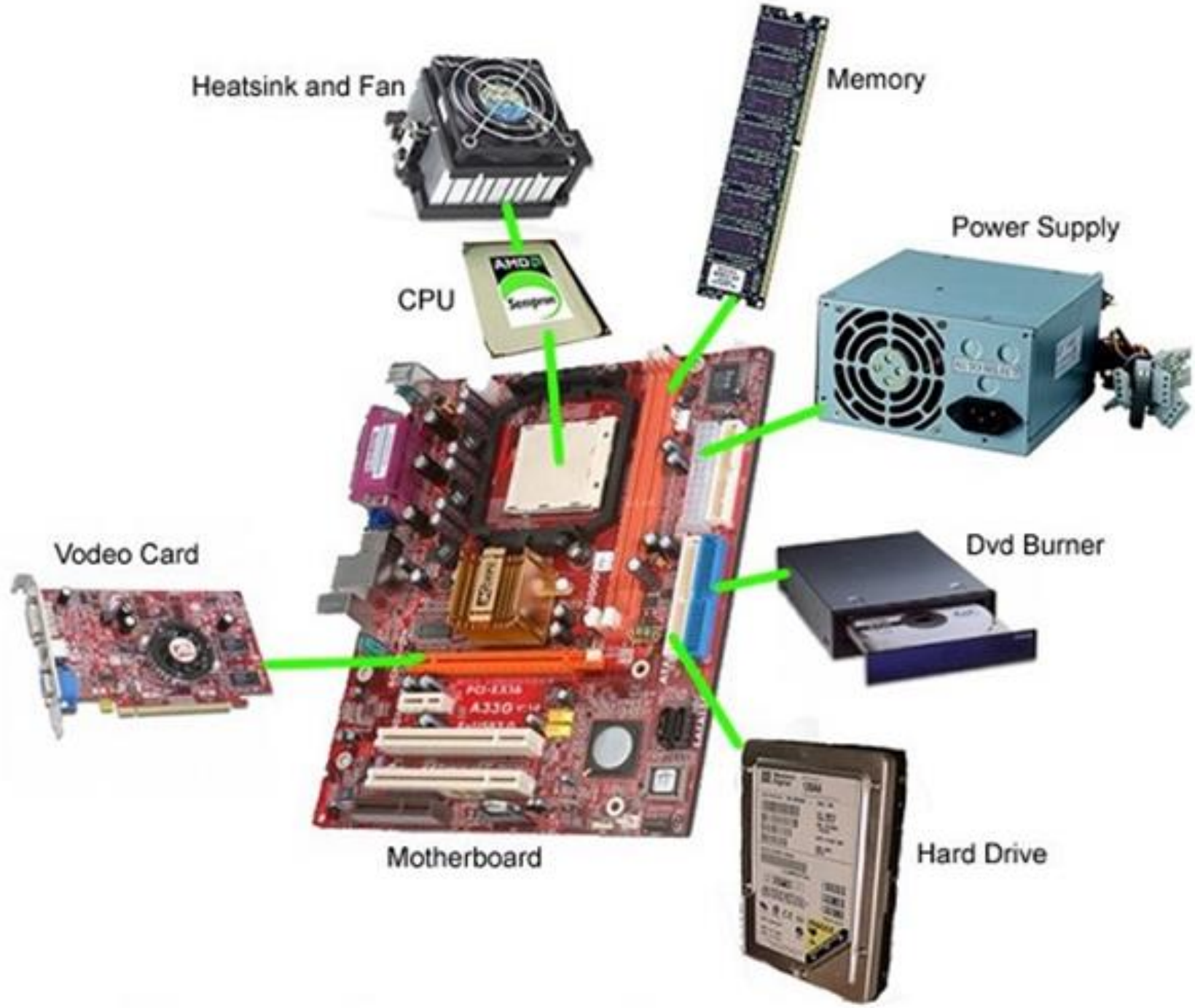
Summer 2025-CSE332

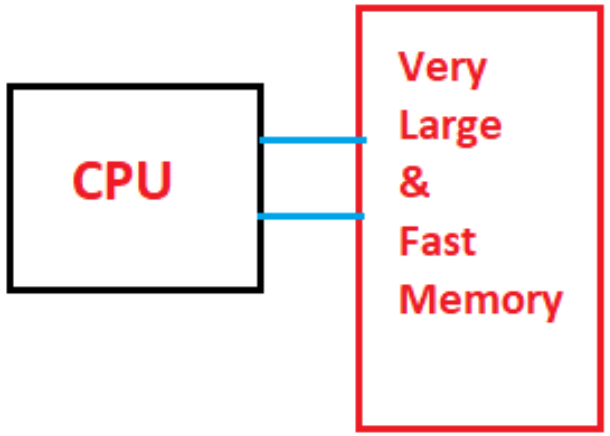
Computer Organization and Architecture

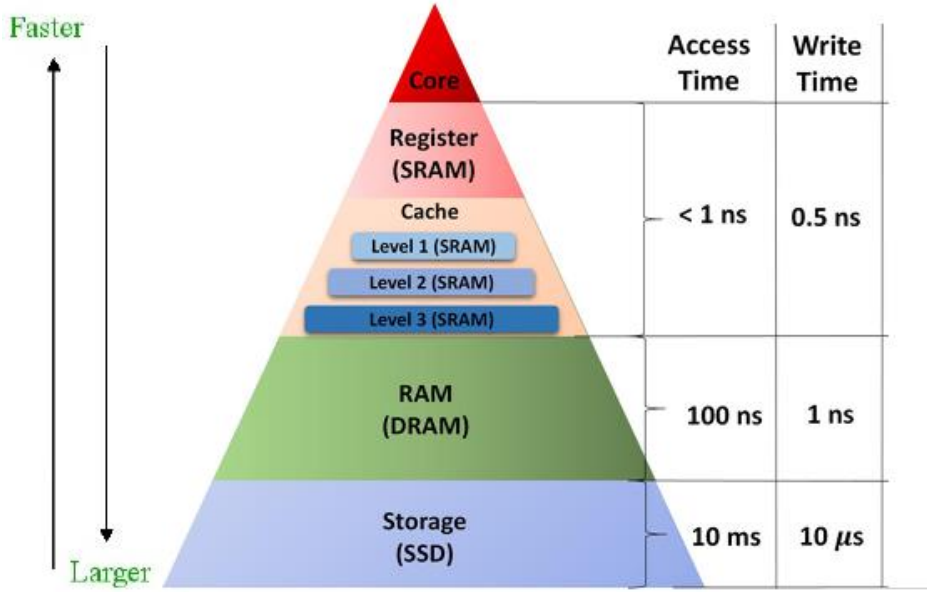

Heatsink and Fan
Memory
CPU
Power Supply
Vodeo Card
Dvd Burner
Motherboard
Hard Drive

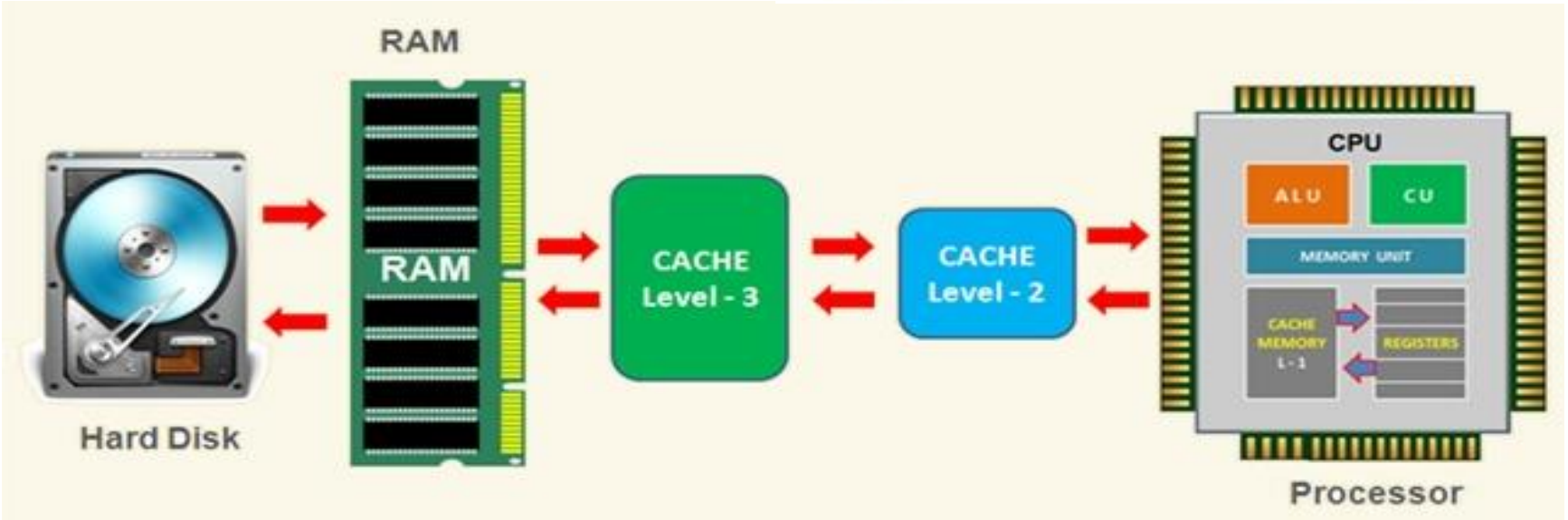
Hierarchy of Memories: why

Programmers want memory to be fast and large but cheap as well.
Fast and large memory is very expensive.

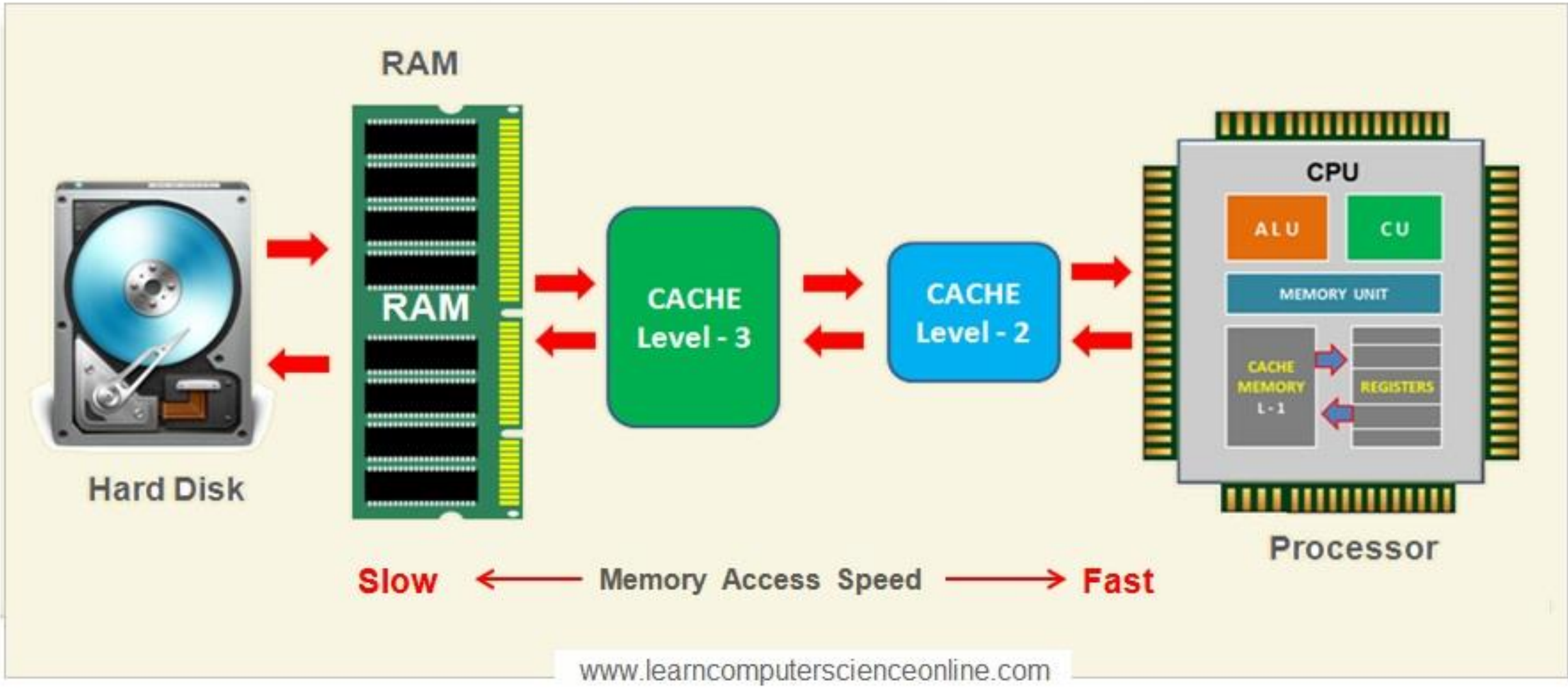
Solution: hierarchy of memories to optimize cost and speed.





