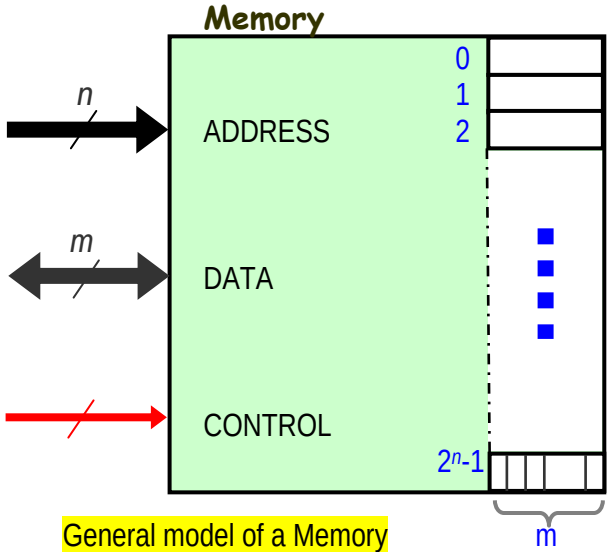
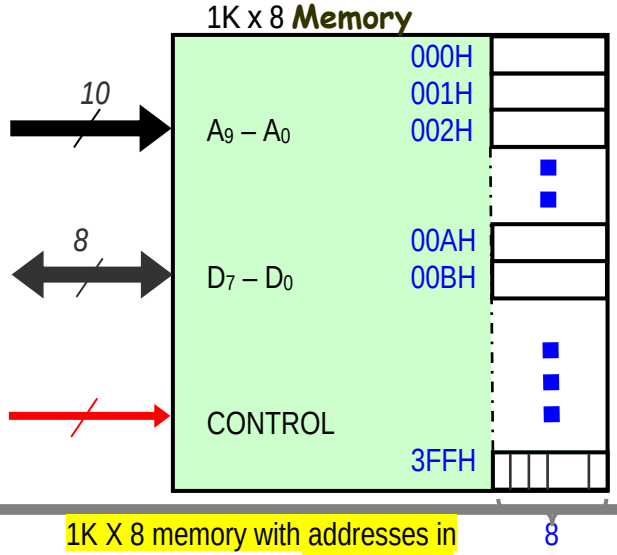
Memory

Devices

Memories ...



General model of a Memory

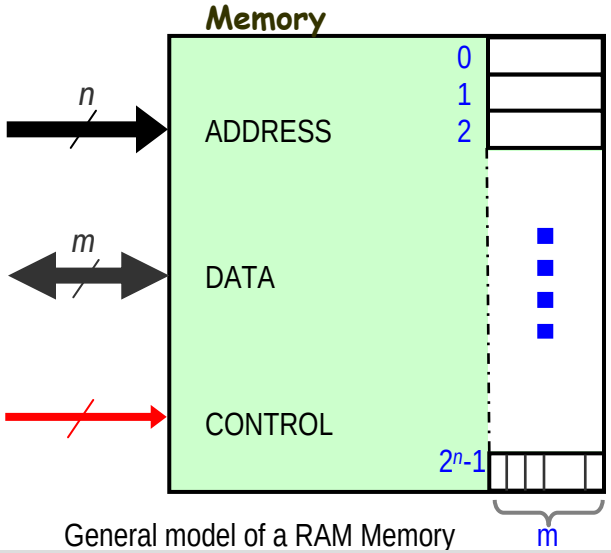


1K X 8 memory with addresses in Hexadecimal

- A memory unit has a number of locations where data can be stored.
- Generally, data can be read from or written into memory locations. The operation to be performed is specified by the *control* signal.
- Each memory location has a unique address associated with it which is specified by the signals coming in on the address lines.
- If *n* address lines are available, then 2^n memory locations, 0, 1, ..., 2^n-1 can be accessed directly.
- These memories have random access capability.
- If the unit has *m* data lines, then each location can store *m* bits of data.
- The capacity of a memory unit is the product of the number of memory locations & the number of bits per location, 2^n x *m*.
- Data lines can be bidirectional to permit read/write into memory locations. Such memories are called Read-Write Memories. (*RWM*)

1K = 2^{10} =1024 {0^{th} to 1023^{th} memory locations : 000H to 3FFH }
2K = 2^{11} =2048 {0^{th} to 2047^{th} memory locations : 000H to 7FFH }
4K = 2^{12} =4096 {0^{th} to 4095^{th} memory locations : 000H to FFFH }

Memories … RAMs (random access memory)



General model of a RAM Memory

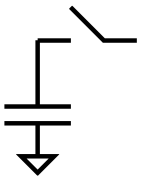
- Address lines & control lines are always unidirectional ⇔
- If data lines are also unidirectional ⇒, then data can only be read from memory. Such devices are called read only memories (ROMs).
- Access time is a measure of the memory's operating speed.
- Memories that are commonly called RAMs (random access memories) are really read/write memories (RWMs).

