

## LECTURE #14: RAM & ROM

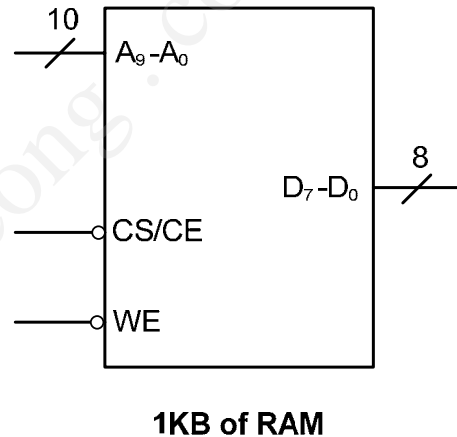
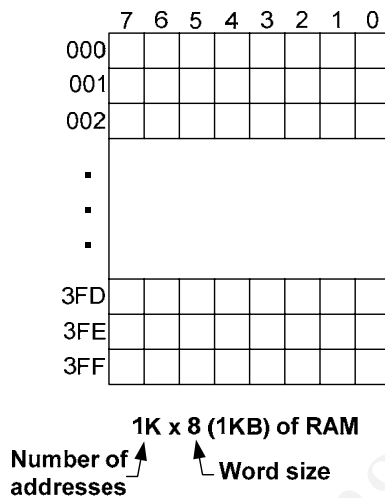
### EEL 3701: Digital Logic and Computer Systems

*Based on lecture notes by Dr. Eric M. Schwartz*

**RAM = “Random Access Memory”**

**Static RAM (SRAM):**

- SRAM is *volatile*, meaning that it only retains data while the power is on.
- Typical access times for CMOS based SRAM are around 10ns-100ns
  - Other types of SRAM types are available
- Memory can be viewed as a *vector of registers*
- Each register is indexed by an *address* (a number, usually in hexadecimal).



- Memory volume terminology:
  - 8 bits = 1 byte, 4 bits = 1 nibble,
  - $2^{10} = 1024 = \text{“1K”}$  (kilo-),  $2^{20} = \text{“1M”}$  (mega-),
  - $2^{30} = \text{“1G”}$  (giga-),  $2^{40} = \text{“1T”}$  (tera-)
- Memory requires  $n$  address lines, where  $2^n$  is the number of memory addresses.
- Usually, the number of output lines is the same as the word size.
- Data lines are usually bi-directional (time-multiplexed)
  - Serve as an input when the “Write” state is True
  - Serve as an output when the “Read” state is True

Example: A 1KB RAM (see figure above) requires:

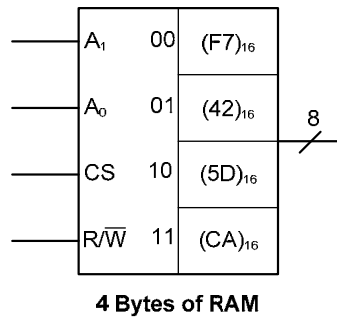
10 address lines ( $A_9-A_0$ )

8 data lines ( $D_7-D_0$ )

Some control lines:

Chip Select (CS), Chip Enable (CE), Write Enable (WE),  
Read (RD), Output Enable (OE), etc.

Example: A 4-Byte SRAM:

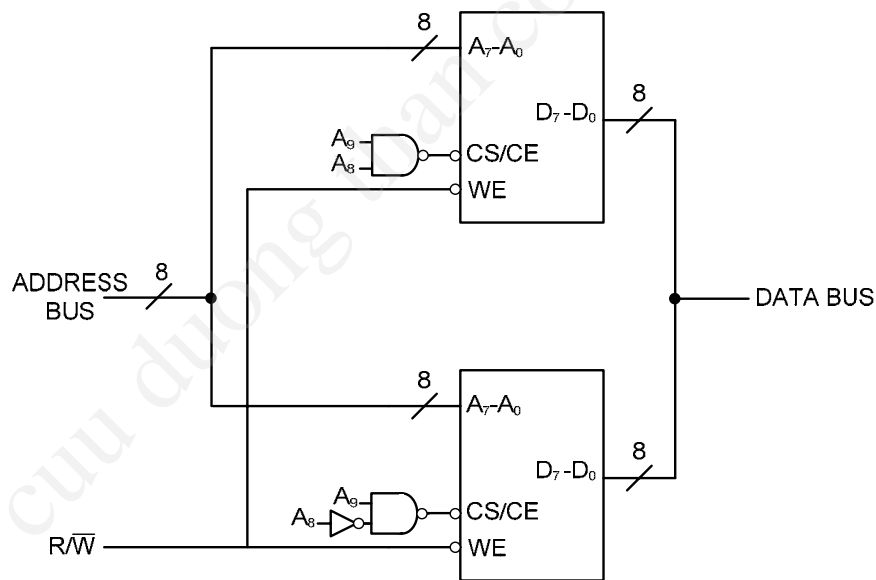


(The following assumes that CS is True.)