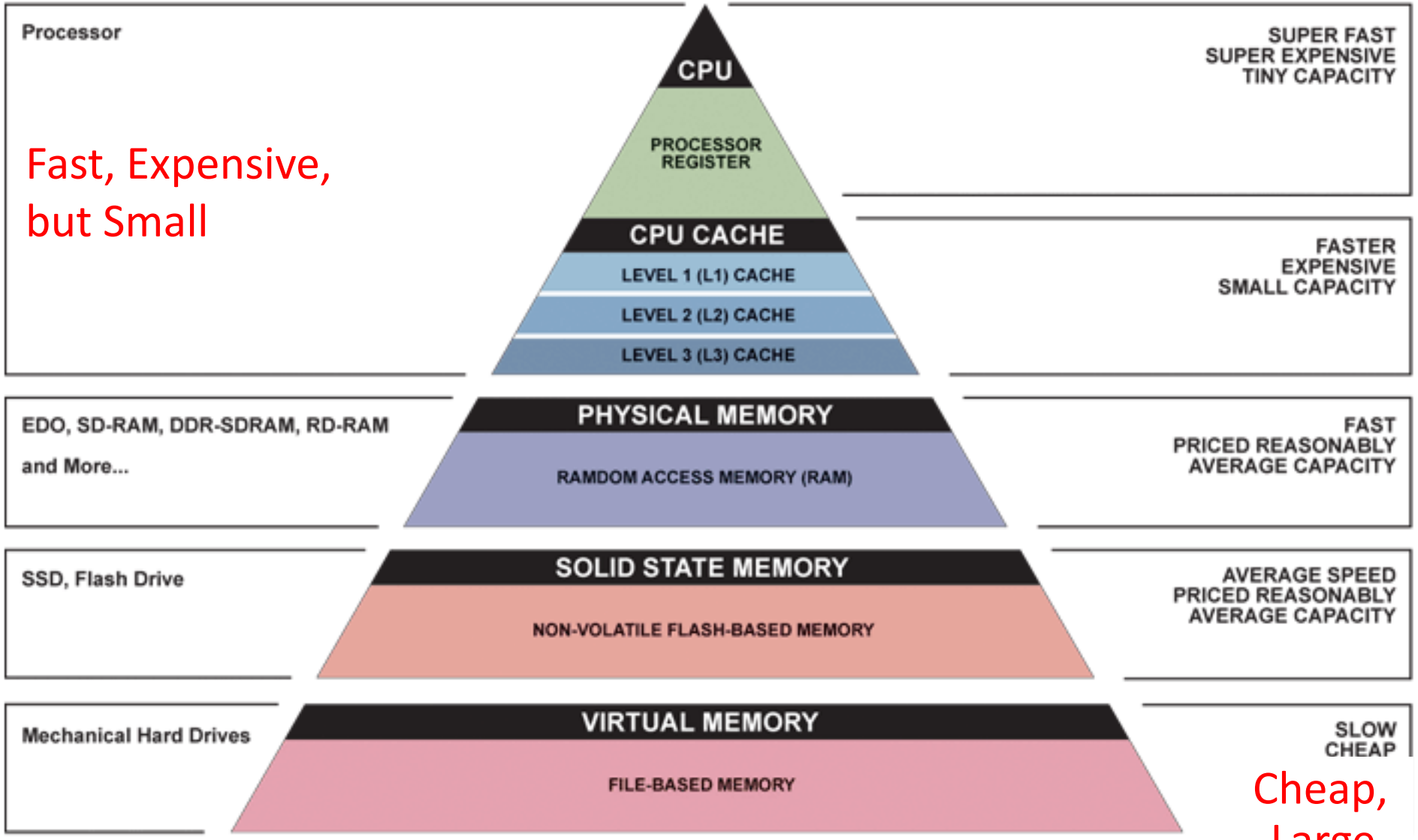


Computer System Memory Hierarchy



Memory Hierarchy

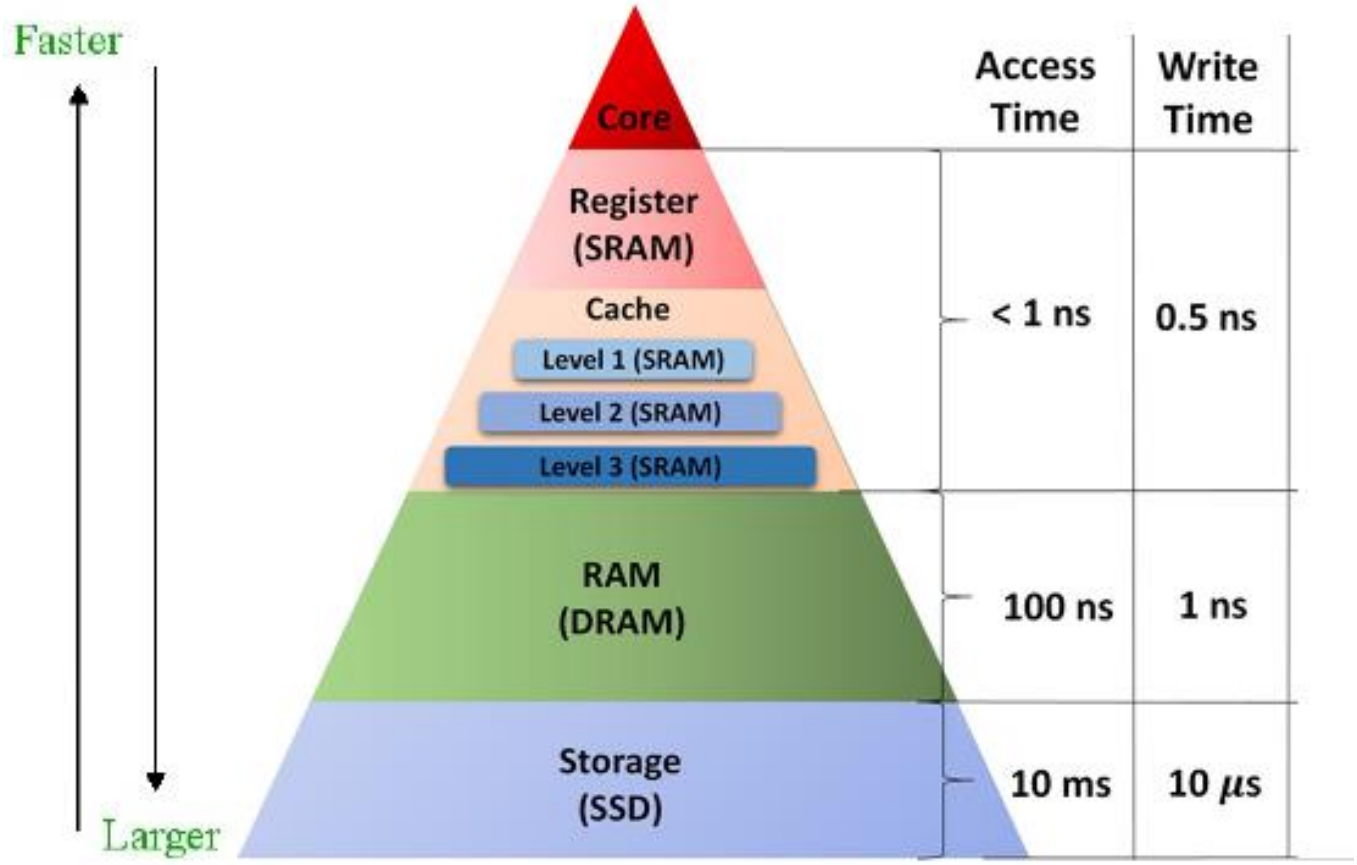
Memory hierarchy goal: look ≈as fast as most expensive memory, ≈ as big as cheapest



Fast, Expensive, but Small

Cheap, Large, but Small

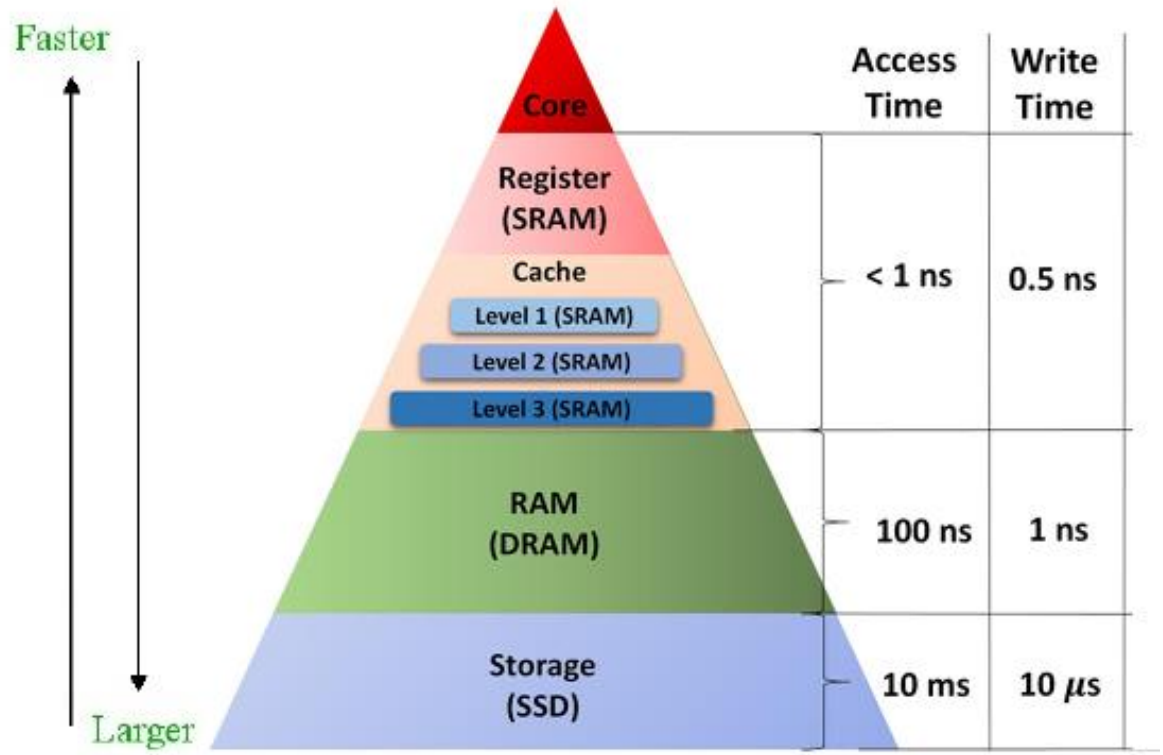
Memory Hierarchy: Optimize Cost and Access Time



The fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.

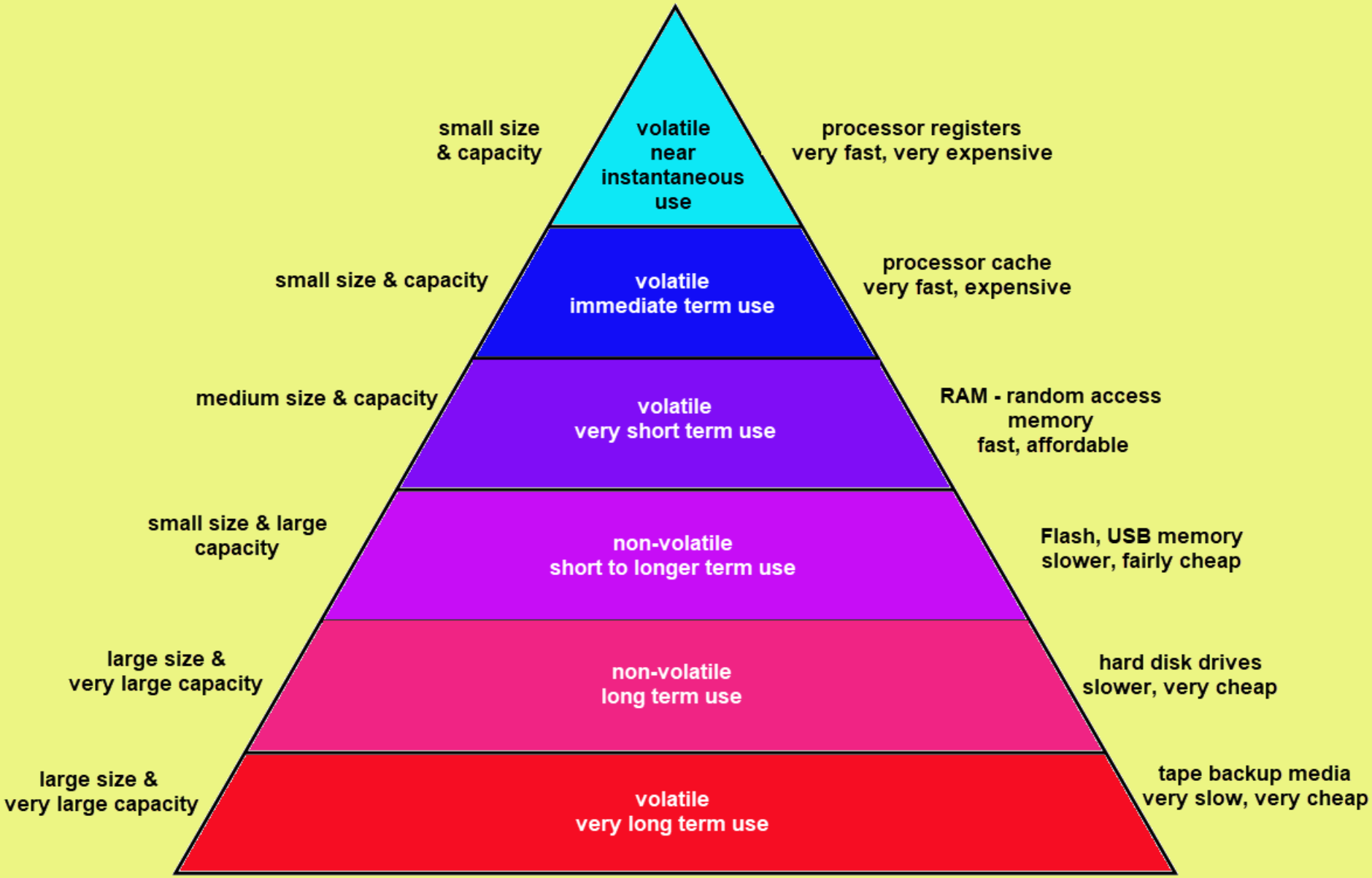
Caches give the programmer the illusion that main memory is nearly as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy.

Hierarchy of Memories



The principle of locality states that memory that has been accessed recently is likely to be accessed again in the near future. That is, accessing recently accessed data is a common case for memory accesses. To make this common case faster you need a cache — a small high-speed memory designed to hold recently accessed data.

Computer Memory Hierarchy



Memory

Memory

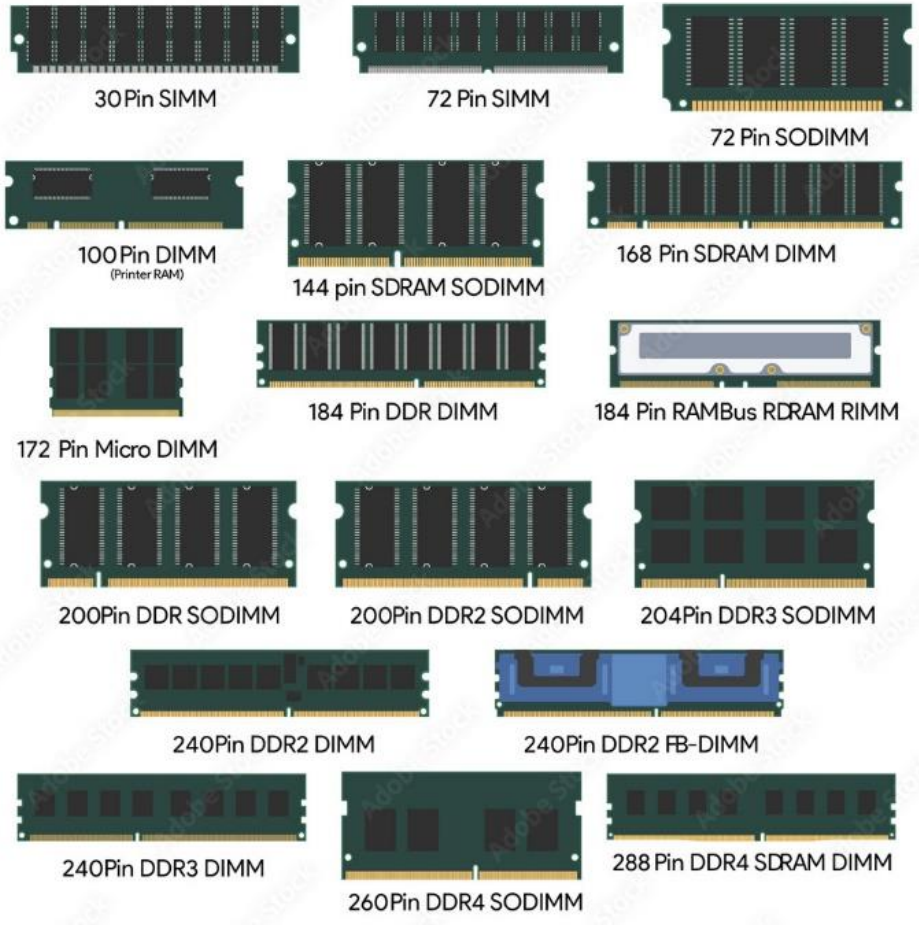
- CPU has only few registers but most computational tasks require a lot more memory.
- Main memory is the next fastest memory within a computer and is much larger in size.
- RAM (Random Access Memory) is the most common form of Main Memory. RAM is normally located on the motherboard.
- ROM (Read Only Memory) is like RAM except that its contents cannot be overwritten and its contents are not lost if power is turned off (ROM is non-volatile).

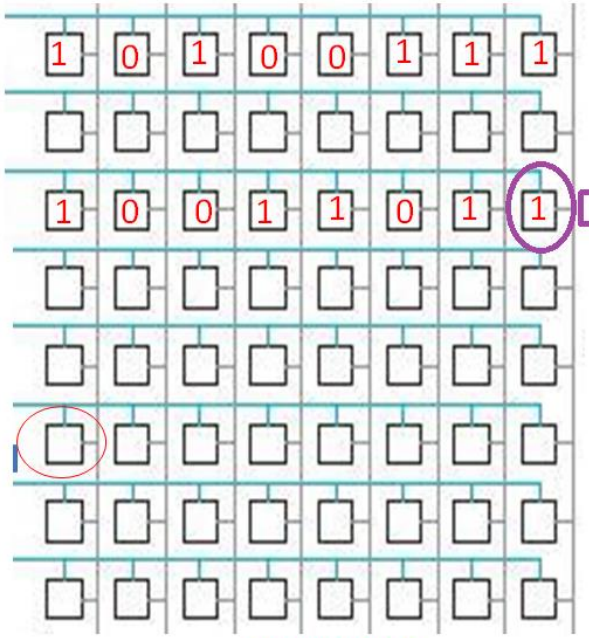
- Although slower than register memory, the contents of any location in RAM can still be “read” or “written” very quickly. The time to read or write is referred to as the **access time** and is constant for all RAM locations.
- RAM is used to hold both program code (instructions) and data (numbers, strings etc).
- Programs are “loaded” into RAM from a disk prior to execution by the CPU.
- Locations in RAM are identified by an **addressing scheme** *e.g.* numbering the bytes in RAM from 0 onwards.

Types of RAM

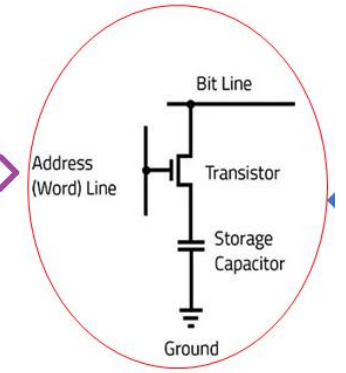
- There are many kinds of RAM and new ones are invented all the time. One of aims is to make RAM access as fast as possible in order to keep up with the increasing speed of CPUs.
- SRAM (Static RAM) is the fastest form of RAM but also the most expensive. Due to its cost it is not used as main memory but rather for cache memory. Each bit requires a 6-transistor circuit.
- DRAM (Dynamic RAM) is not as fast as SRAM but is cheaper and is used for main memory. Each bit uses a single capacitor and single transistor circuit. Since capacitors lose their charge, DRAM needs to be refreshed every few milliseconds. The memory system does this transparently. There are many implementations of DRAM, two well-known ones are SDRAM and DDR SDRAM.
- SDRAM (Synchronous DRAM) is a form of DRAM that is synchronised with the clock of the CPU's system bus, sometimes called the front-side bus (FSB). As an example, if the system bus operates at 167Mhz over an 8-byte (64-bit) data bus , then an SDRAM module could transfer 167 x 8 ~ 1.3GB/sec.
- DDR SDRAM (Double-Data Rate DRAM) is an optimisation of SDRAM that allows data to be transferred on both the rising edge and falling edge of a clock signal. Effectively doubling the amount of data that can be transferred in a period of time. For example a PC-3200 DDR-SDRAM module operating at 200Mhz can transfer 200 x 8 x 2 ~ 3.2GB/sec over an 8-byte (64-bit) data bus.