Static RAMs (SRAMS)

- basically Flip-Flops made with bipolar semiconductor technology
- tend to be very fast, but consume a lot of power. Hence are mainly used as cache memory in computers.
- Static memories store data until powered off.
- Bipolar devices are comparatively large, & hence packing density is not very high.

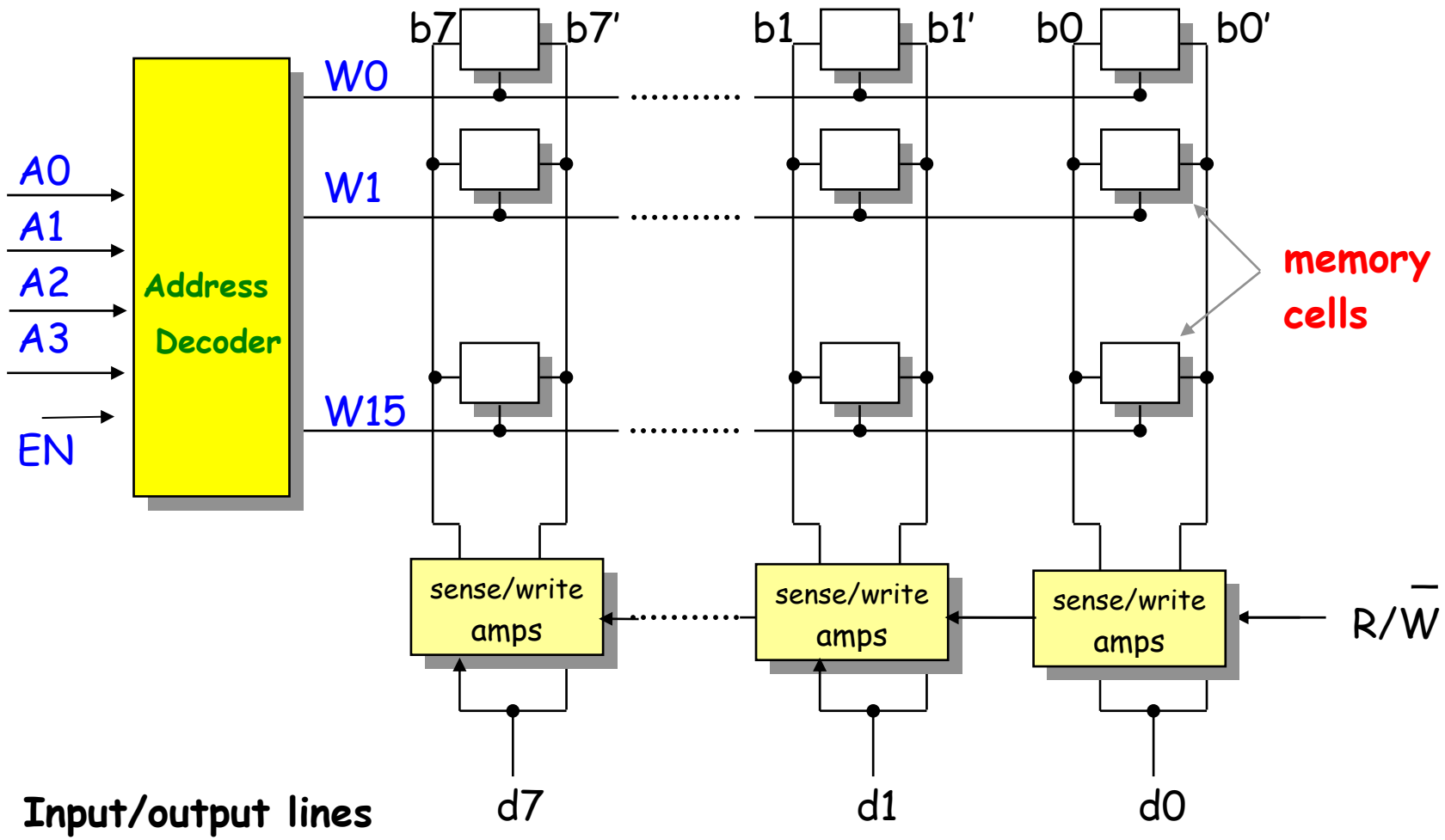
Dynamic RAMs (DRAMS)

- are basically capacitors that store charge and are made with MOS semiconductor technology.
- tend to lose charge (data) due to leakage even when powered on. Hence they have special circuitry to refresh the capacitors with charge. Hence the name dynamic.
- Packing density is very high and hence very large capacities can be obtained on a single chip.