- What output do we get if we “Read”  $A_1 = 0$  and  $A_0 = 1$ ?  $(42)_{16}$
- How can we change  $(CA)_{16}$  to  $(F1)_{16}$ ?
  - Pull Read/Write low and introduce  $(F1)_{16}$  on the data bus.

Example: Construct a 512-byte memory module from two 256-byte SRAM's.

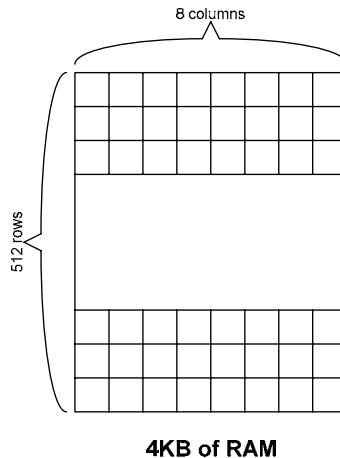
The module should be wired to interface with a microprocessor that has 10 address lines, 8 data bits, and the necessary control bits.



**Address Wiring Diagram Combining Two  
256-Byte Memories into a 512-Byte Memory**

- Where in memory are these located?  
Answer:  $\$300-\$3FF$  and  $\$200-\$2FF$   
(Note: Symbols: \$ => Hexadecimal, @ => Octal, % => Binary)
- How could we move them from “high” memory to “low” memory?
  - On the first, swap the NAND for an OR.
  - On the second, move the NOT gate from  $A_8$  to  $A_9$ .
- Could you mix around the address lines?
  - Yes, but the data wouldn't be contiguous with normal addressing.

- Sometimes RAM is organized like a matrix internally:



- How many address lines would this RAM require?  
12, 3 for the columns and 9 for the rows.
- Does the internal arrangement matter to the user?  
No. The inputs and outputs work the same.

#### Dynamic RAM (DRAM):

- DRAM is also *volatile* RAM.
- Uses RAS and CAS (Row/Column Address Select)
  - Always uses a matrix architecture.
- Row/Column addressing are time-multiplexed on the same address lines.

#### PRO:

- 1) Higher data density than SRAM (more data in less area = less cost)
  - 16MB ( $2^{24}$ ) and even larger
- 2) Faster access time than SRAM

#### CON:

- 1) Must be periodically refreshed (re-written) or the data will be lost.

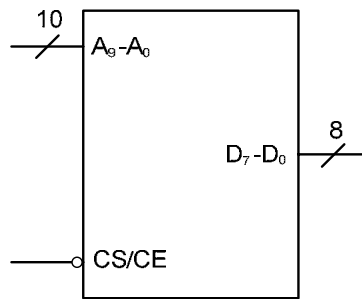
#### Read Only Memory (ROM):

- ROM is *non-volatile*, meaning that the data is saved even without power.
- ROM cannot be changed “on the fly.”

#### Types of ROM:

- ROM (masked ROM): Set by manufacturer. Cannot be modified.
- PROM (Programmable ROM): You program once by blowing fuses.
- EPROM (Erasable PROM): Reprogrammable. Erase using UV light.
- EEPROM (Electrically EPROM): Erased with voltage pulses.
- FLASH EEPROM: Newer and cheaper than standard EEPROM
  - Example: USB Flash Drives

- ROM is used for storing unchanging software
  - i.e. Bootstraps for startup, BIOS (Basic I/O Services), etc.



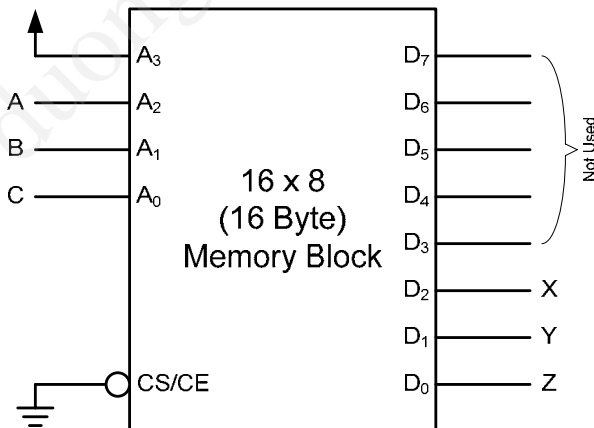
1KB of ROM

Note: There is no “Write Enable” input.

### Implementing a Truth Table with Memory:

Truth Table

A	B	C	X	Y	Z
0	0	0	1	1	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	0



- Inputs are connected to the address lines (i.e.  $A_2$ ,  $A_1$ , and  $A_0$ ).
  - These look up a byte in memory, whose values become the output.
- Extra address lines (i.e.  $A_3$ ) can be tied high or low.
  - This effects which part of the memory is used, but not the operations.
- Internally, the Truth Table output values are stored.
  - The outputs are the LSb's of the byte at the corresponding address.
  - Ex. Address 1000 should contain xxxxx110 (see Truth Table above).