Memory is an array of storage, each having capacity of 8 bits or so and holds program and data in binary format

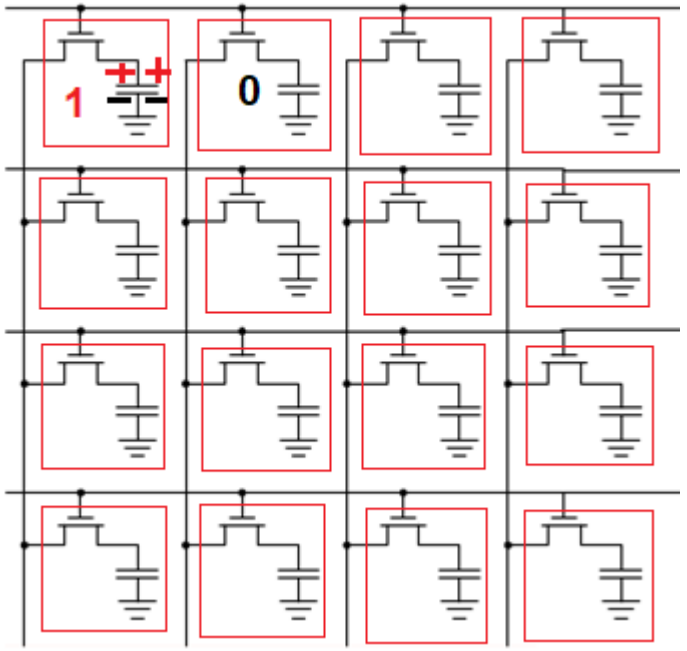




Internal of IC



Single Memory Cell



High-level program

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

COMPILER

Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

INTERPRETER

Machine code

```
0001001001000101
0010010011101100
10101101001...
```

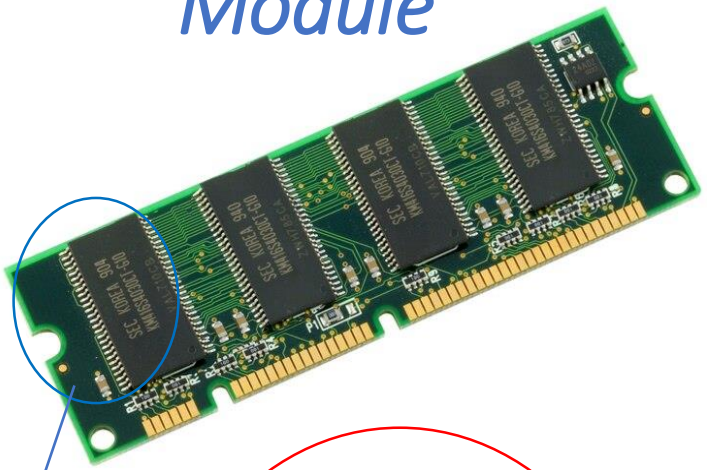
Adobe Stock | #280576861

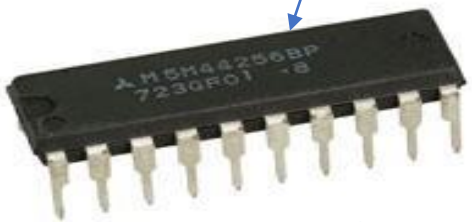
Memory is an array of storage, each having capacity of 8 bits or so and holds program and data in binary format

User Program & Data

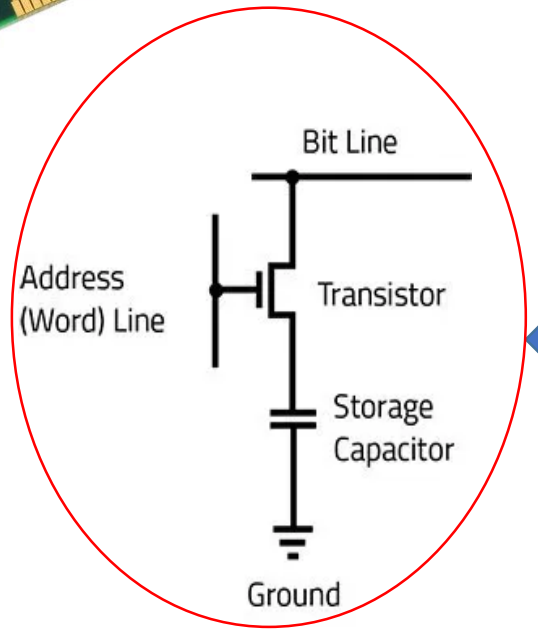
0101000001111111
1100000111100000
0101010100001111
1110000000111111
0101000001111111
1100000111100000
0101010100001111
1110000000111111

Module

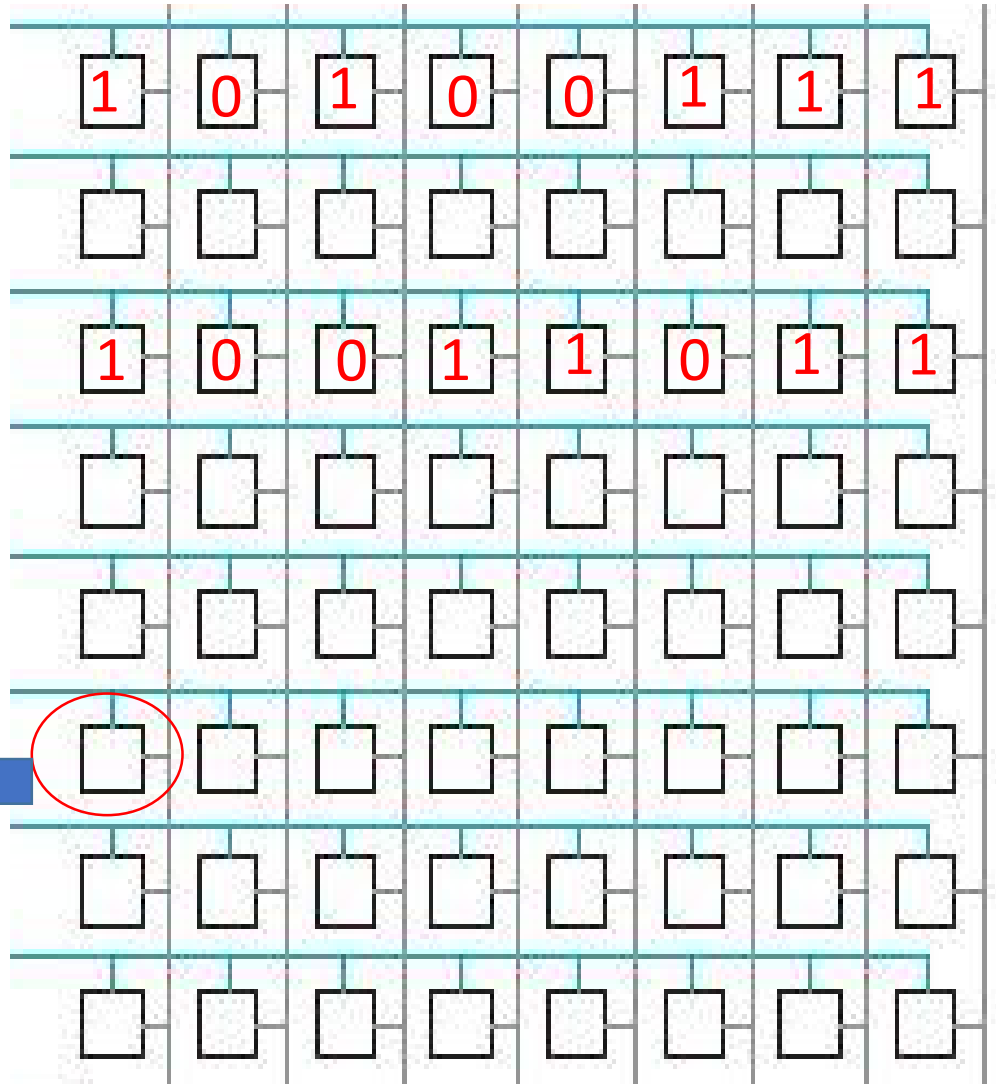




IC



Single Memory Cell



Internal of IC

High-level program

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

COMPILER

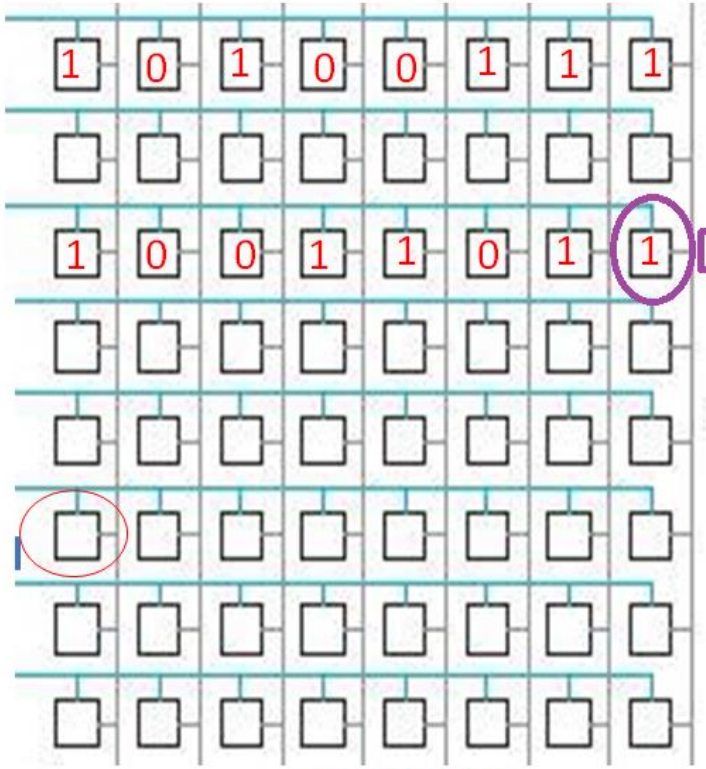
Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

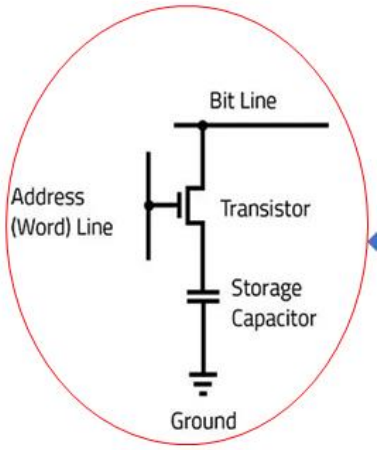
INTERPRETER

Machine code

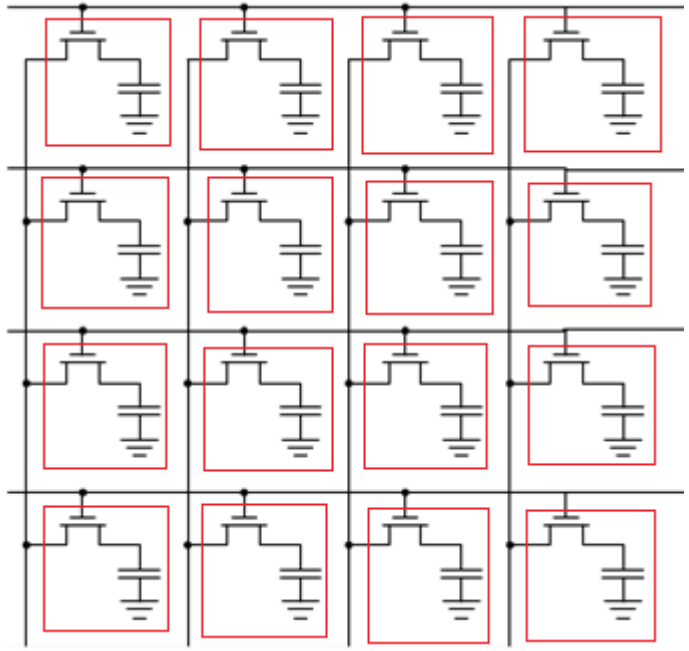
```
0001001001000101
0010010011101100
10101101001...
```

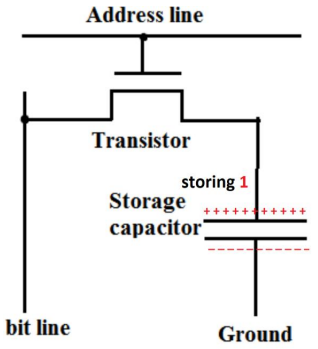


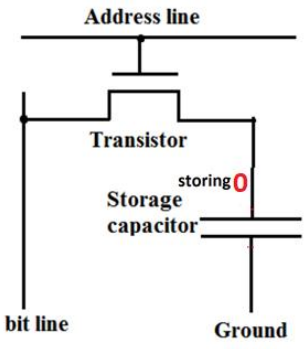
Internal of IC



Single Memory Cell



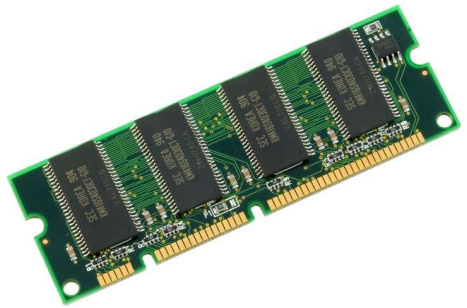


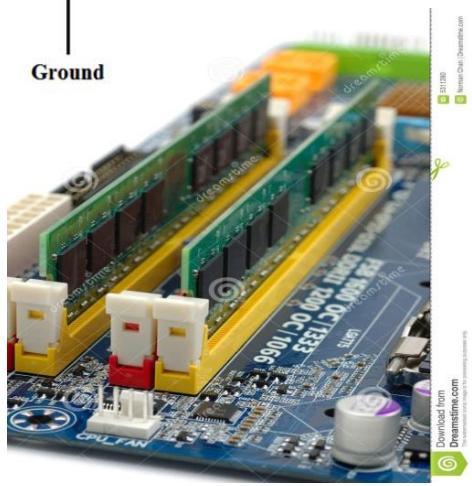


Program and Data

```
0101000001111111
1100000111100000
0101010100001111
1110000000111111
0101000001111111
1100000111100000
0101010100001111
1110000000111111
```