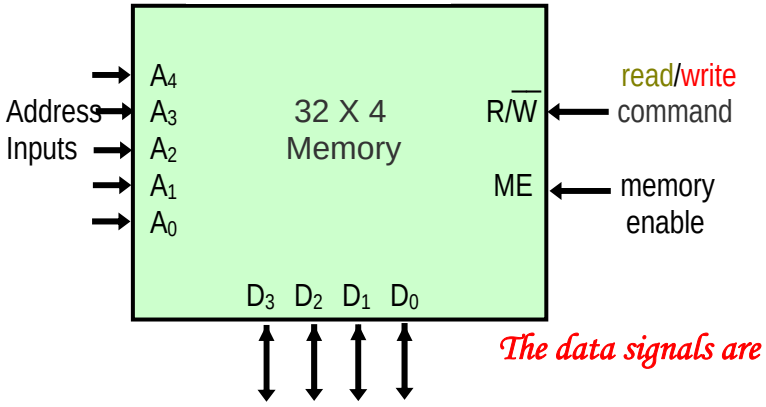


Both SRAMs and DRAMs are volatile memories, *i.e., they lose their contents when powered off.*
Non-volatile memories retain data even when powered off (ROM, CDROM, tapes, disk, etc.).

RAM configuration… 16 x 8 example…



Example: A 32 x 4 RWM (SRAM or DRAM)

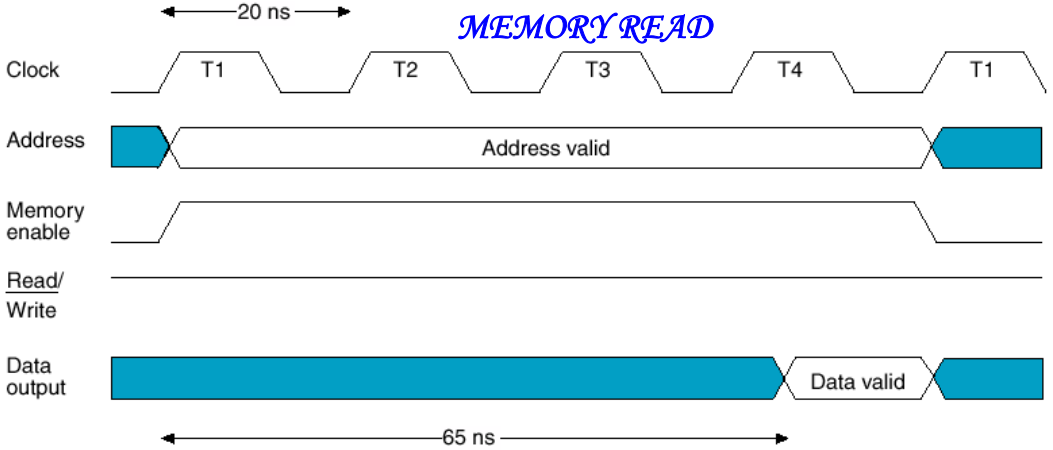


Virtual arrangement of memory cells into 32 four-bit words

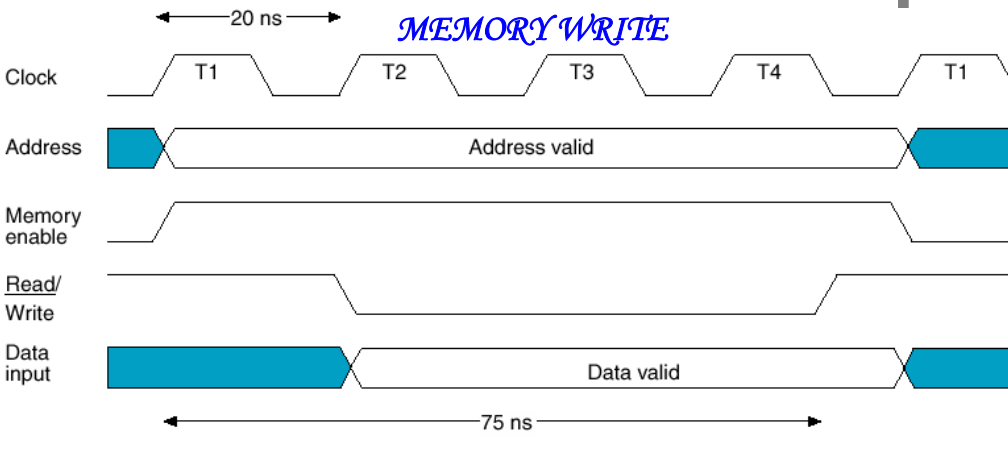
The data signals are bidirectional..

	ADDRESS hex	ADDRESS binary	Memory Cells			
0	00H	0 0 0 0 0	1	1	0	1
1	01H	0 0 0 0 1	0	1	0	1
2	02H	0 0 0 1 0	1	0	0	1
3	03H	0 0 0 1 1	1	1	1	0
4	04H	0 0 1 0 0	0	0	1	1
28	1CH	1 1 1 0 0	0	1	0	0
29	1DH	1 1 1 0 1	1	0	0	1
30	1EH	1 1 1 1 0	1	1	0	1
(2⁵-1) 31	1FH	1 1 1 1 1	0	1	1	0

