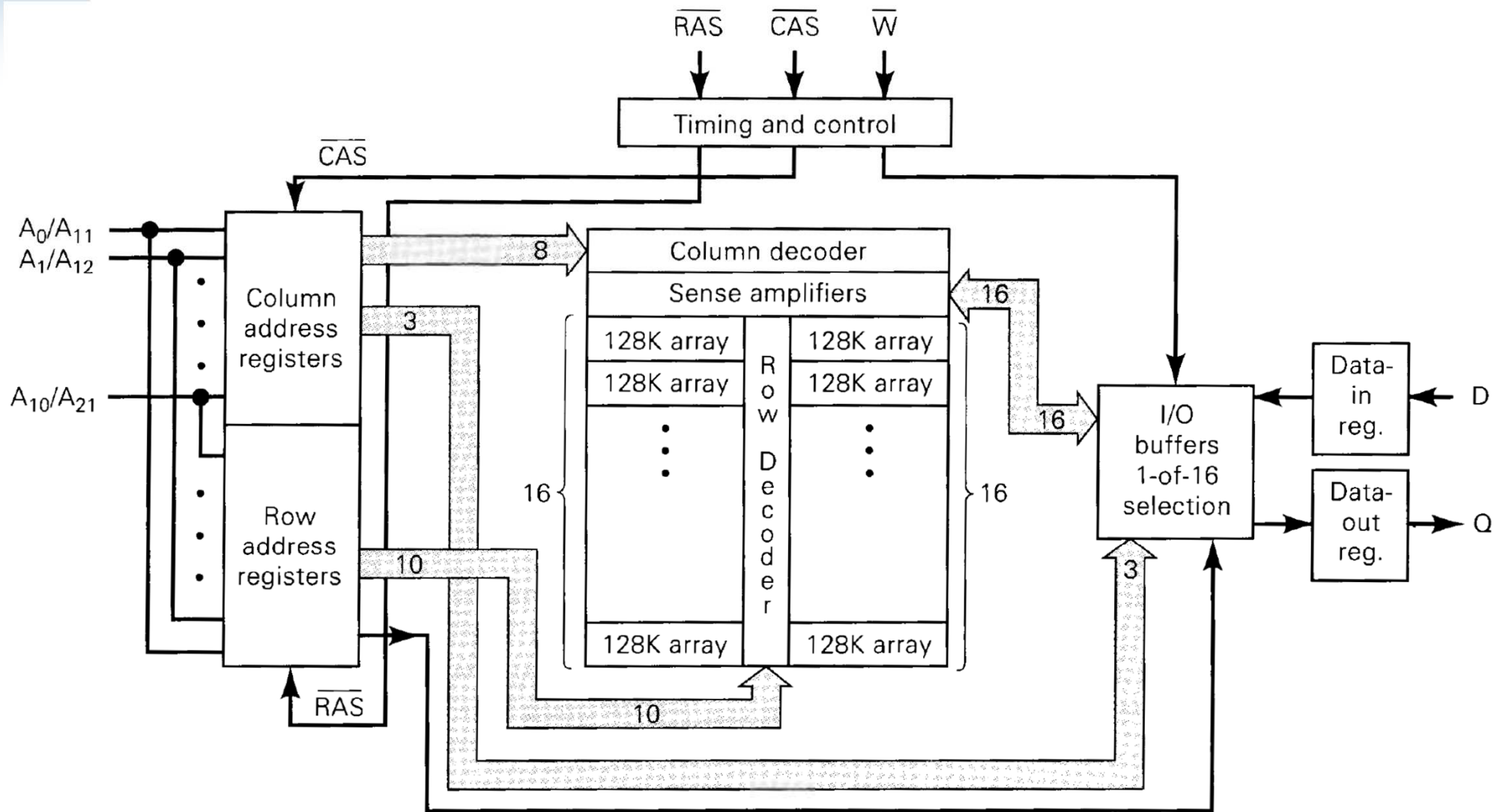
# DRAM Structure and Operation