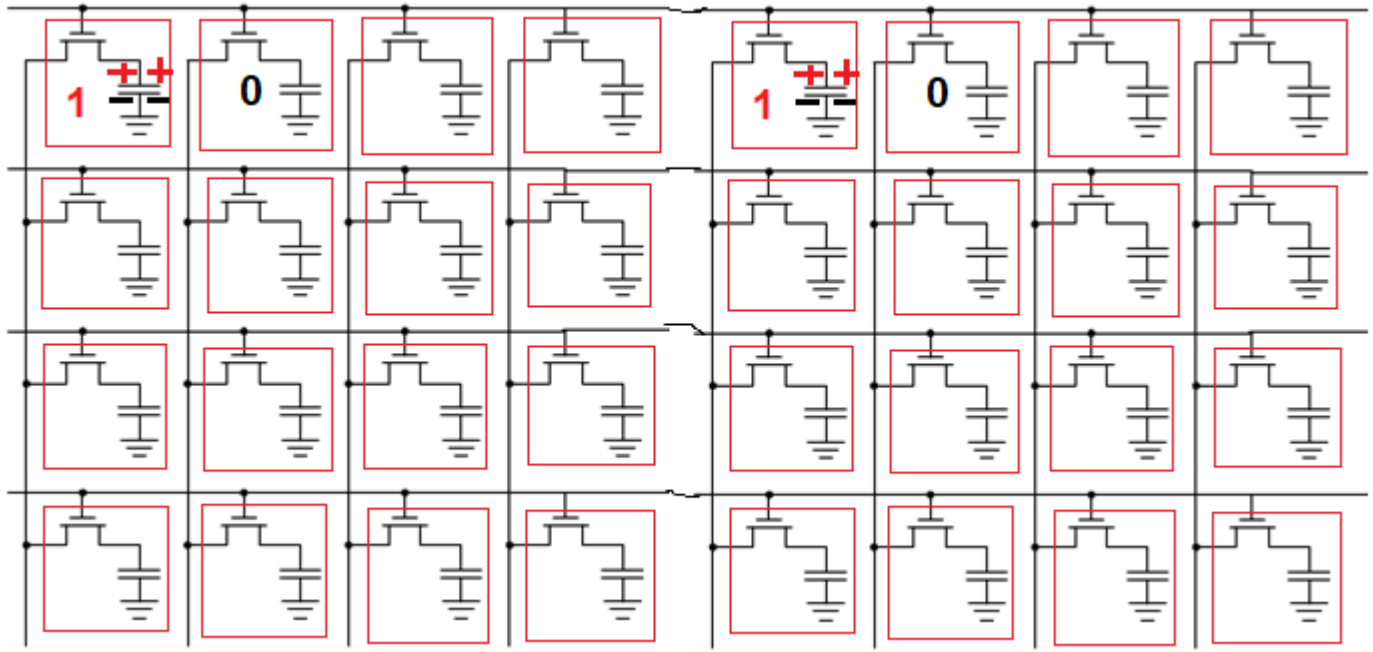
Stored in RAM

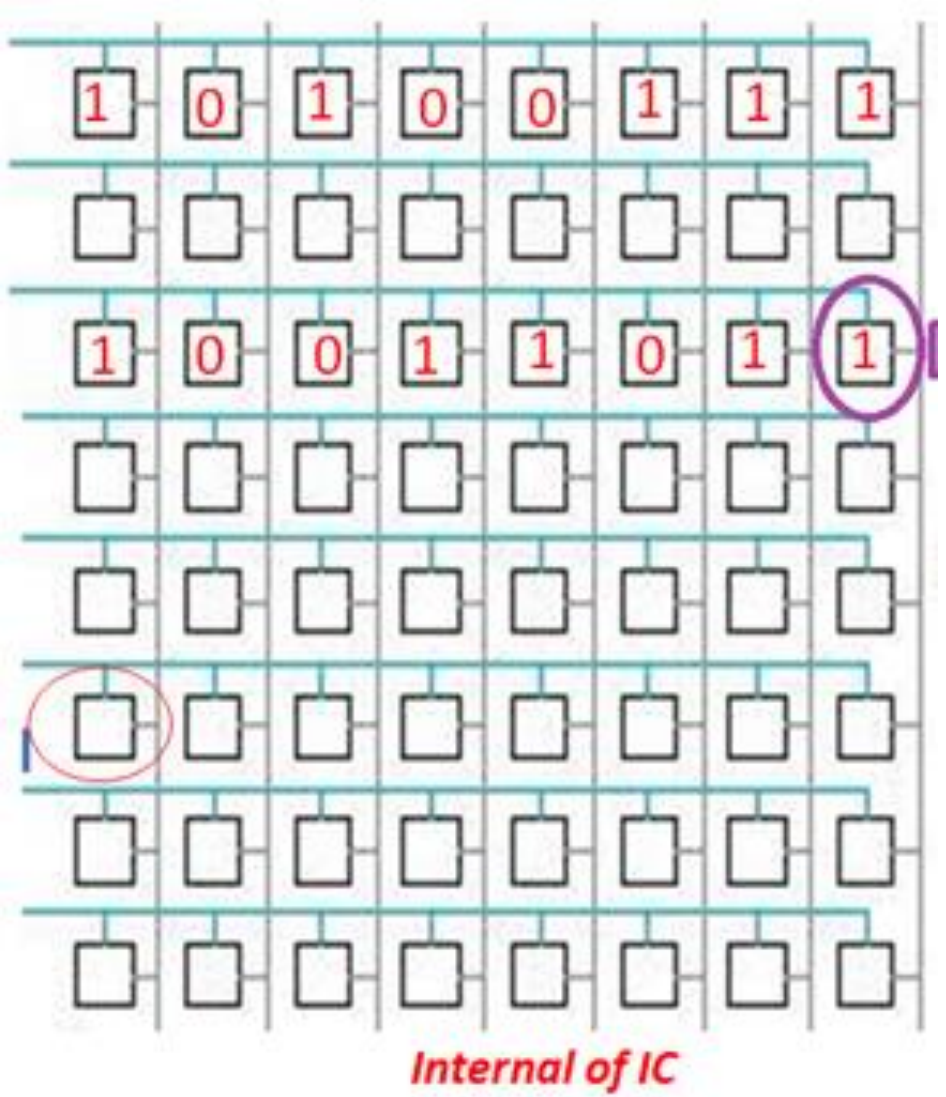


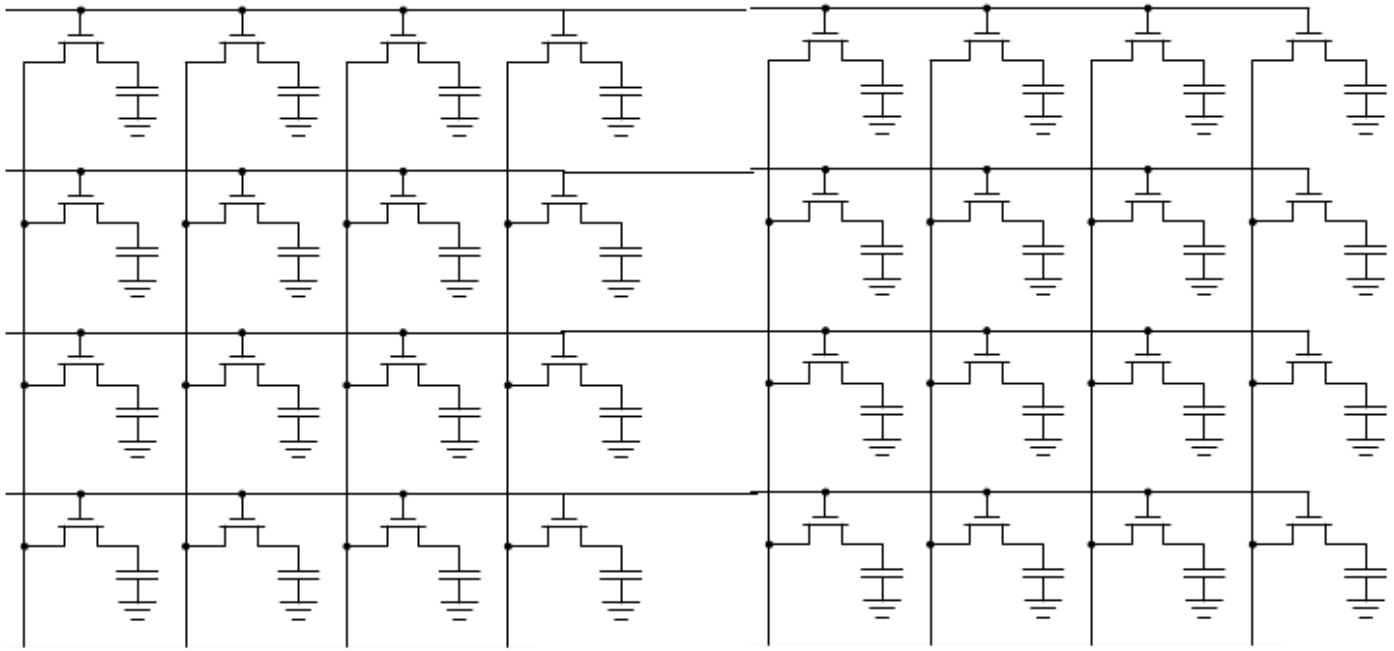


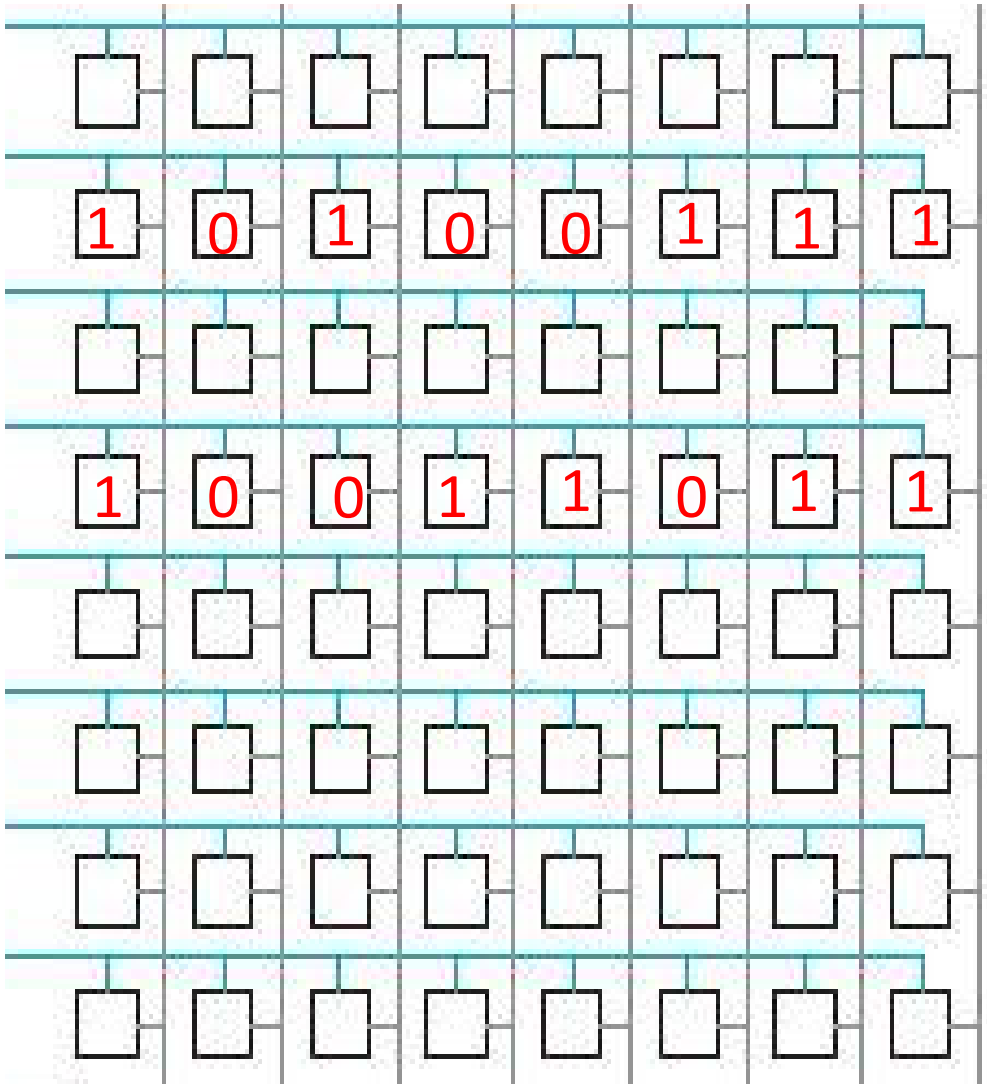
Main Memory

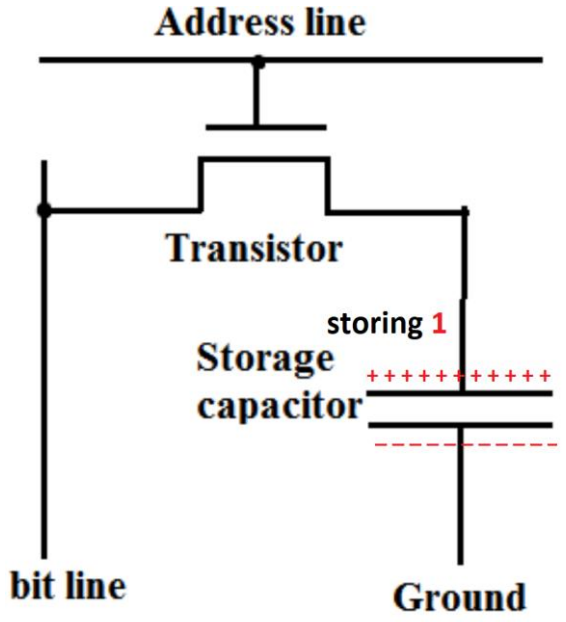
- Main memory can be considered to be organised as a matrix of bits.
- Each row represents a memory location, typically this is 1 Byte (8 cells contain 8 bits)

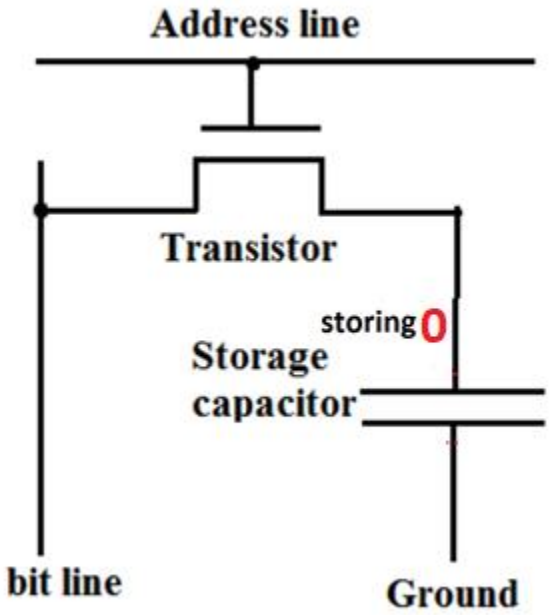




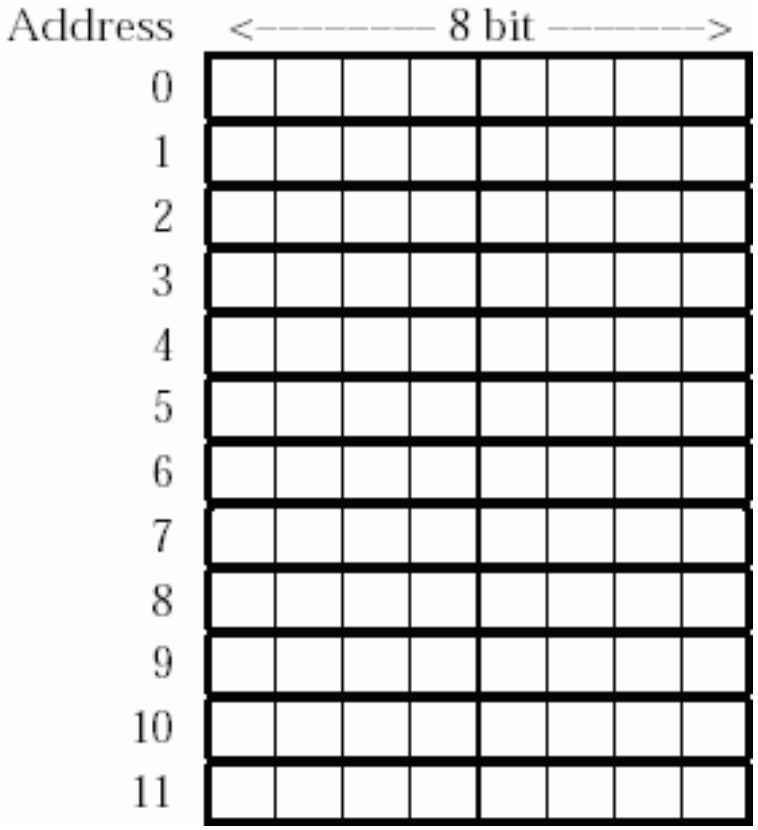


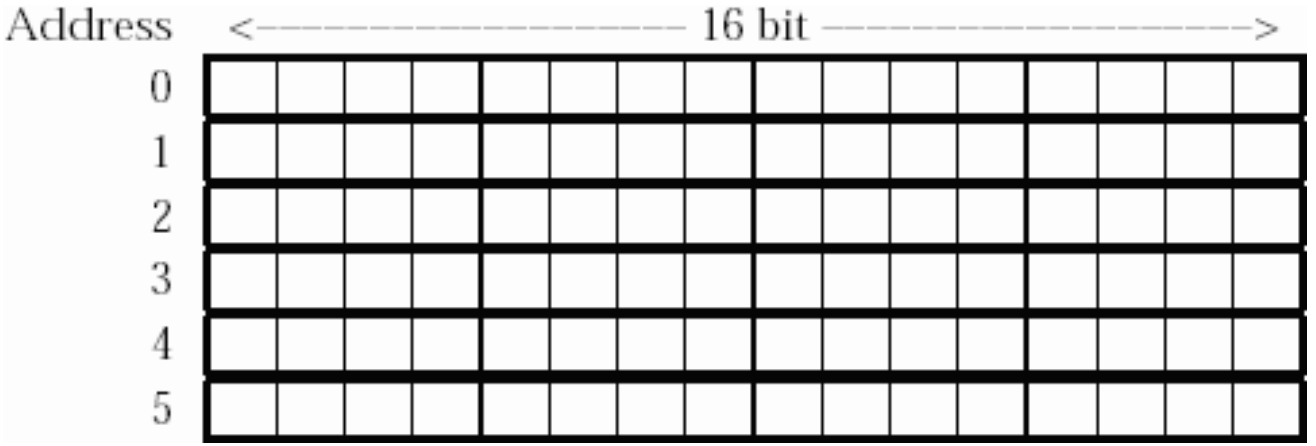

1 0 1 0 0 1 1 1
1 0 0 1 1 0 1 1

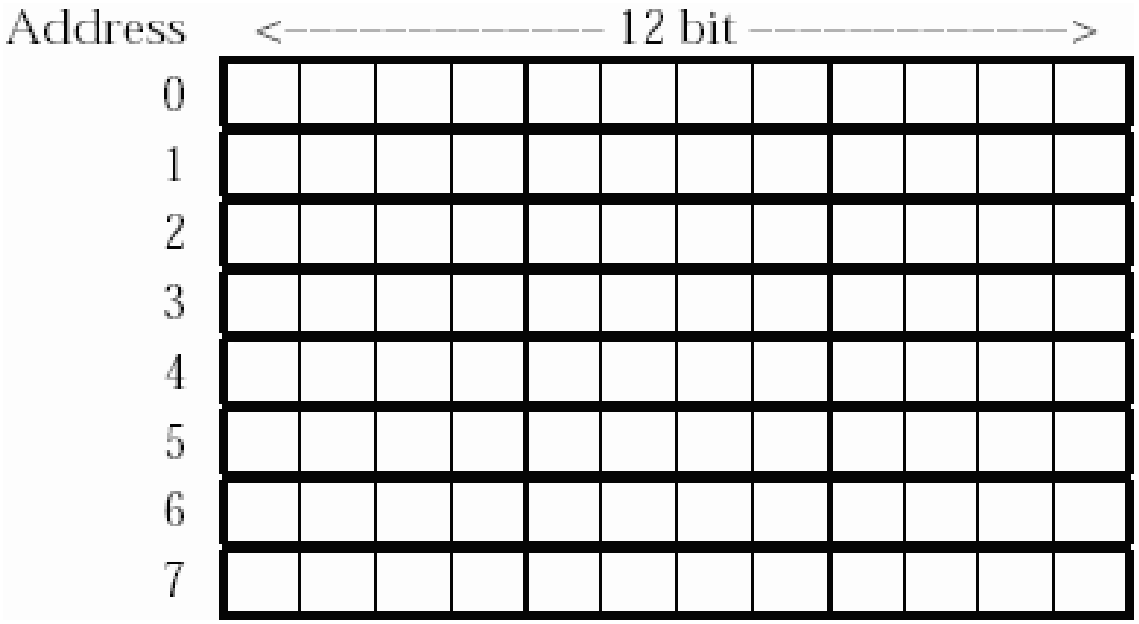

Address line
Transistor
storing 1
Storage capacitor
bit line
Ground


Address line
Transistor
storing 0
Storage capacitor
bit line
Ground

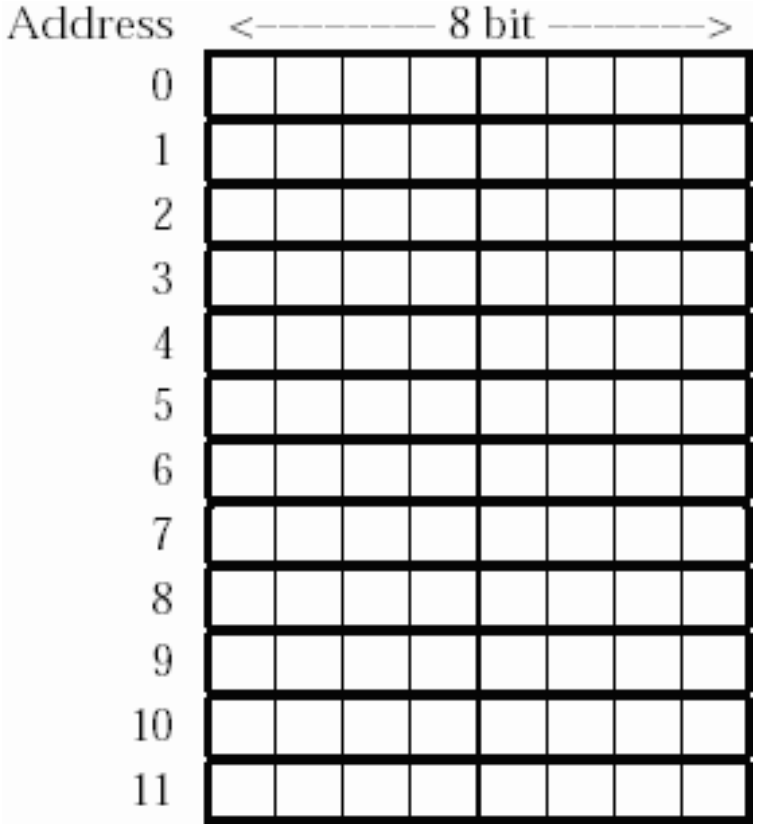
- For a 96-bit memory we could organise the memory as 12x8 bits, or 8x12 bits or, 6x16 bits, or even as 96x1 bits or 1x96 bits.
- Each row also has a natural number called its **address** which is used for selecting the row







- Most architectures however, make Main Memory byte-addressable.
- In such architectures the CPU and/or the Main Memory hardware is capable of reading/writing any individual byte.