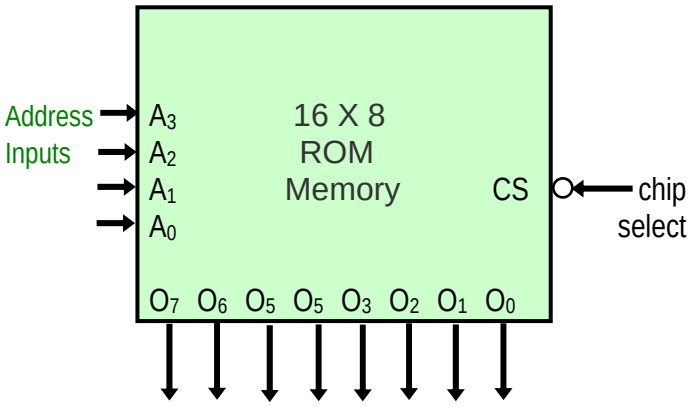
MEMORY READ



MEMORY WRITE



Memories … ROMs (read only memory)



- During normal operation data can be read from ROMs memory location, but cannot be written into.
- Principal advantage of ROM over RAM is that stored data will be retained after power off.
- Types of ROMs
- Different types of ROMs are available.
  - Normal read operations for all types is same.
  - Programming/Erasing/Re-programming is different

Mask Programmed ROM

- Programmed by manufacturer according to customer specs.
- A photographic negative called a mask is used to control electrical connections.
- Once programmed, no changes possible.
- A custom mask is produced for each design, hence economical only if manufactured in large volumes.

Field Programmable ROM PROM

- User programmable. Has fusible links (like PLA & PALs) which can be broken (intact fuse = 1, blown fuse = 0).
- Can be programmed only once. Done by commercially available programming devices.
- Suitable for low volume applications.

Erasable Programmable ROM EPROM

- Can be erased & re-programmed as many times as desired.
- Erasing takes place by exposing EPROM to UV light. 15-30 min to erase all data.
- can be programmed by commercial programmers.
- Ideal for prototype implementation

Electrically Erasable PROM EEPROM

- Unlike EPROMs: individual words can be electrically erased & re-programmed.
- Entire EEPROM can be erased in ~10ms
- Since erasing/re-programming is done electrically, need not remove chip from circuit to do so.
- Relatively fast (access time ~ 250 ns)

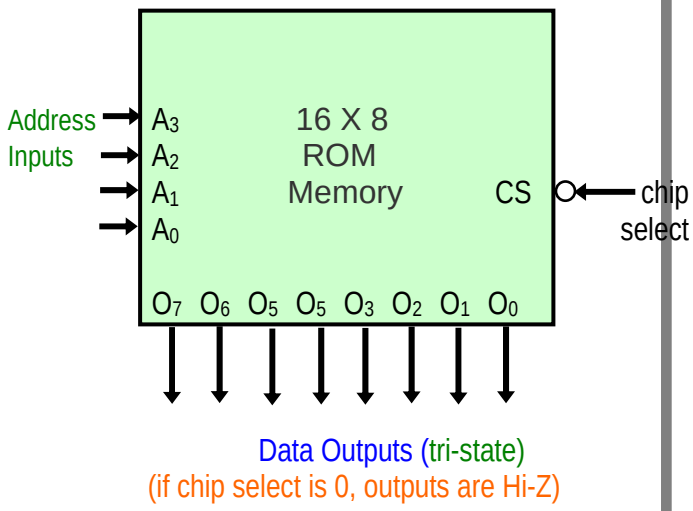
ROMs are non-volatile memories, *i.e., they don't lose their contents when powered off.*

ROM example…

A 16 x 8 ROM:

	Address				Data							
Word	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
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

	Address	Data
Word	A3 A2 A1 A0	D7-D0
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



ROM applications…

Microcomputer program storage (firmware): *ROM* **stores control program needed for computer operation e.g. electronic games, cash registers, etc.**

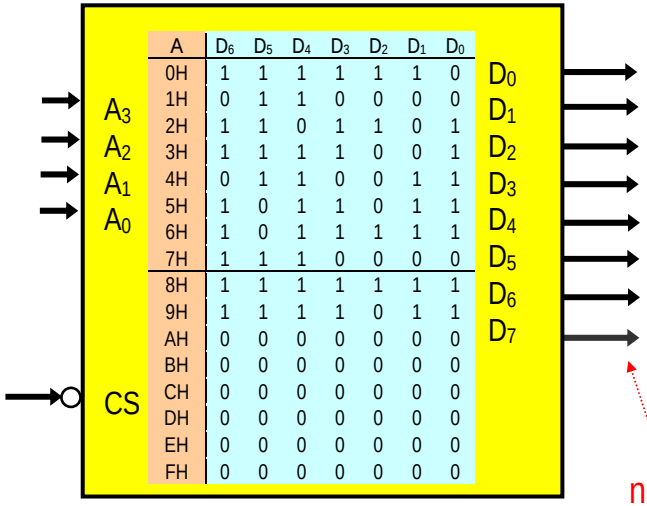
Bootstrap memory (BIOS): Most computers store their OS on hard disk, not *ROM*. **On power-up, CPU accesses** *ROM* **which has a small bootstrap program which:** *initializes system hardware, loads OS into* ***RAM*** *from hard disk, etc.*

Data conversion: Convert data from one type of code to another, e.g. conversion of BCD code to drive 7-segment displays.