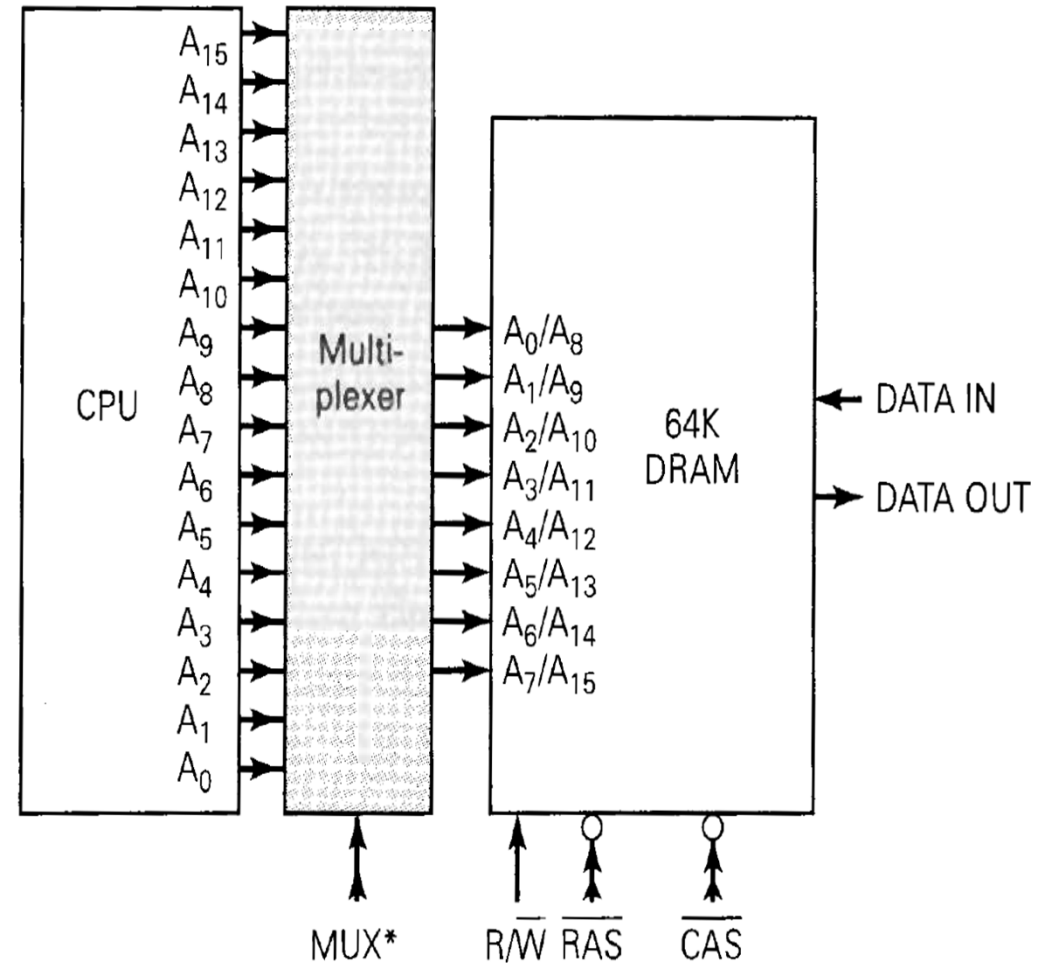
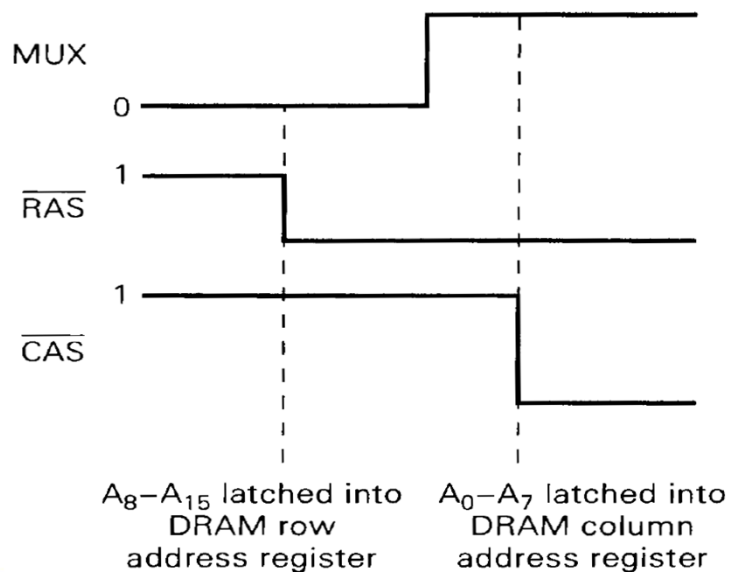