Each row is uniquely identified by a number/code, starting from ‘0’, called Address, usually represented in Hexadecimal form and used in Assembly language programming

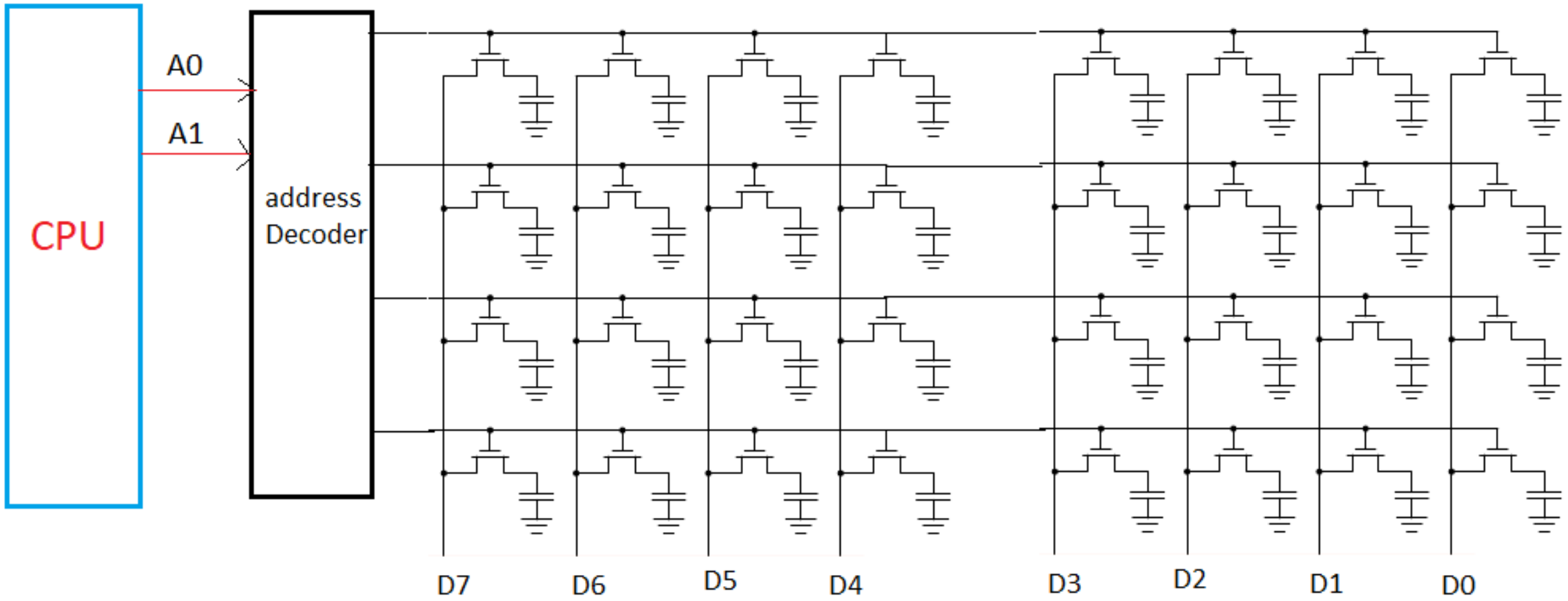
Address in HEX	Address in Binary				Contents (Machine code & data)
0H	0	0	0	0	1 0 0 0 1 1 1 0
1H	0	0	0	1	0 1 0 0 0 1 1 1
2H	0	0	1	0	
3H	0	0	1	1	
7H	0	1	1	1	

If each location of memory contains 8 bits or 1 byte, then it is called

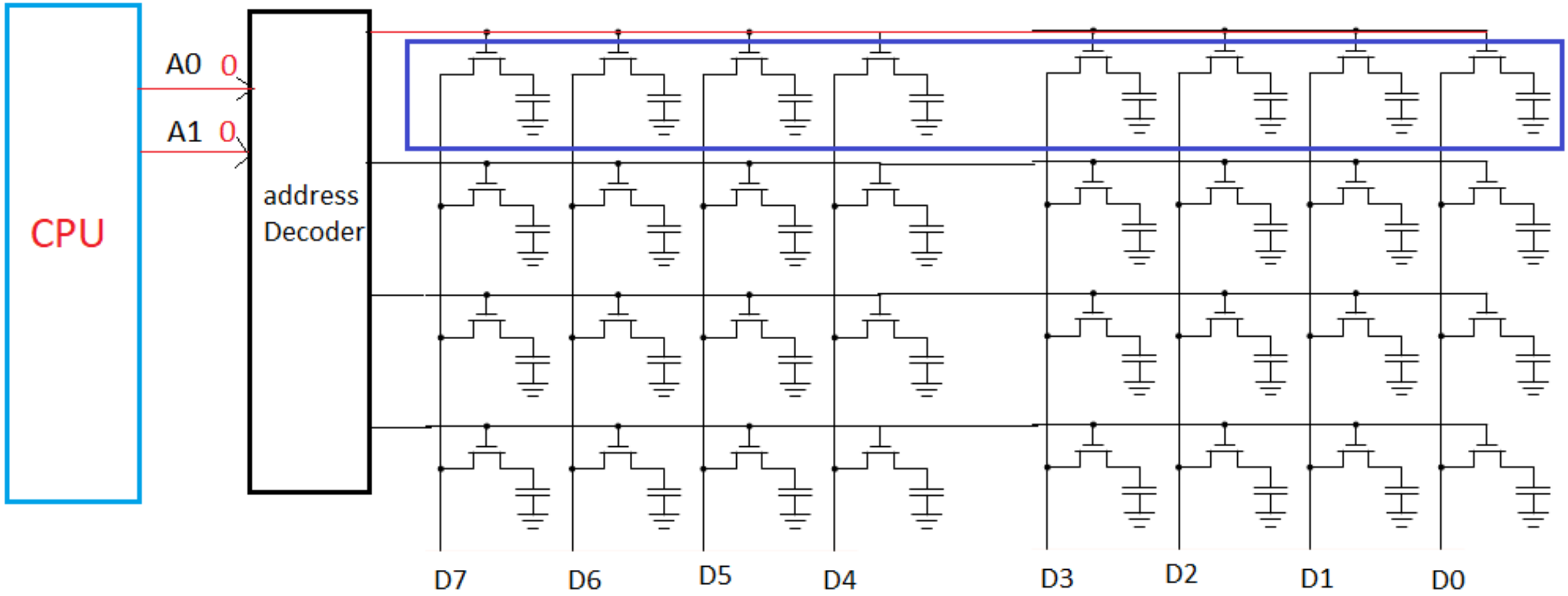
Byte-Addressable memory

Address in HEX	Address in Binary	Contents (Machine code or data)
0H	0 0 0 0	1 0 0 0 1 1 1 0
1H	0 0 0 1	0 1 0 0 0 1 1 1
2H	0 0 1 0	
3H	0 0 1 1	
7H	0 1 1 1	1 0 1 0 1 0 0 1