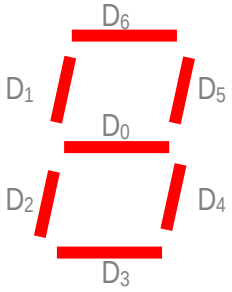
Alternatively, this example can be viewed as a systematic realization of a complex combinational circuit.

Example illustrates that ROMs can also be used for realizing logic functions.

PROMs are closely related to PLA/PALs have hardwired AND and programmable OR gates.



A	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0H	1	1	1	1	1	1	0
1H	0	1	1	0	0	0	0
2H	1	1	0	1	1	0	1
3H	1	1	1	1	0	0	1
4H	0	1	1	0	0	1	1
5H	1	0	1	1	0	1	1
6H	1	0	1	1	1	1	1
7H	1	1	1	0	0	0	0
8H	1	1	1	1	1	1	1
9H	1	1	1	1	0	1	1
AH	0	0	0	0	0	0	0
BH	0	0	0	0	0	0	0
CH	0	0	0	0	0	0	0
DH	0	0	0	0	0	0	0
EH	0	0	0	0	0	0	0
FH	0	0	0	0	0	0	0



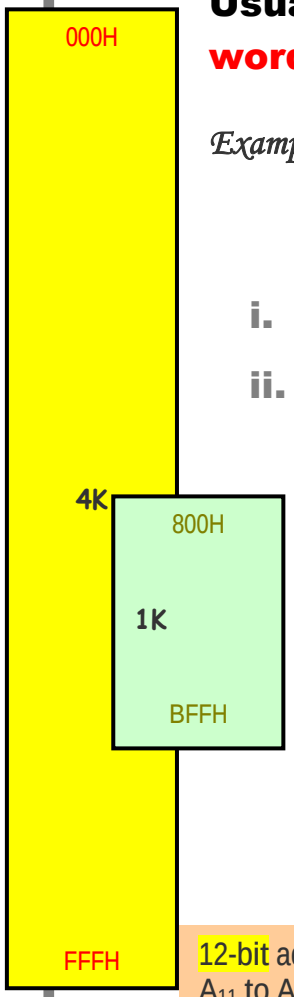
Expanding word size and capacity…

Usually several memory devices/chips have to be combined for a desired word size and/or capacity.

Example: **Design the following memory modules using only 1Kx4 *RAM* memory devices for use in 12-bit memory address system:** →

i. **A block of 1K x 8 *RAM* memory from address 800H**

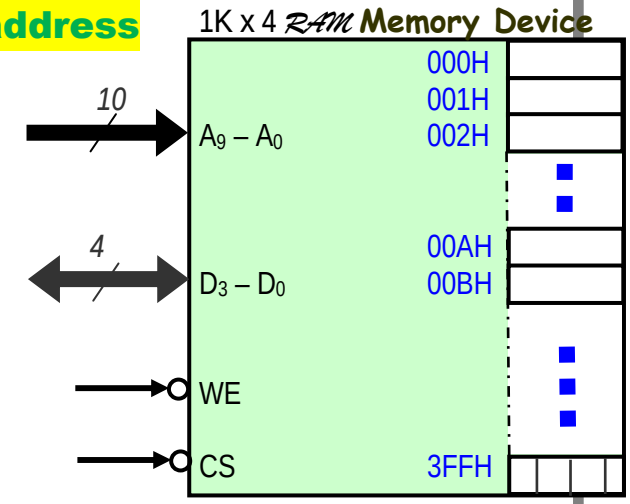
ii. **A block of 2K x 4 *RAM* memory from address 800H**