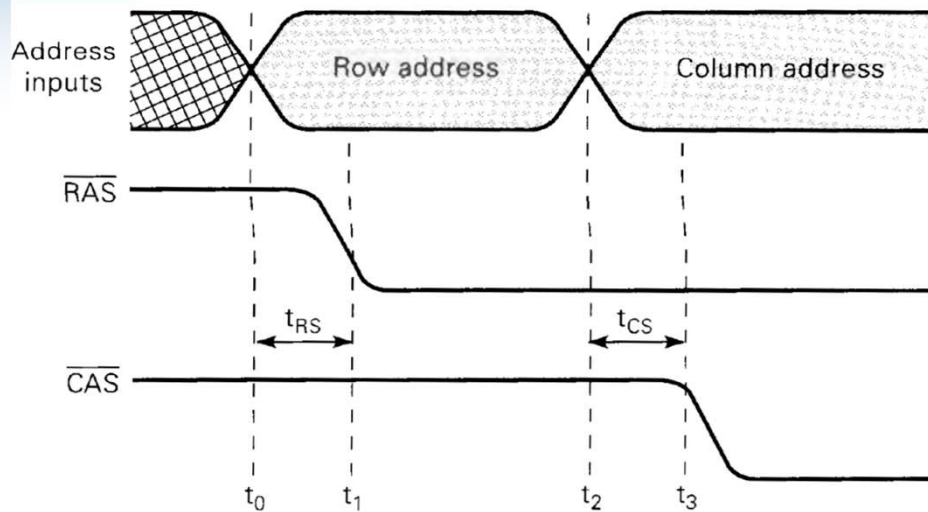
# Address Multiplexing