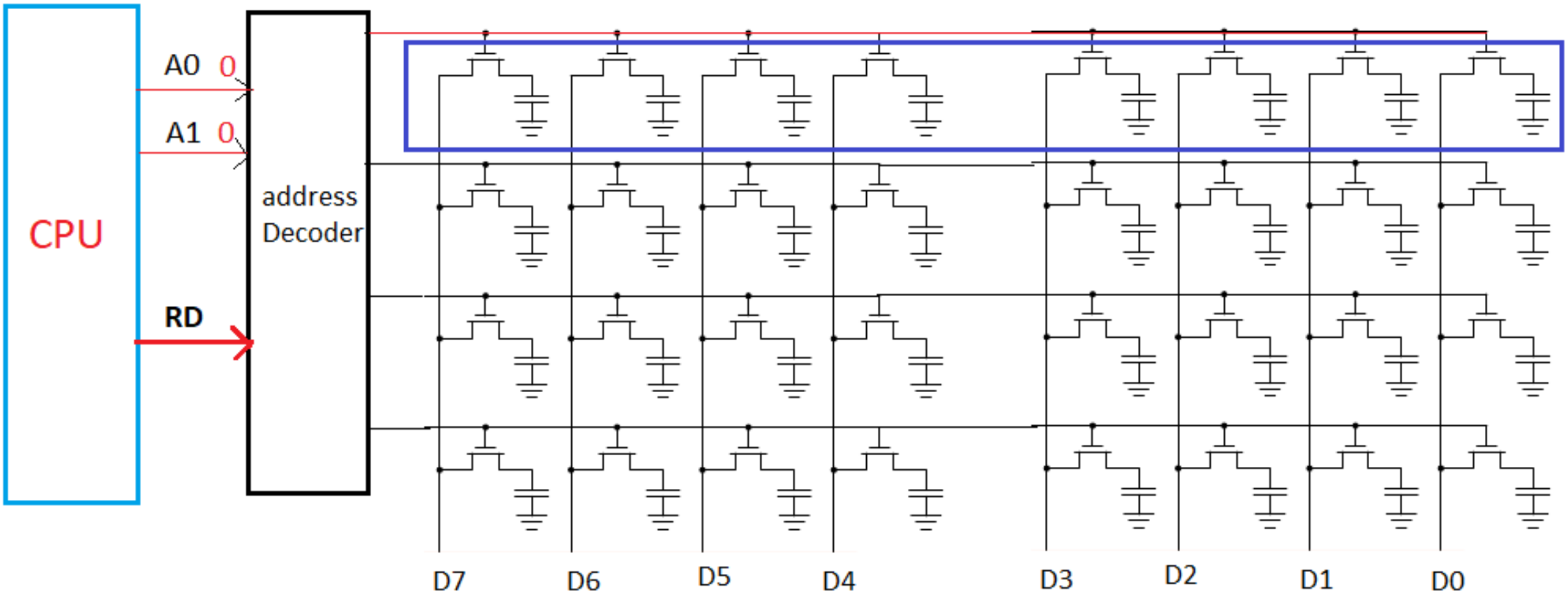
Interfacing CPU-RAM



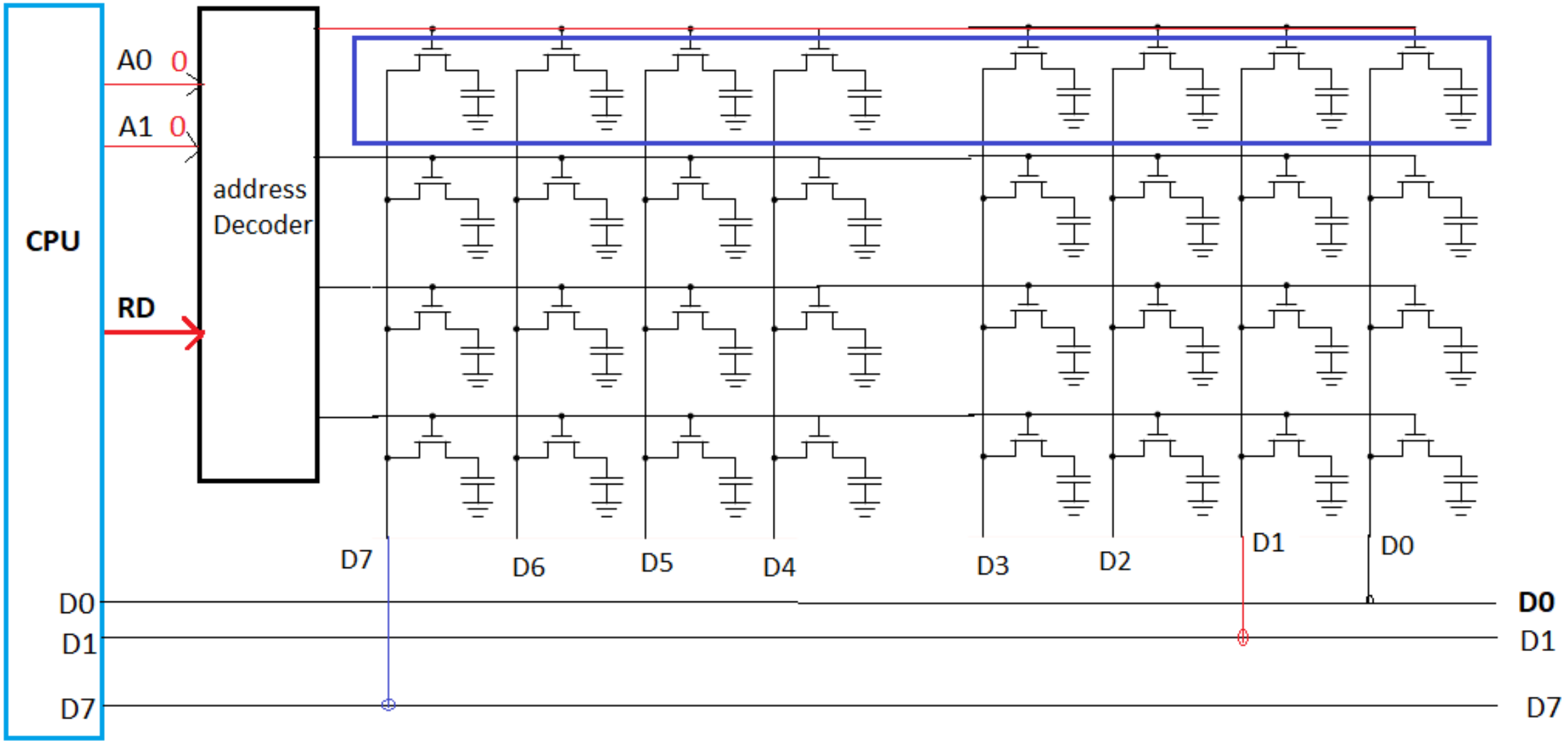
Interfacing CPU-RAM



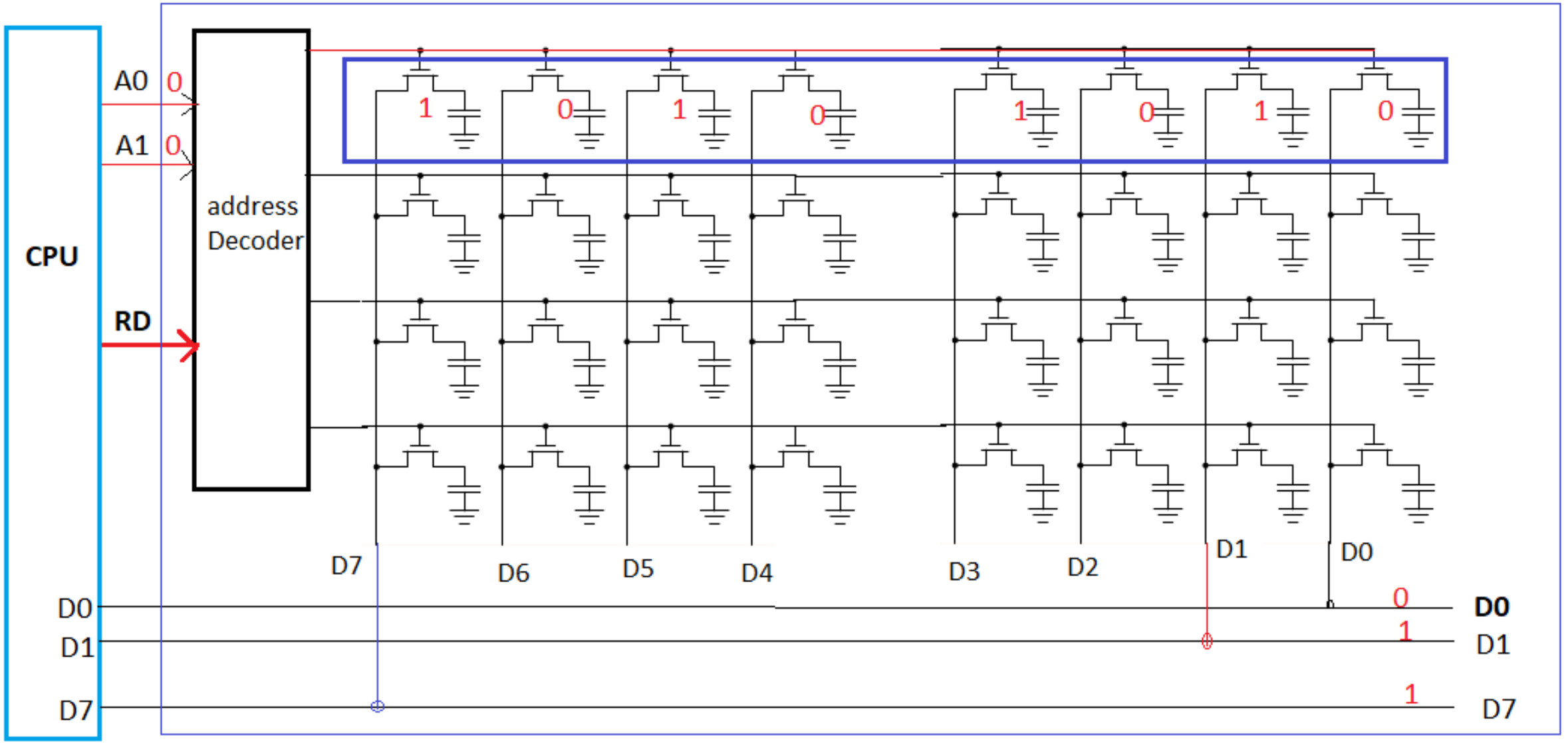
Interfacing CPU-RAM

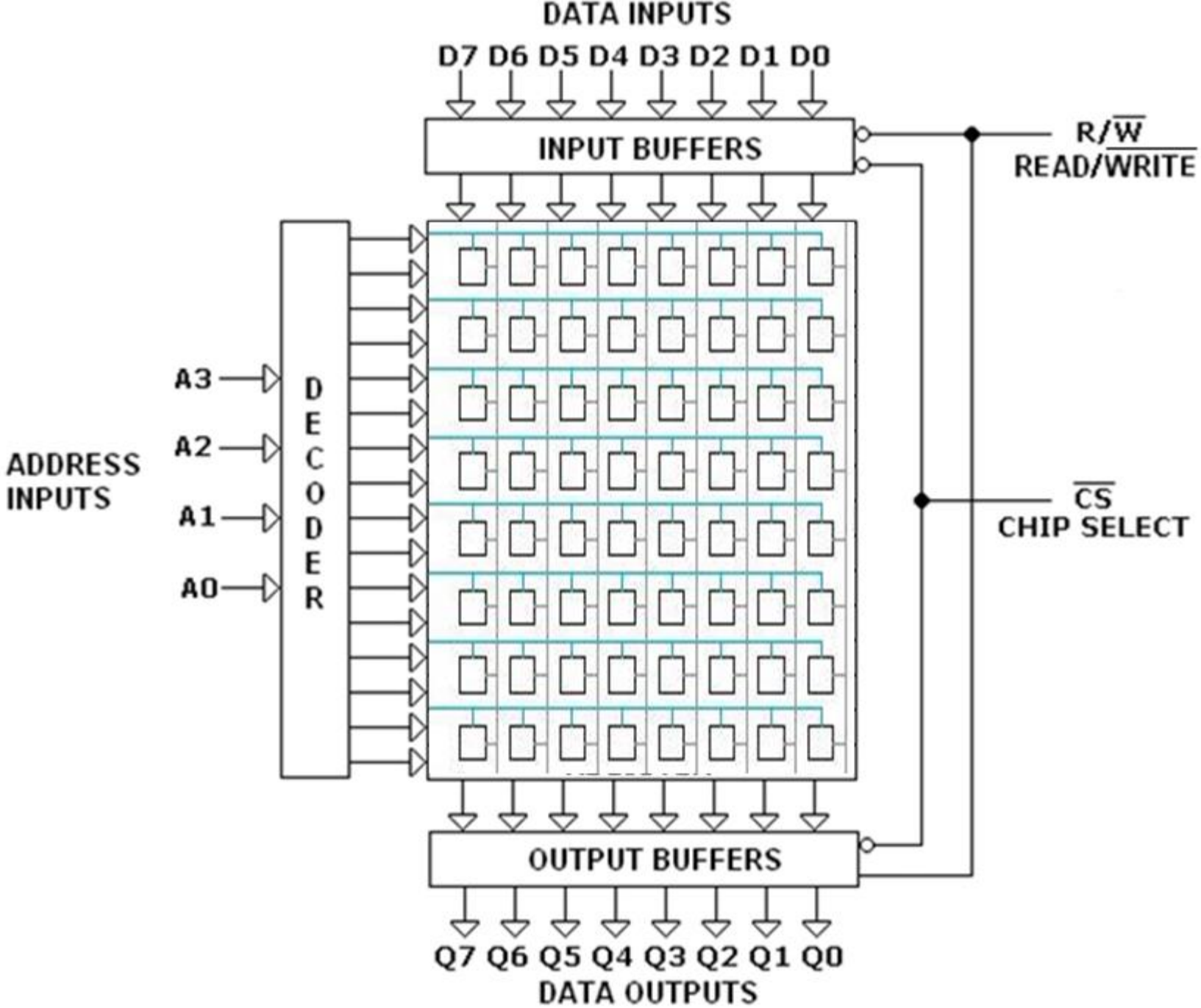


Interfacing CPU-RAM



Interfacing CPU-RAM




DATA INPUTS
D7 D6 D5 D4 D3 D2 D1 D0
INPUT BUFFERS
R/W
READ/WRITE
A3
A2
A1
A0
ADDRESS INPUTS
DECODER
CS
CHIP SELECT
OUTPUT BUFFERS
Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
DATA OUTPUTS

Overview of Memory Organization

- Memory is one of the most important sub-systems of a computer that determines the overall performance.
- Conceptual view of memory:
  - Array of storage locations, with each storage location having a unique address.
  - Each storage location can hold a fixed amount of information (multiple of bits, which is the basic unit of data storage).
- A memory system with *M* locations and *N* bits per location, is referred to as an *M x N* memory.
  - Both *M* and *N* are typically some powers of 2.
  - Example: 1024 x 8, 65536 x 32, etc.

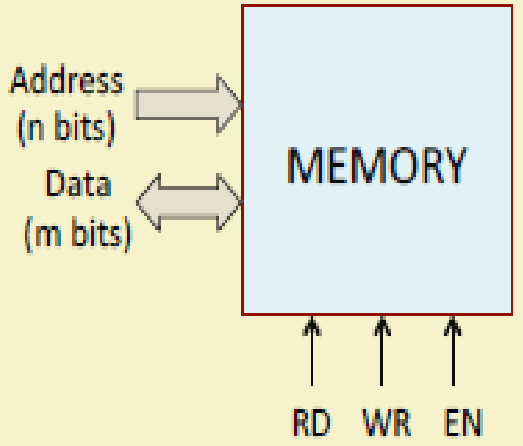
Some Terminologies

- Bit: A single binary digit (0 or 1).
- Nibble: Collection of 4 bits.
- Byte: Collection of 8 bits.
- Word: Does not have a unique definition.
  - Varies from one computer to another; typically 32 or 64 bits.

How do we Specify Memory Sizes?

Unit	Bytes	In Decimal
8 bits (B)	1 or 2^0	10^0
Kilobyte (KB)	1024 or 2^{10}	10^3
Megabyte (MB)	1,048,576 or 2^{20}	10^6
Gigabyte (GB)	1,073,741,824 or 2^{30}	10^9
Terabyte (TB)	1,099,511,627,776 or 2^{40}	10^{12}
Petabyte (PB)	2^{50}	10^{15}
Exabyte (EB)	2^{60}	10^{18}
Zettabyte (ZB)	2^{70}	10^{21}

- If there are *n* bits in the address, the maximum number of storage locations can be 2^n.
  - For n=8, 256 locations.
  - For n=16, 64K locations.
  - For n=20, 1M locations.
  - For n=32, 4G locations.
- Modern-day memory chips can store several Gigabits of data.
  - Dynamic RAM (DRAM).



Example: 1KB IC (D2708)

- 1KB: Approximately 1 Thousand (exact value 1024) locations each having capacity of 8 bits/1Byte
- Address starts at 0 and ends at 1 less than 1 Thousand, actually encoded in BINARY
- In Binary, first address requires 1 bit (0) and final addressable location requires 10 bits (all 1's: 11…11), since $\mathbf{2^{10} = 1K}$
- For ease of Decoder design, uniform address format is used for all the locations; Maximum number of bits!
- For convenience/ease of representation/programming/discussion, Hexadecimal number system is used to represent Memory address

Physical Address of Memory (for 1KB)

Address (10 bits in binary)	Content (8 bits)
1111111111B	11001100 (machine code/data)
…	
0000000000B	00110101(machine code/data)

Address (3 digits in hexadecimal)	Content(8 bits)
3FFH	11001100 (machine code/data)
000H	00110101(machine code/data)

Capacity of Memory

- Example: 1MB: Approximately 1 Million (exact value 1024 x 1024) locations each having capacity of 1Byte
- Address starts at 0 and ends at 1 less than 1 Million, actually encoded in BINARY
- In Binary, first address requires 1 bit (0) and final addressable location requires 20 bits (all 1's: 11…11), since $\mathbf{2^{20} = 1M}$
- For ease of Decoder design, uniform address format is used for all the locations; Maximum number of bits!
- For convenience/ease of representation/programming/discussion, Hexadecimal number system is used to represent Memory address

Some Examples

1. A computer has 64 MB (megabytes) of byte-addressable memory. How many bits are needed in the memory address?
   - Address Space = 64 MB = 2^6 X 2^{20} B = 2^{26} B
   - If the memory is byte addressable, we need 26 bits of address.
2. A computer has 1 GB of memory. Each word in this computer is 32 bits. How many bits are needed to address any single word in memory?
   - Address Space = 1 GB = 2^{30} B
   - 1 word = 32 bits = 4 B
   - We have 2^{30} / 4 = 2^{28} words
   - Thus, we require 28 bits to address each word.

Byte Ordering Conventions

- Many data items require multiple bytes for storage.
- Different computers use different data ordering conventions.
  - Low-order byte first
  - High-order byte first
- Thus a 16-bit number 11001100 10101010 can be stored as either:

11001100	10101010

or

10101010	11001100

Data Type	Size (in Bytes)
Character	1
Integer	4
Long integer	8
Floating-point	4
Double-precision	8

Typical data sizes

Byte Ordering

- The ordering of bytes within a **multi-byte** data item defines the endian-ness of the architecture.
- In BIG-ENDIAN systems the most significant byte of a multi-byte data item always has the lowest address, while the least significant byte has the highest address.

  In LITTLE-ENDIAN systems, the least significant byte of a multi-byte data item always has the lowest address, while the most significant byte has the highest address.
- In the following example, table cells represent bytes, and the cell numbers indicate the address of that byte in main memory. Note: by convention we draw the bytes within a memory word left-to-right for big-endian systems, and right-to-left for little-endian systems.

- The two conventions have been named as:
  a) Little Endian
    - The least significant byte is stored at lower address followed by the most significant byte. Examples: Intel processors, DEC alpha, etc.
    - Same concept followed for arbitrary multi-byte data.
  b) Big Endian
    - The most significant byte is stored at lower address followed by the least significant byte. Examples: IBM's 370 mainframes, Motorola microprocessors, TCP/IP, etc.
    - Same concept followed for arbitrary multi-byte data.

An Example

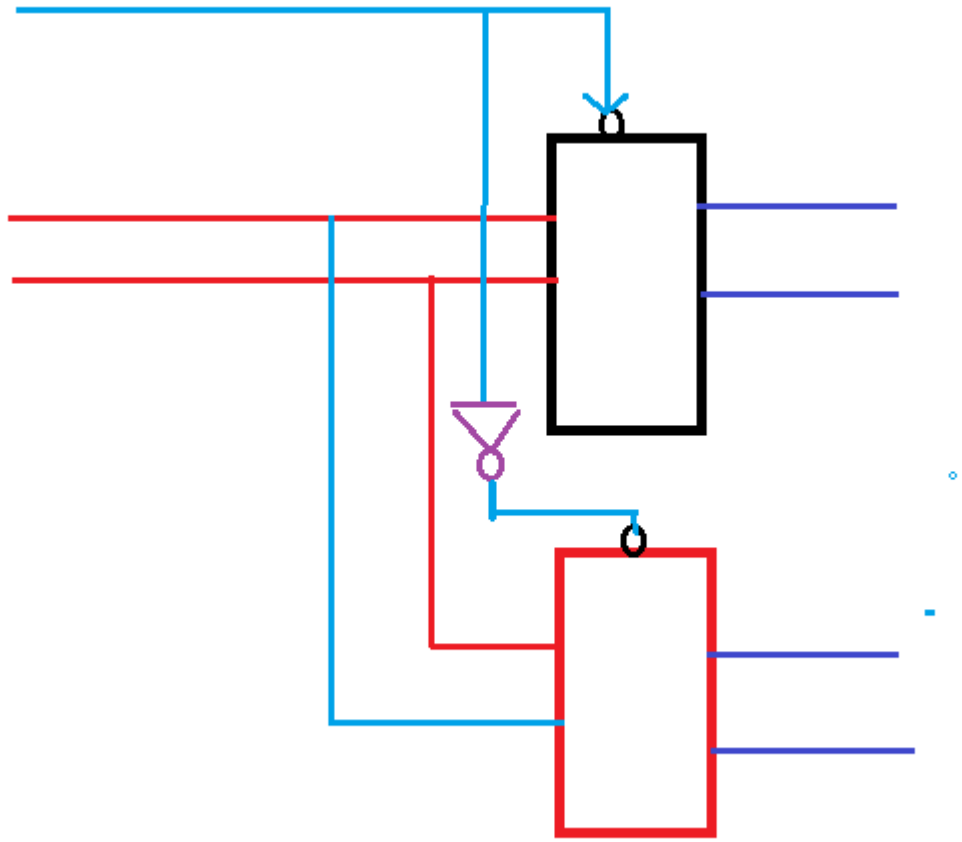
- Represent the following 32-bit number in both Little-Endian and Big-Endian in memory from address 2000 onwards:

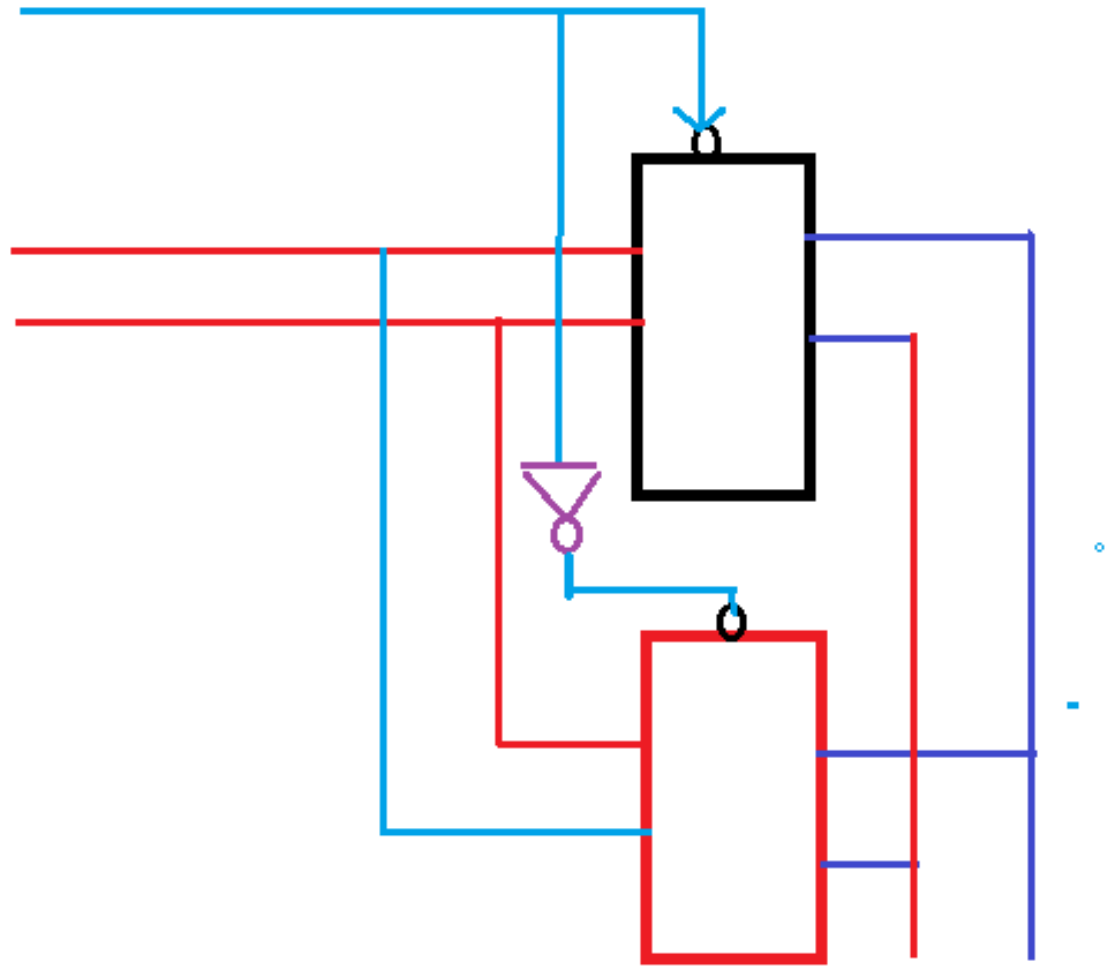
  01010101 00110011 00001111 11000011

Little Endian	
Address	Data
2000	11000011
2001	00001111
2002	00110011
2003	01010101

Big Endian	
Address	Data
2000	01010101
2001	00110011
2002	00001111
2003	11000011

Expanding word size and capacity

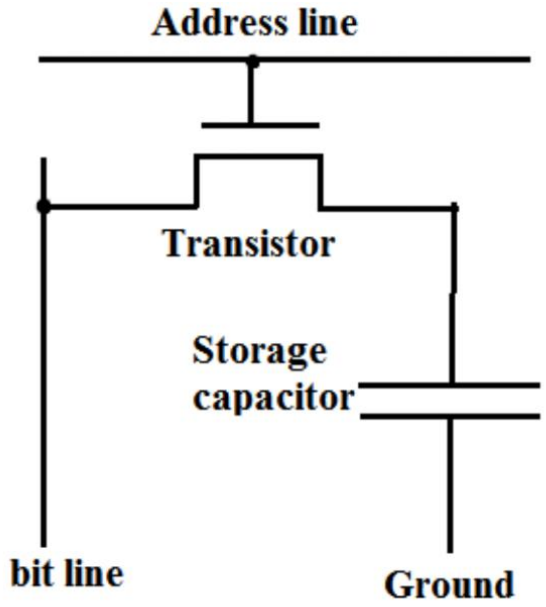




DRAM Basics: Internals, Operation

In DRAM, Binary “1” stored in a cell as a charge in a capacitor.