12-bit address memory system:
A_{11} to A_0 : 000H to FFFH

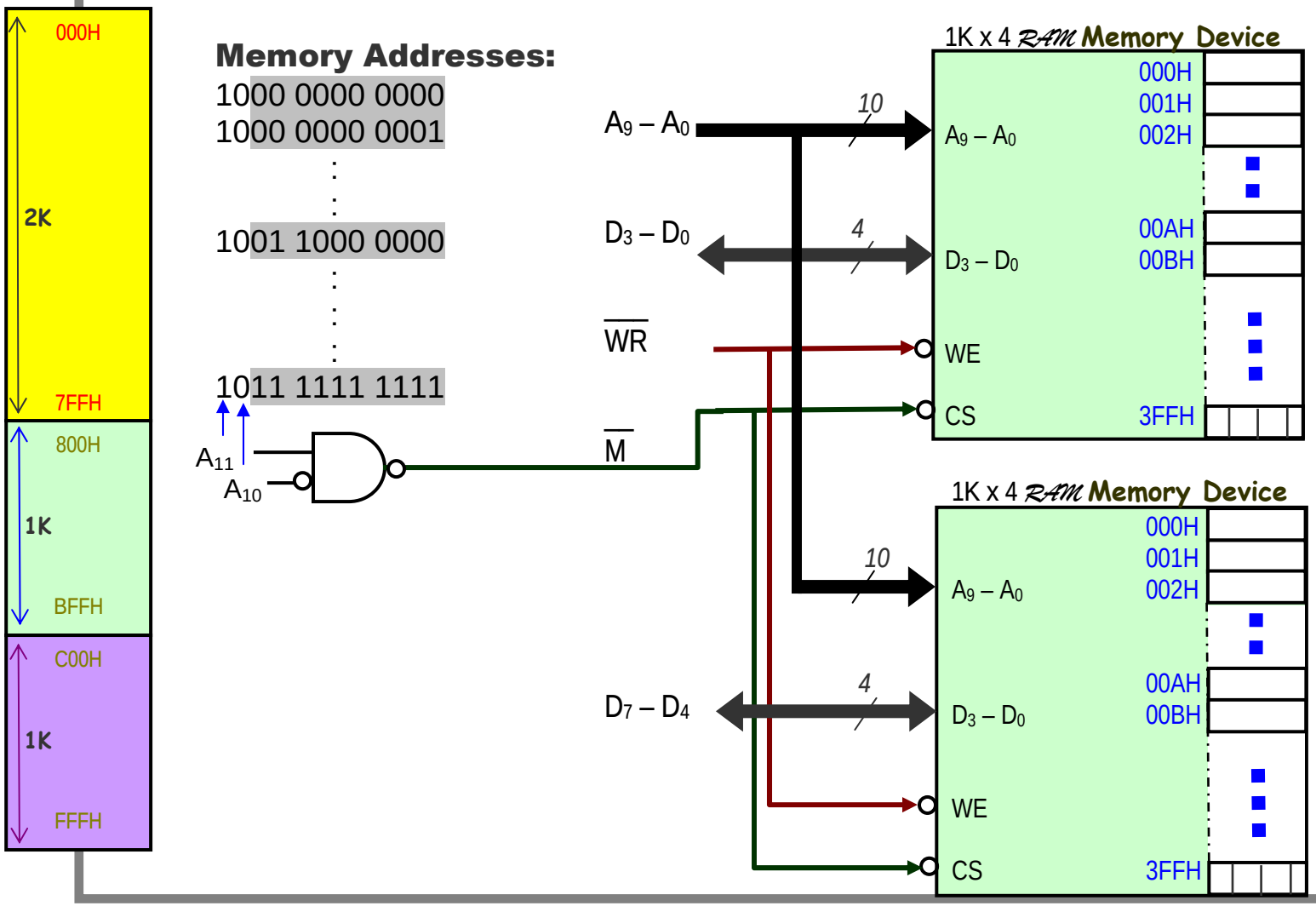
Truth table of 1K x 4 RAM:

Operation	CS	WE
Hi-Z outputs	H	X
Read	L	H
Write	L	L



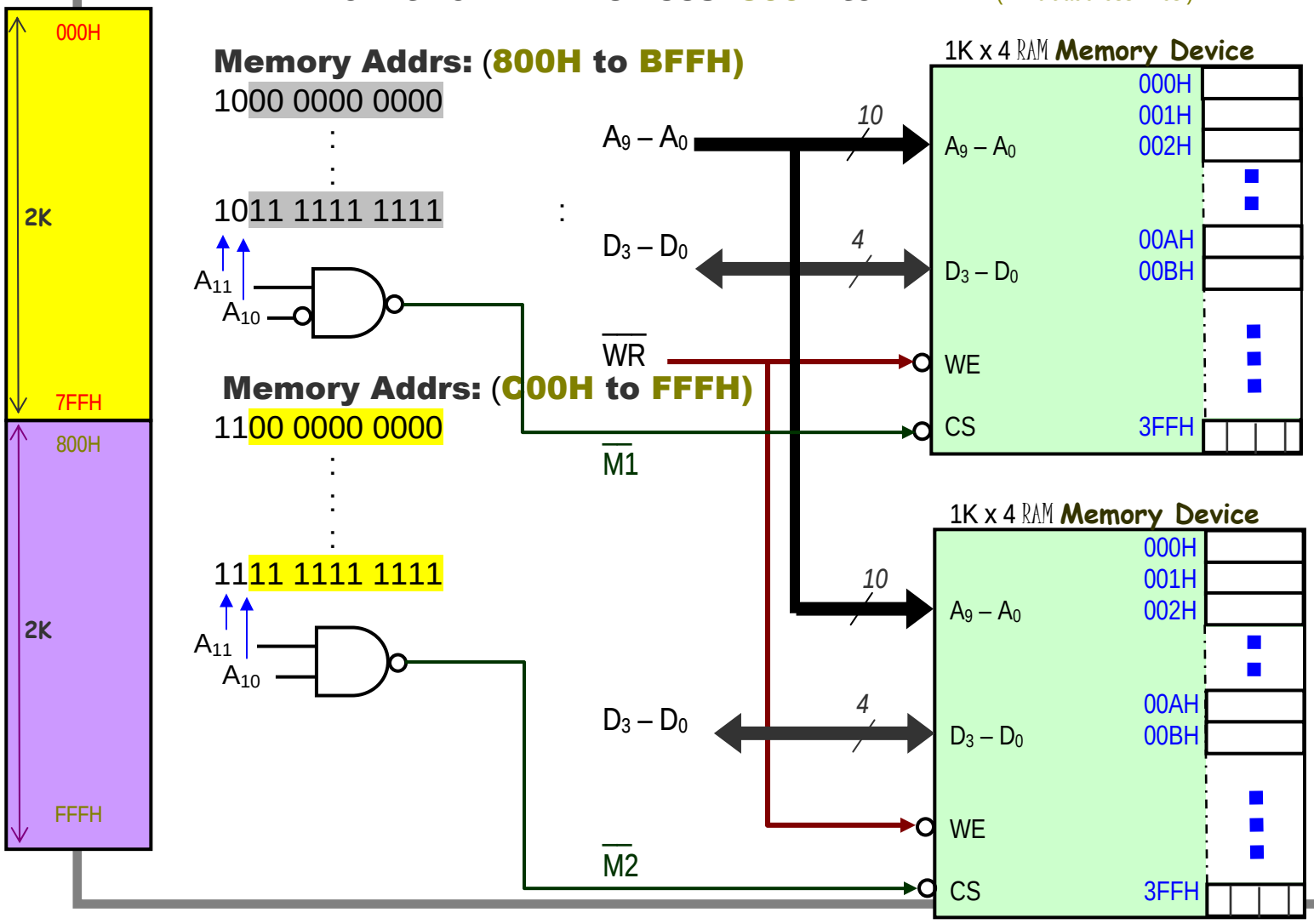
Expanding word size …1Kx8 from two 1Kx4 Devices