- Each address input pin can accommodate two different address bits → saving in pin will decrease in the size of the IC packages
- Address lines go to both the row and column registers
  - The row register stores the upper half
  - The column register stores the lower half
- Two very important strobe inputs control when the address information is latched
  - *Row address strobe* ( $\overline{RAS}$ )
  - *Column address strobe* ( $\overline{CAS}$ )

# Address Multiplexing

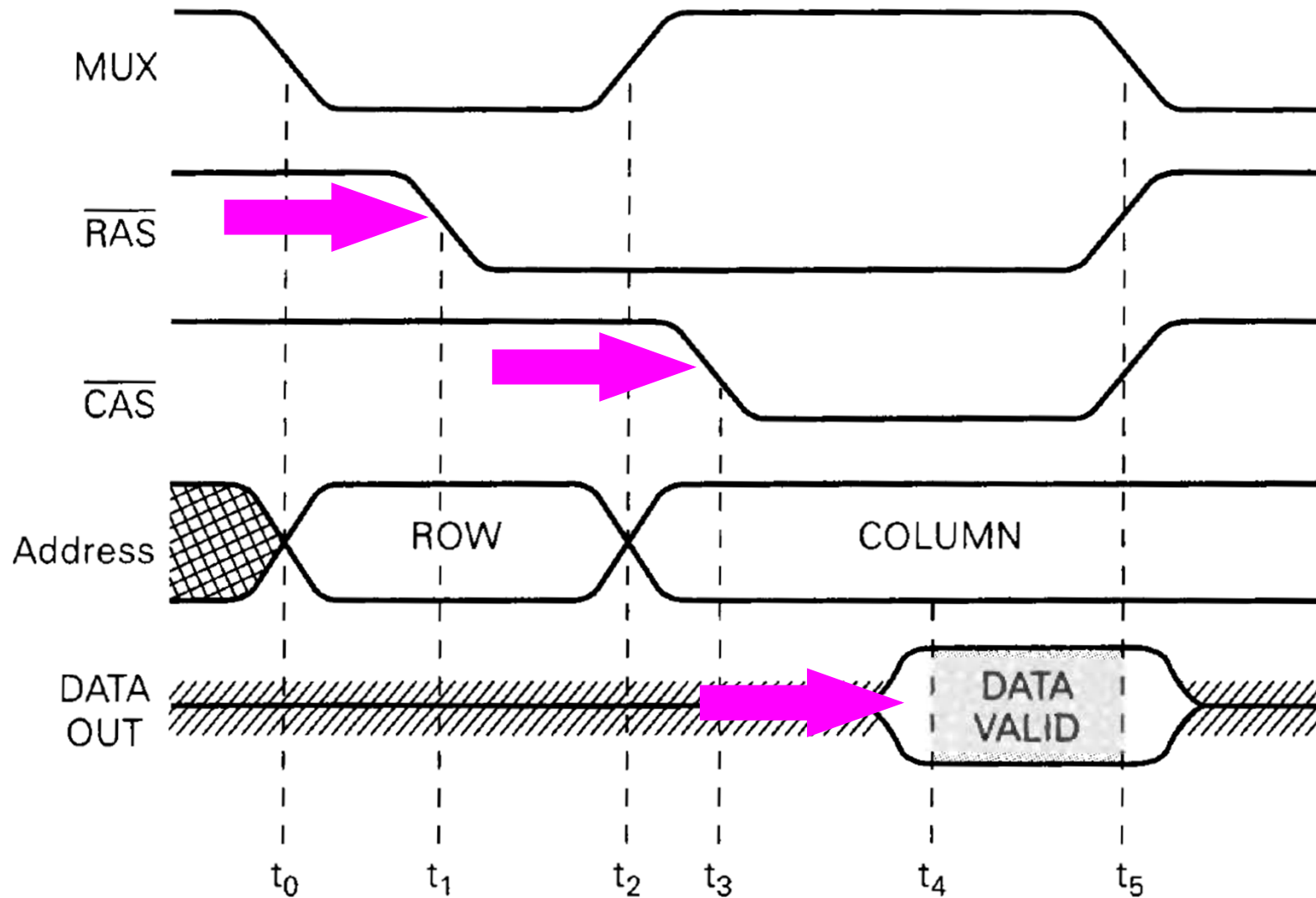


# Address Multiplexing

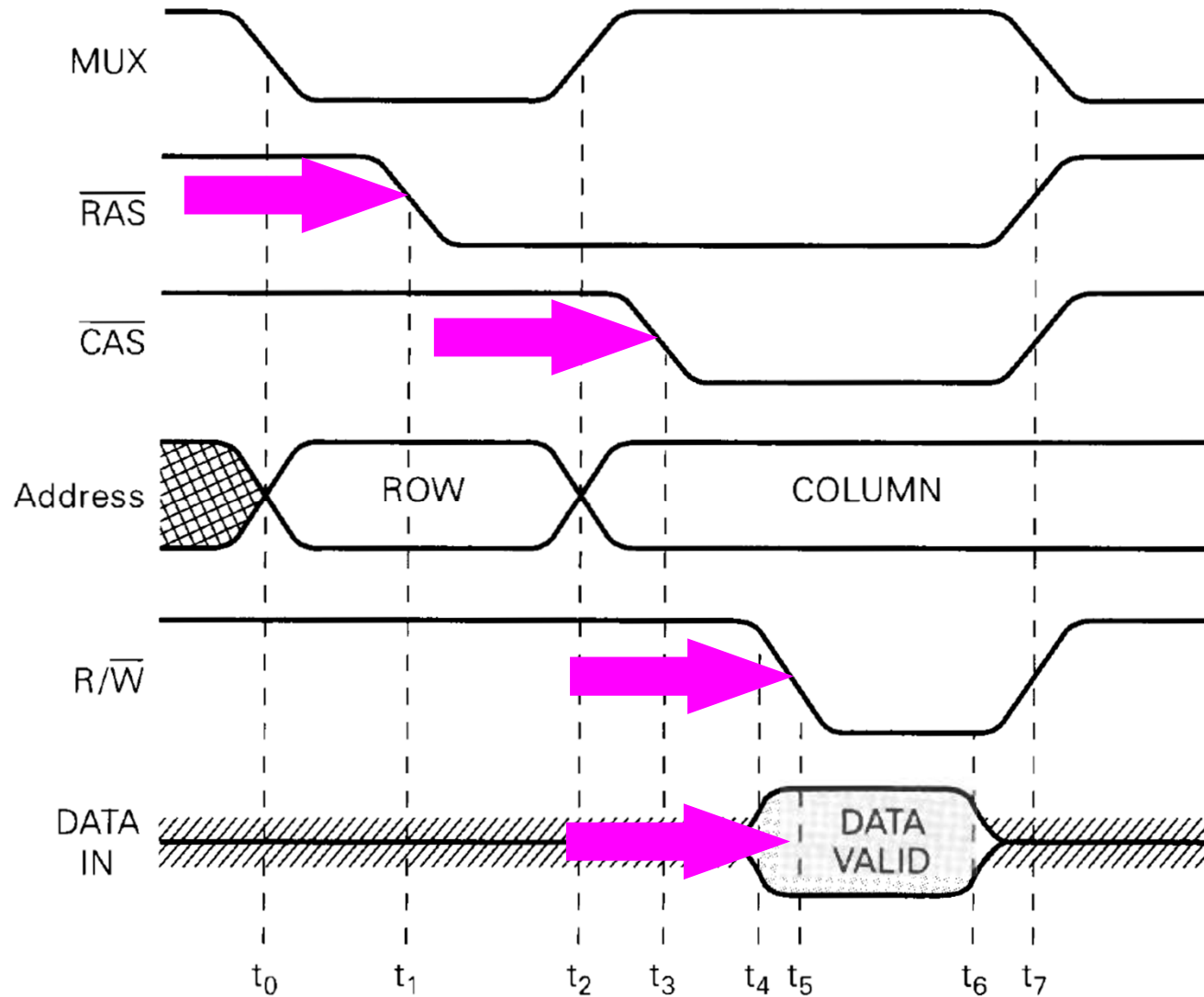