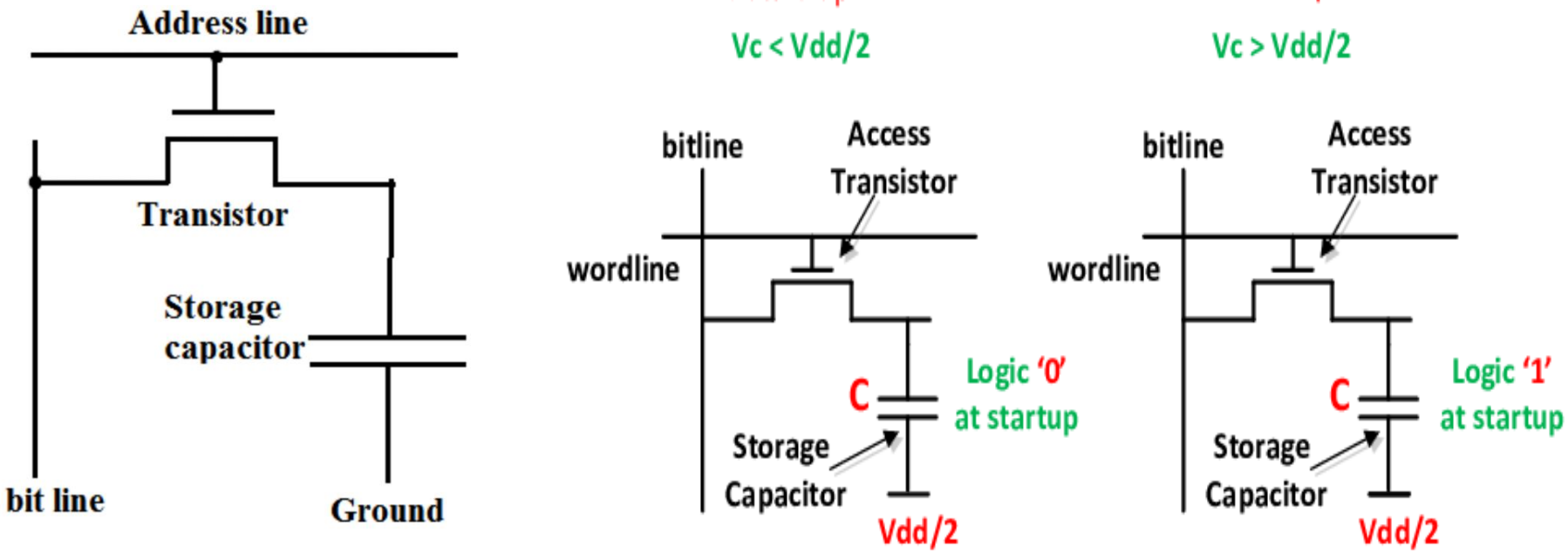
This circuit is *dynamic* because the capacitors cannot store charges indefinitely due to leakage current and discharge .



To retain information stored there, each capacitor in the DRAM must be periodically *refreshed* (i.e., read and rewritten). Refresh interval: 64ms.

A single transistor is then used to access this stored charge, either to read or write

A typical DRAM cell has one transistor and one capacitor, as shown is Figure. It keeps its state in capacitor C. The transistor is used to provide access to the state. To read the state of the cell the access line Address Line is raised; this either causes a current to flow on the bit line or not, depending on the charge in the capacitor. To write to the cell the bit line is appropriately set and then Address Line is raised for a time long enough to charge or drain the capacitor. A DRAM cell is much simpler than a other cell, allowing for greater memory density (being a smaller cell) and lower manufacturing cost.



DRAM has a number of drawbacks:

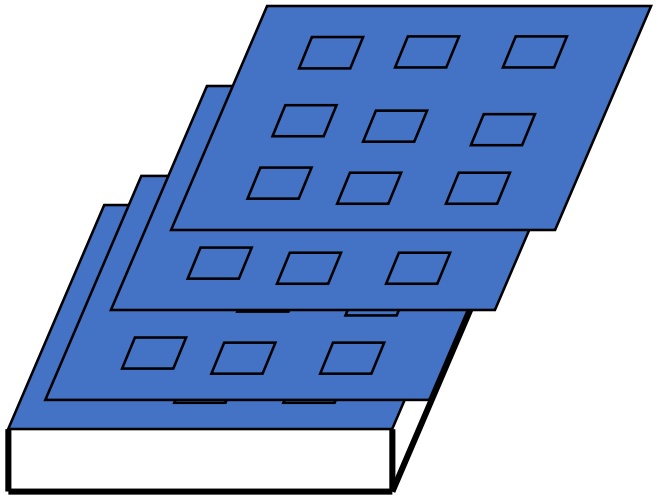
In order to allow huge numbers of cells, the capacity of the capacitor must be low. It only takes a short time for the capacity to dissipate, which is known as current leakage.

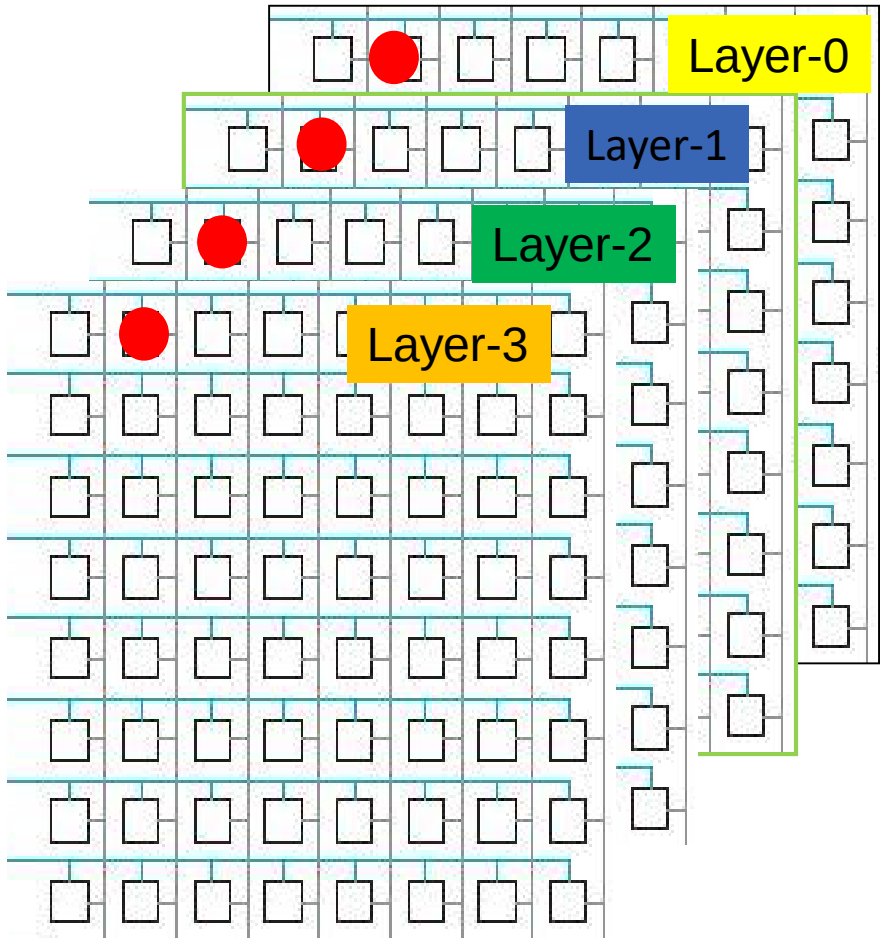
In order to address this problem, DRAM cells must be refreshed frequently (every 64ms in most current DRAM devices).

This tiny charge also creates another issue: the information read from the cell is not immediately usable, since the data line must be connected to a sense amplifier which can distinguish between a stored 0 or 1.

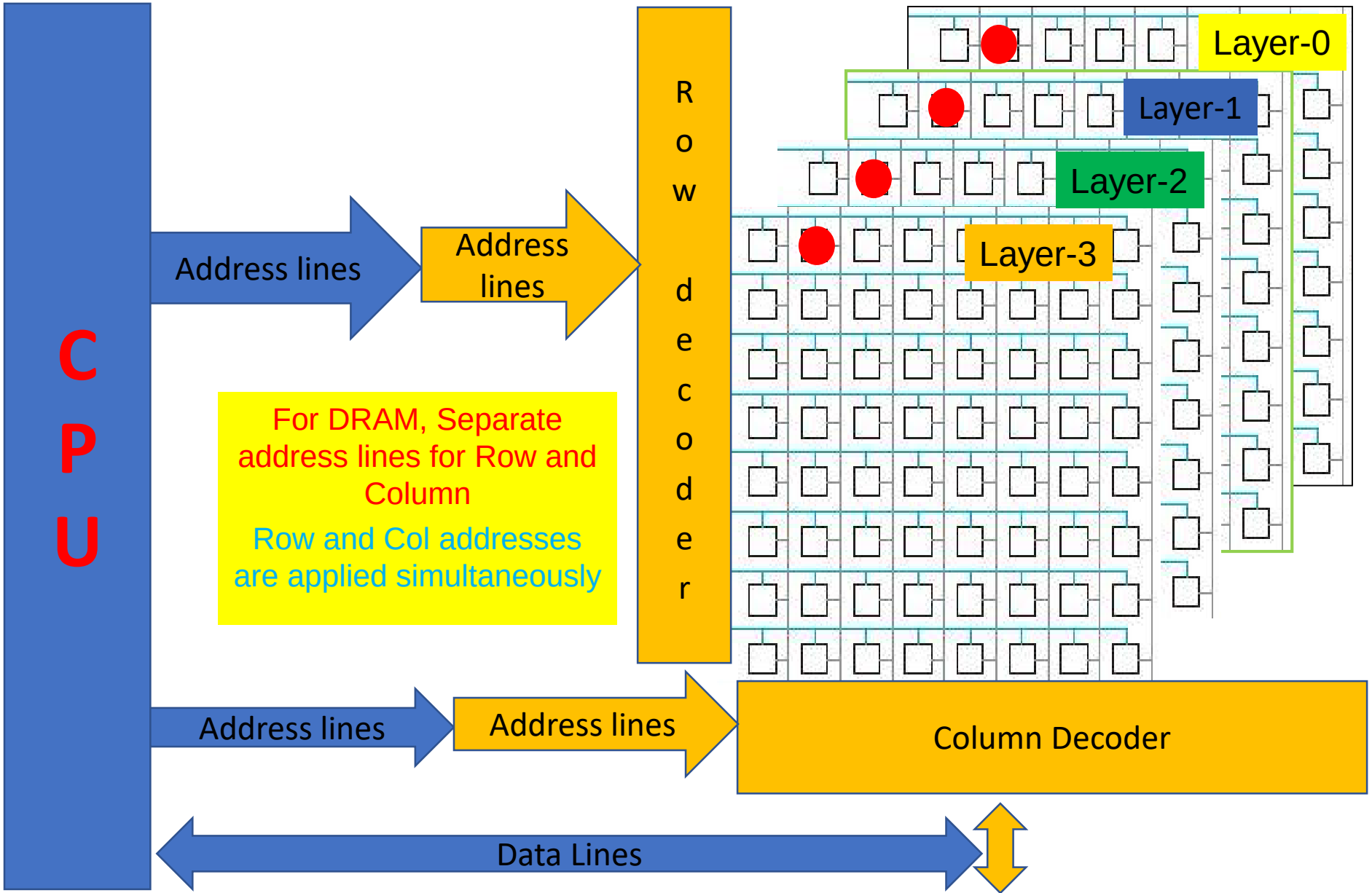
Finally, the capacitor is discharged by read operations, so every read operation must be followed by an operation to recharge the capacitor. This requires time and energy.

Several arrays are read at the same time to provide the contents of a memory word. This group of arrays ganged together are called banks.

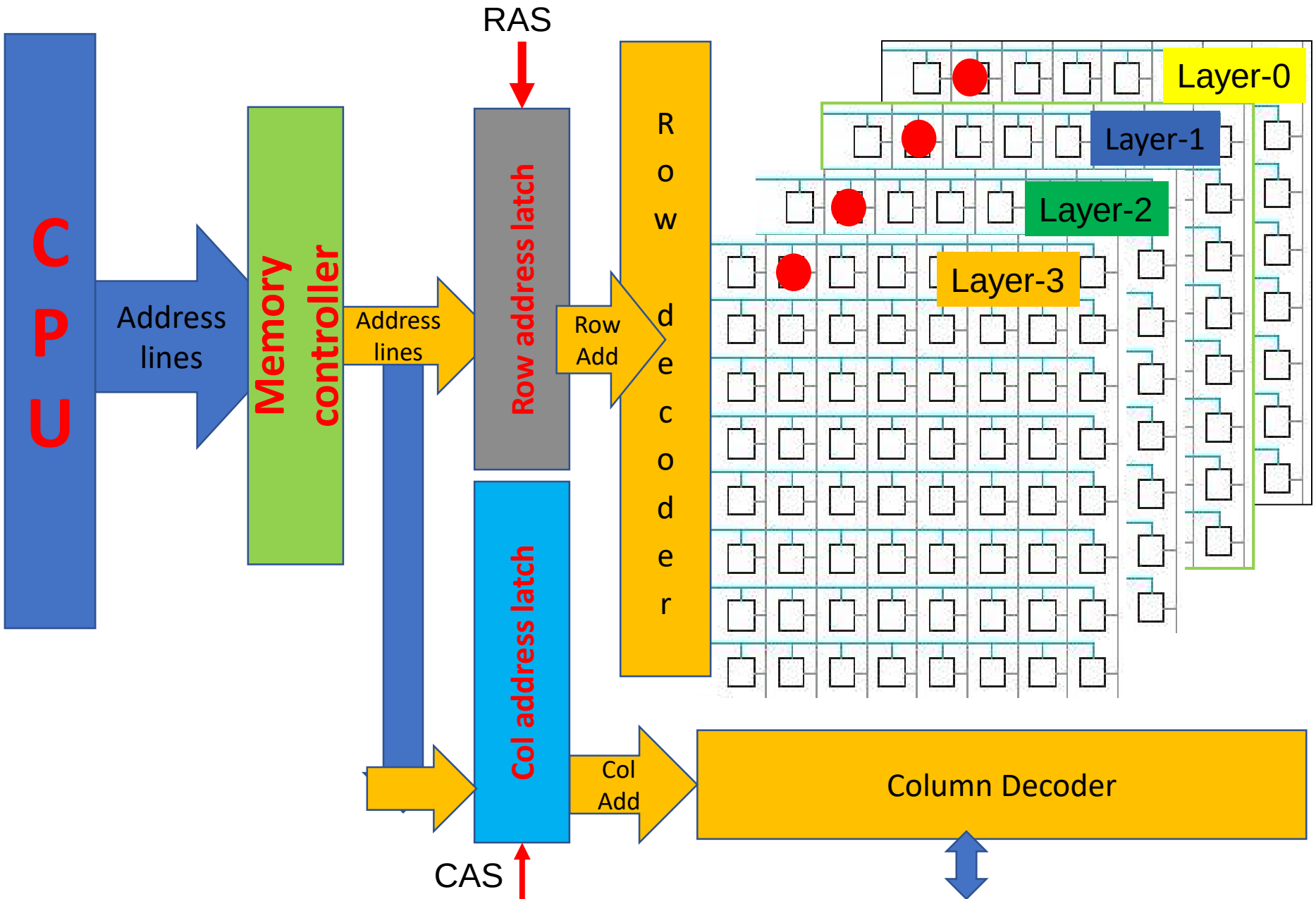




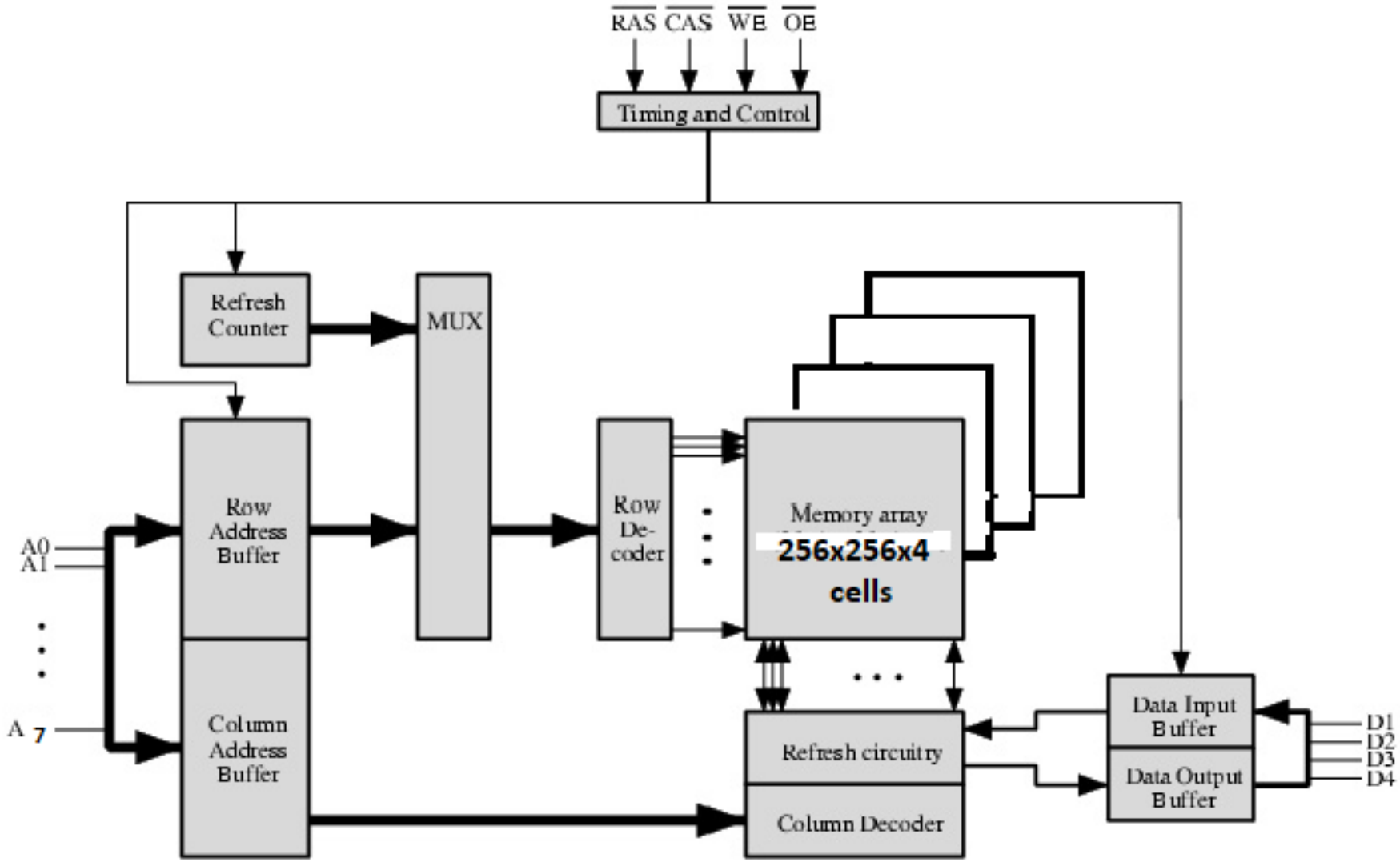
Several arrays are read at the same time to provide the contents of a memory word. This group of arrays ganged together are called banks.