1Kx8 from two 1Kx4 Devices: 800H to BFFH (12-bit address lines)



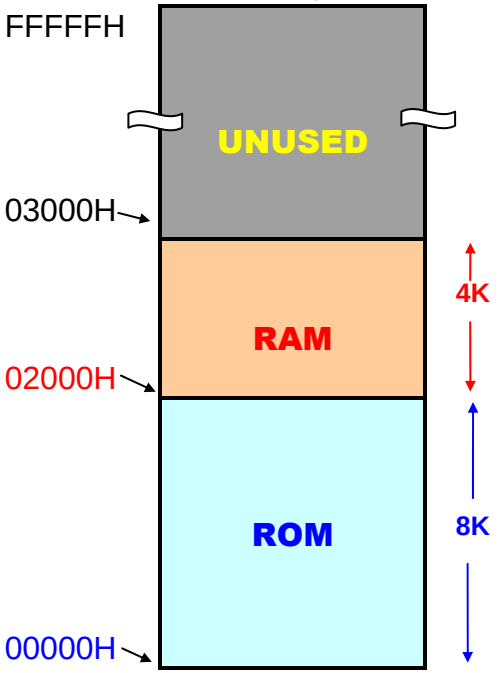
Expanding capacity… 2Kx4 from two 1Kx4 Devices

2Kx4 from two 1Kx4 Devices: 800H to FFFH (12-bit address lines)



Exhaustive Decoding… decode all address lines…

A 20-bit address system has a 8K ROM from 00000H & 4K RAM from 02000H.



memory map

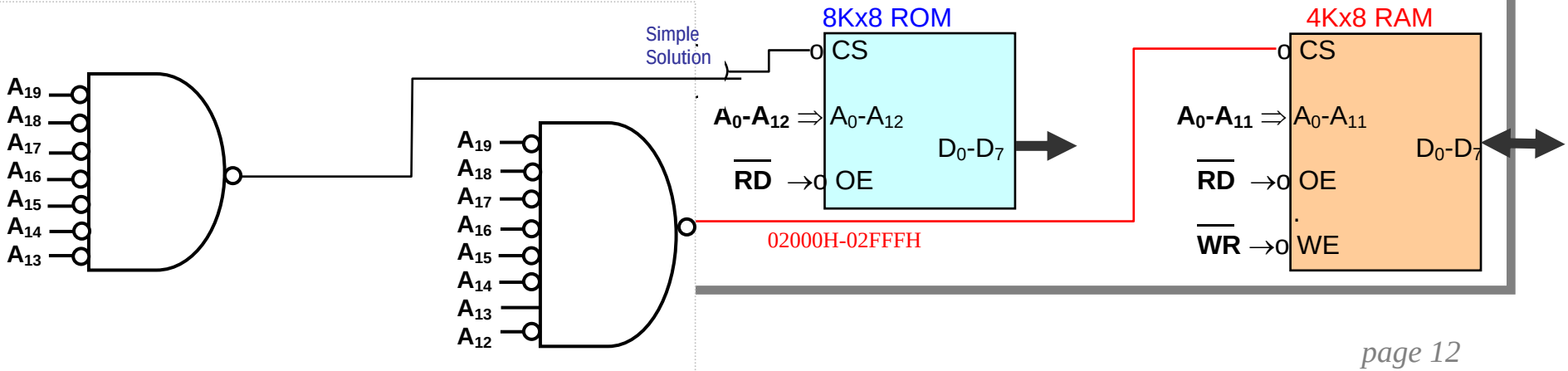
Step#1: Write memory map (complete address range) of each memory module in hex/binary form:

A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	(HEX)
0000	0000	0000	0000	0000	**ROM**	00000H
0000	0001	1111	1111	1111		01FFFH

A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	
0000	0010	0000	0000	0000	**RAM**	02000H
0000	0010	1111	1111	1111		02FFFH

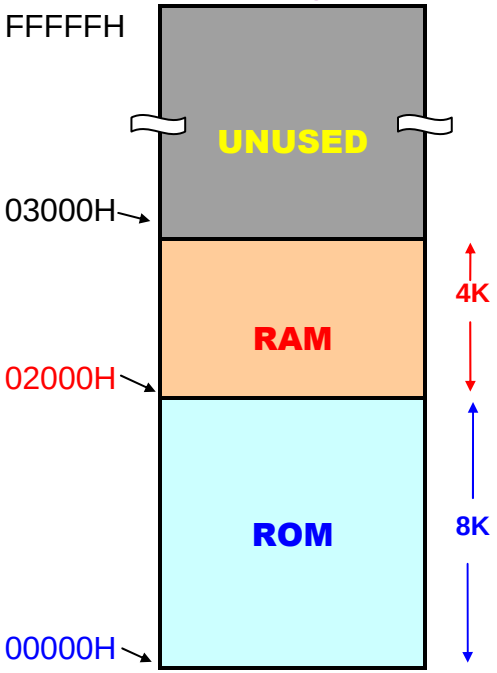
Step#2: Decode address bits to select/enable memory modules. All address bits of the memory system have to be decoded.

RAM/ROM Decoding:



Exhaustive Decoding… decode ALL address lines…

A 20-bit address system has a 8K ROM from 00000H & 4K RAM from 02000H.



memory map

Step#1: Write memory map (complete address range) of each memory module in hex/binary form:

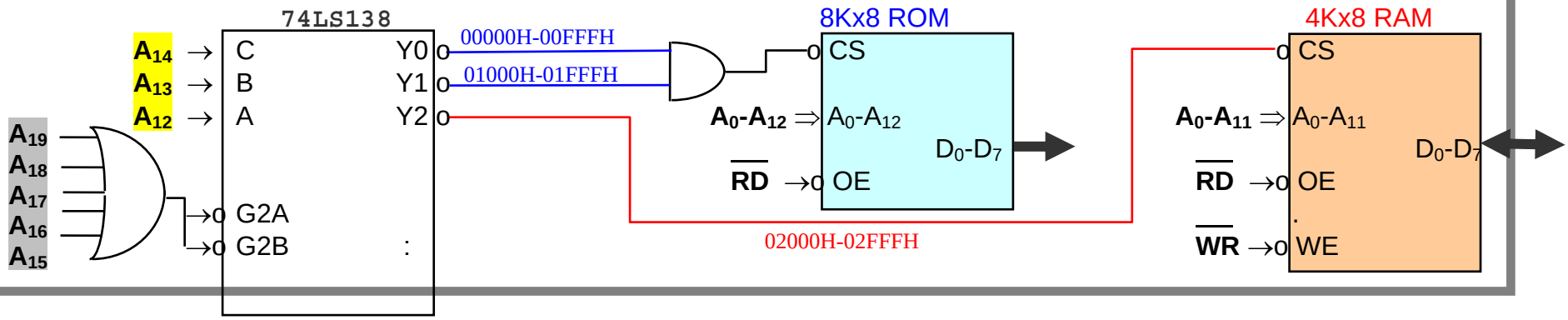
A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	(HEX)
0000	0000	0000	0000	0000	**ROM**	00000H
0000	0001	1111	1111	1111		01FFFH

A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	
0000	0010	0000	0000	0000	**RAM**	02000H
0000	0010	1111	1111	1111		02FFFH

Step#2: Decode address bits to select/enable memory modules. All address bits of the memory system have to be decoded.

What happens if not all addresses are decoded????

RAM/ROM Decoding: (Solution using decoders)



In-exhaustive Decoding… not all address lines are decoded

A 20-bit address system has a 8K ROM from 00000H & 4K RAM from 02000H.

memory map

Step#1: Write memory map (complete address range) of each memory module in hex/binary form:

A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	(HEX)
xxxx	0000	0000	0000	0000	**ROM**	X0000H
xxxx	0001	1111	1111	1111		X1FFFH
A19-A16	A15-A12	A11-A08	A07-A04	A03-A00	DEVICE	
xxxx	0010	0000	0000	0000	**RAM**	X2000H
xxxx	0010	1111	1111	1111		X2FFFH

: :

: :

12000H → RAM

10000H → ROM

02000H → RAM

00000H → ROM

Step#2: Decode address bits to select/enable memory modules. All address bits of the memory system have to be decoded.

Each memory location has 16 addresses…

RAM/ROM Decoding: (Solution using decoders)