- MUX = 0 transmits CPU address  $A_8 - A_{15}$  to DRAM. MUX = 1 transmits  $A_0 - A_7$  to DRAM.

# DRAM Read Cycle



# DRAM Write Cycle



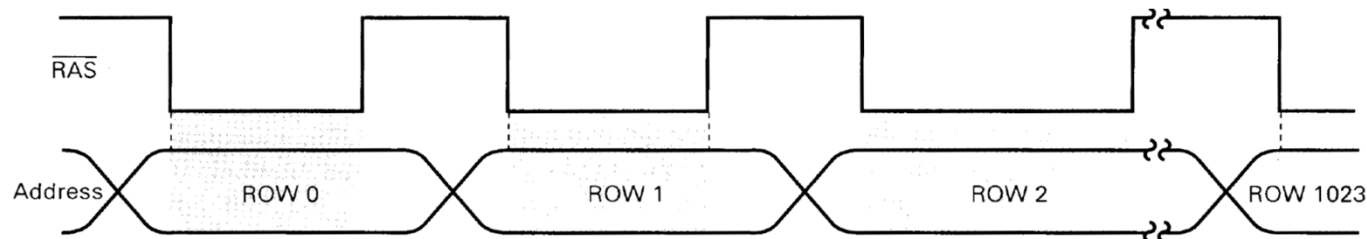


# DRAM Refreshing

- Whenever a *read operation* is performed on a cell, all of the cells in the row will be refreshed
- Two refresh modes
  - **Burst refresh:** the normal memory operation is suspended, and each row of the DRAM is refreshed in succession until all rows have been refreshed
  - **Distributed refresh:** the row refreshing is interspersed with the normal operations of memory