To reduce the size of DRAM IC, the number of address lines is reduced: Memory address pins are reduced to half the required number and multiplexed between row and column



64K x 4 bits DRAM



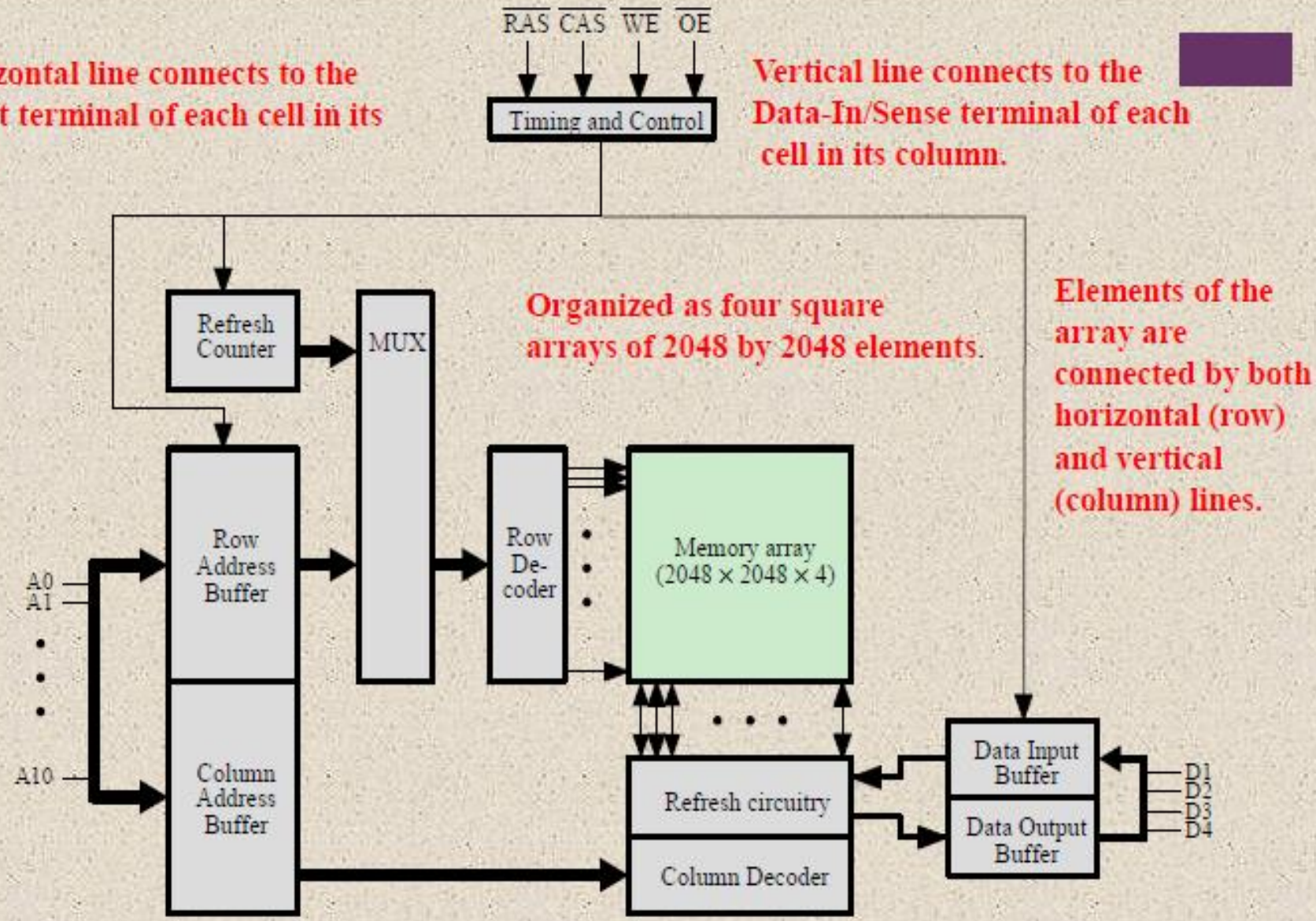
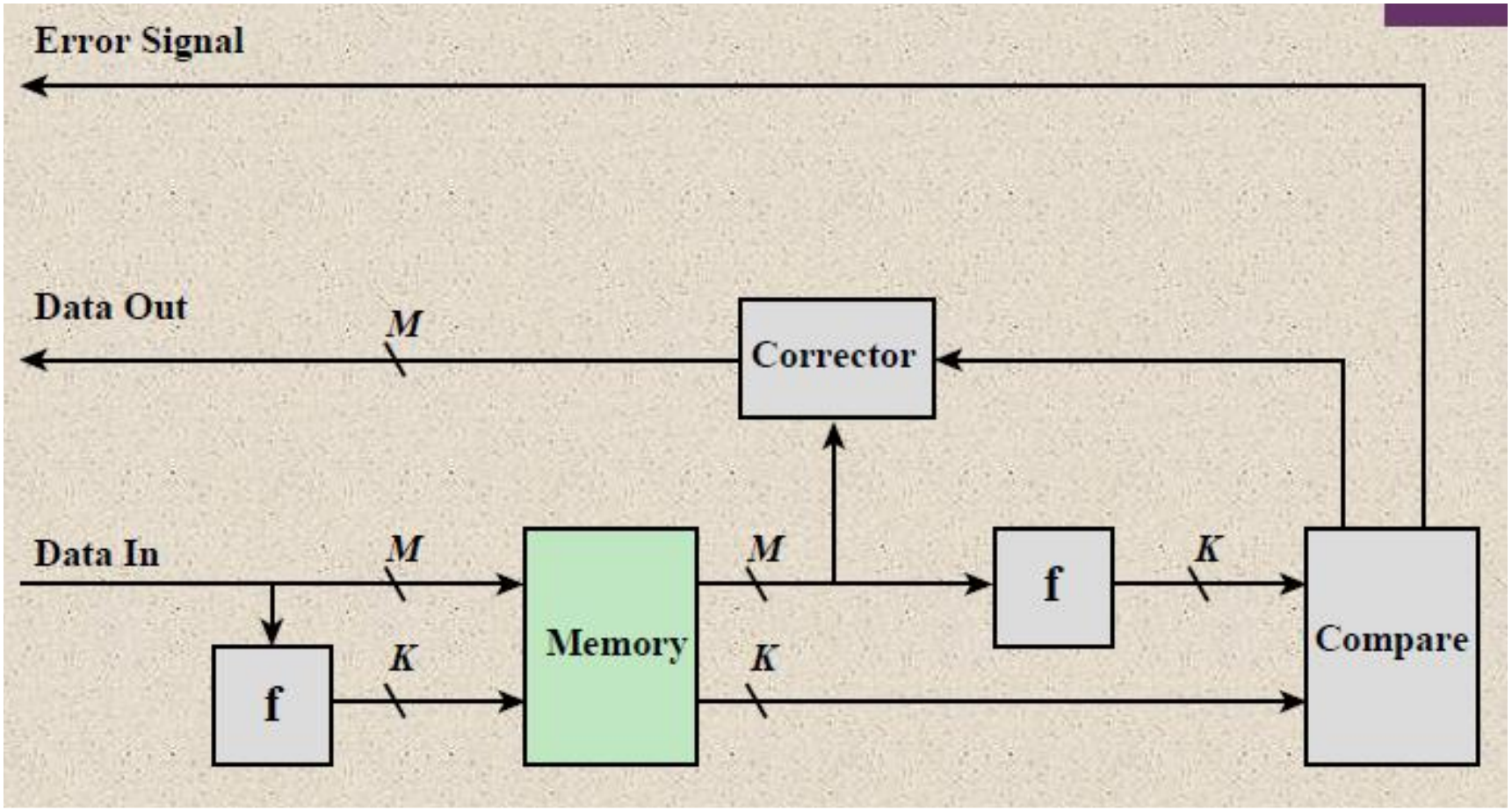


Figure 5.3 Typical 16 Megabit DRAM (4M × 4)

Error Correction

- Hard Failure
  - Permanent physical defect
  - Memory cell or cells affected cannot reliably store data but become stuck at 0 or 1 or switch erratically between 0 and 1
  - Can be caused by:
    - Harsh environmental abuse
    - Manufacturing defects
    - Wear
- Soft Error
  - Random, non-destructive event that alters the contents of one or more memory cells
  - No permanent damage to memory
  - Can be caused by:
    - Power supply problems
    - Alpha particles


Error Signal
Data Out
M
Corrector
Data In
M
M
f
K
Compare
Memory
f
K
K

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1

$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$

$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$

$C4 = D2 \oplus D3 \oplus D4 \oplus D8$

$C8 = D5 \oplus D6 \oplus D7 \oplus D8$

Math problems

- For the 8-bit word 00101011, calculate the check bits. Suppose when the word is read from memory, the check bits are calculated to be 0010.What is the data word that was read from memory?

- A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is 010011111000
- A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is 110011011011 (show the procedure)?

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1

$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$

$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$

$C4 = D2 \oplus D3 \oplus D4 \oplus D8$

$C8 = D5 \oplus D6 \oplus D7 \oplus D8$

For the 8-bit word 00101011, calculate the check bits. Suppose when the word is read from memory, the check bits are calculated to be 0010.What is the data word that was read from memory?

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8 0	D7 0	D6 1	D5 0		D4 1	D3 0	D2 1		D1 1		
Check Bit					C8				C4		C2	C1

$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$

$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$

$C4 = D2 \oplus D3 \oplus D4 \oplus D8$

$C8 = D5 \oplus D6 \oplus D7 \oplus D8$

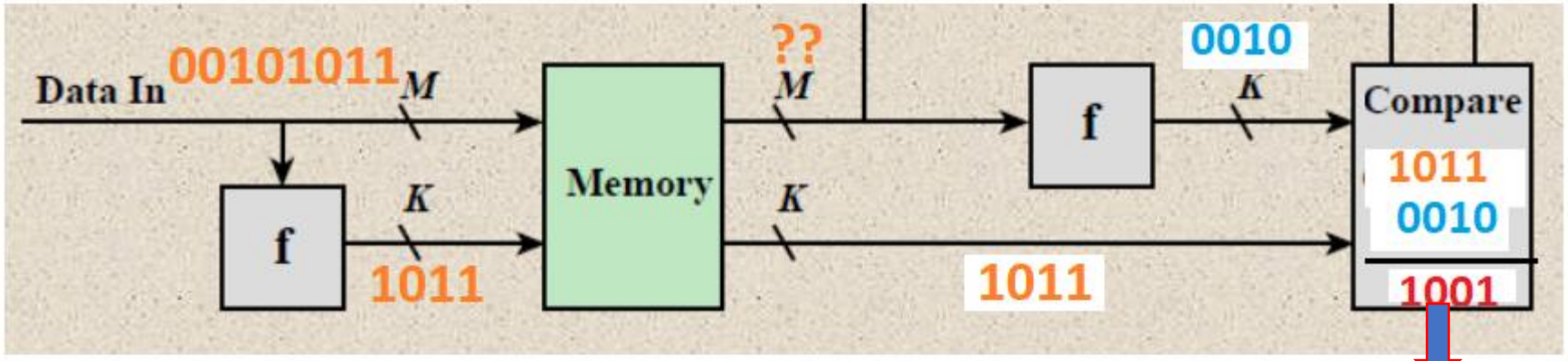
C1= 1

C2= 1

C4= 0

C8= 1

solution

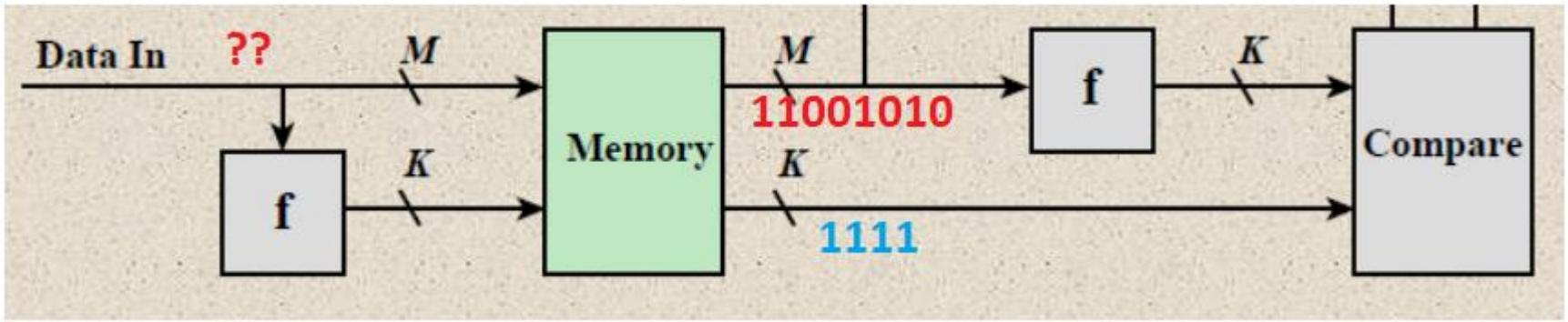


Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8 0	D7 0	D6 1	D5 0		D4 1	D3 0	D2 1		D1 1		
Check Bit					C8				C4		C2	C1

Data Read	D8 0	D7 0	D6 1	D5 1		D4 1	D3 0	D2 1		D1 1		

A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is 110011011011 (show the procedure)?

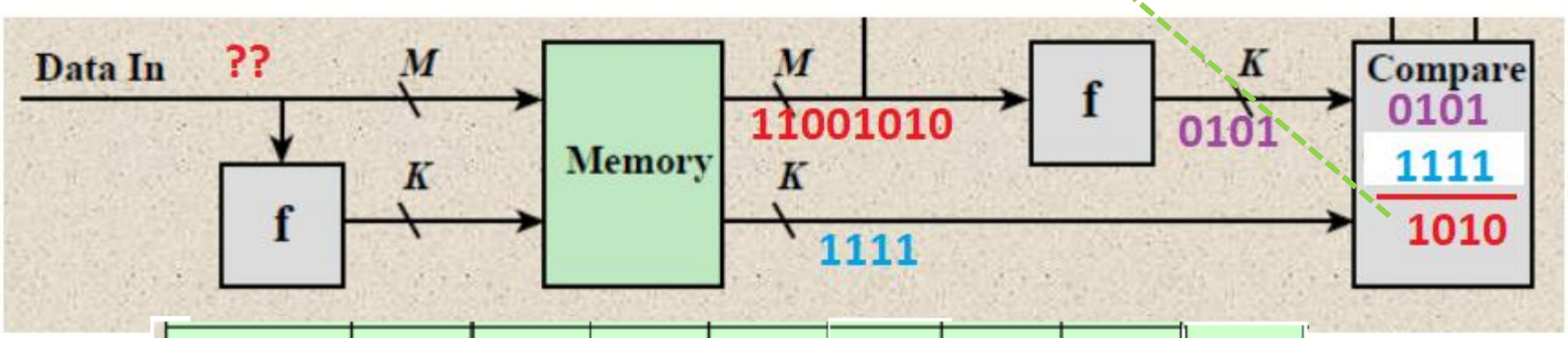
Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8 1	D7 1	D6 0	D5 0	1	D4 1	D3 0	D2 1	1	D1 0	1	1
Check Bit					C8				C4		C2	C1



Calculation checker bits from fetched data bits:

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8 1	D7 1	D6 0	D5 0	1	D4 1	D3 0	D2 1	1	D1 0	1	1
Check Bit					C8				C4		C2	C1

- $C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 1$
- $C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$
- $C4 = D2 \oplus D3 \oplus D4 \oplus D8 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$
- $C8 = D5 \oplus D6 \oplus D7 \oplus D8 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$



Data Bit	D8 1	D7 1	D6 1	D5 0	D4 1	D3 0	D2 1	D1 0