# DRAM Refreshing

- The most universal method for refreshing a DRAM is the  *$\overline{RAS}$ -only refresh*
  - Strobing in a row address with  $\overline{RAS}$  while  $\overline{CAS}$  and  $R/W$  remain HIGH
  - Refresh counter is used to supply 10-bit row addresses to the DRAM address inputs
  - Refresh counter address must be multiplexed with the CPU addresses
  - *Dynamic RAM (DRAM) controller* will perform address multiplexing and refresh count sequence generation

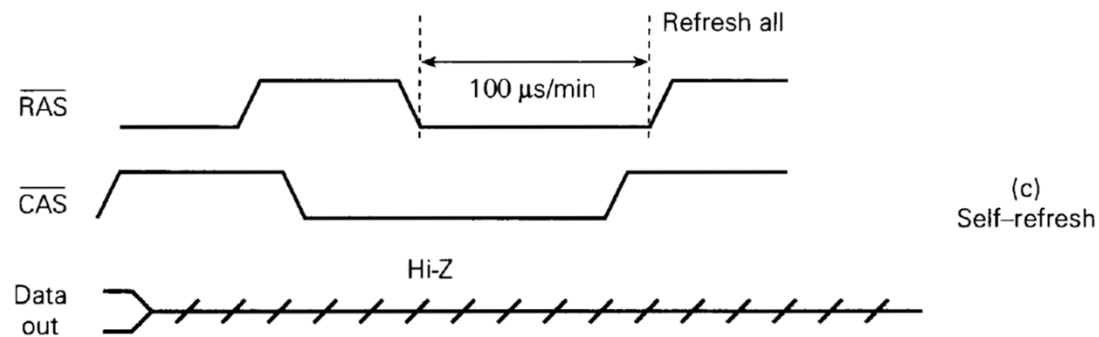
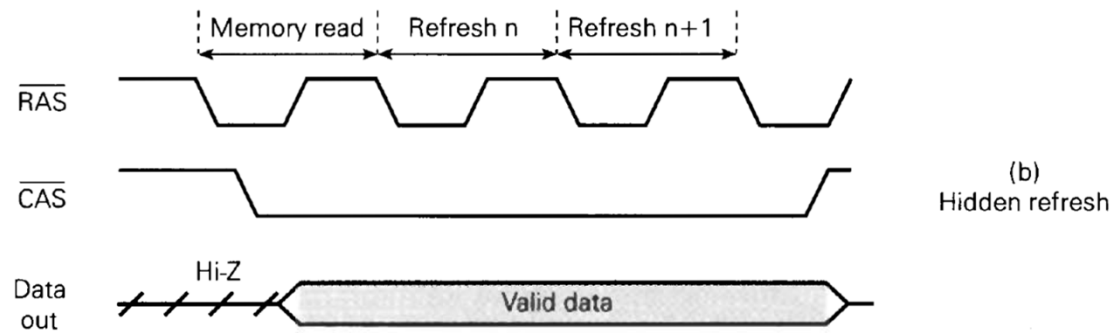
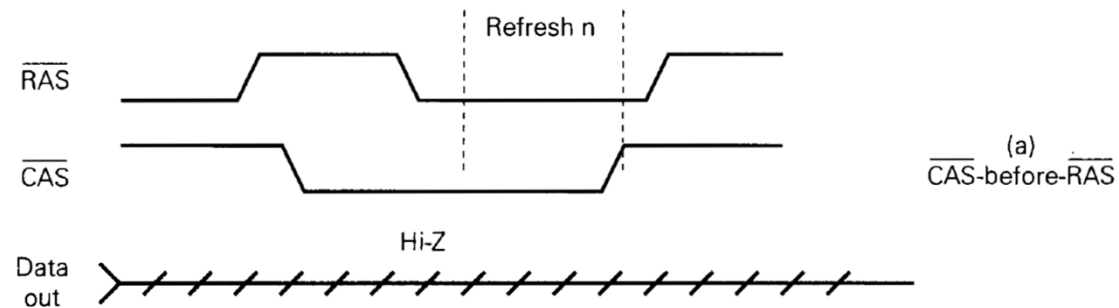


\*  $R/W$  and  $\overline{CAS}$  lines held HIGH

# Refreshing Method

- *CAS-before-RAS refresh*:  $\overline{\text{CAS}}$  signal is driven LOW first and is held LOW until after  $\overline{\text{RAS}}$  goes LOW
- *Hidden refresh*: allow a row to be refreshed while holding data on the output
  - Holding  $\overline{\text{CAS}}$  LOW after a read cycle and the pulsing  $\overline{\text{RAS}}$
- *Self-refresh*: fully automate the process
  - Forcing  $\overline{\text{CAS}}$  LOW before  $\overline{\text{RAS}}$  and then holding them both LOW for at least 100  $\mu\text{s}$

# Refreshing Method

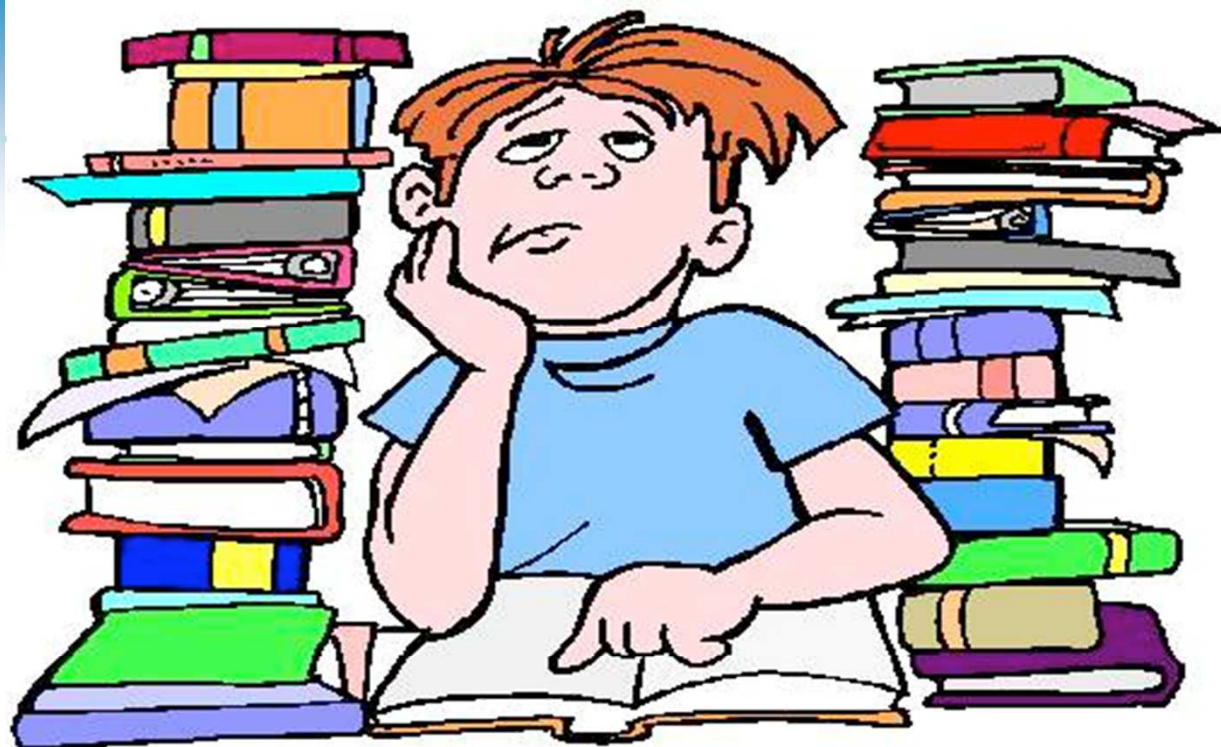


# DRAM Technology

- Single-in-line memory module (SIMM)
- Dual-in-line memory module (DIMM)
- Rambus In-line memory module (RIMM)
- FPM RAM
- EDO DRAM
- SDRAM
- DDRSDRAM
- SLDRAM
- DRDRAM

# Summarize

- 
- ✓ Memory Terminology
  - ✓ ROM operation
  - ✓ RAM operation



# Reference

- Chapter 11, Digital System – Principles and Applications, Ronald J.Tocci, Neal S. Widmer