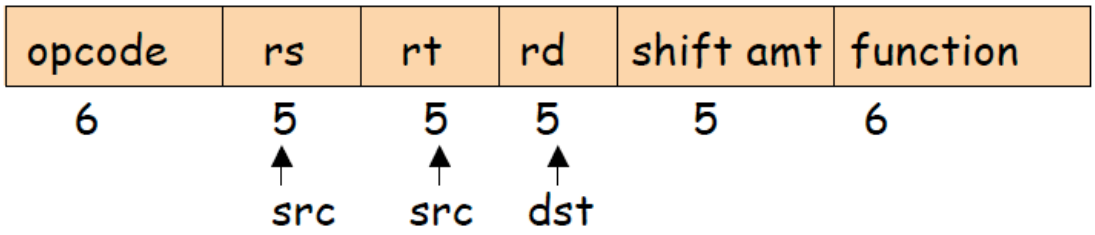
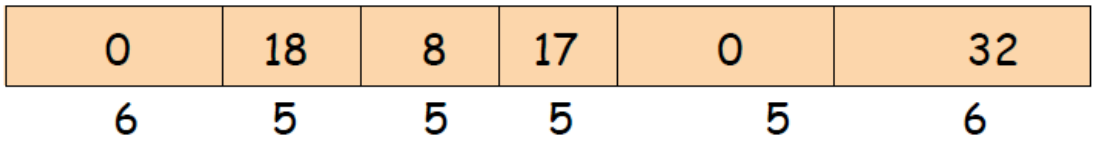
Selection of Registers

Recall the R-type MIPS instructions



add $s1, $s2, $t0 will be coded as



How are the registers selected from a bank of 32 registers (called register file)

Register file construction

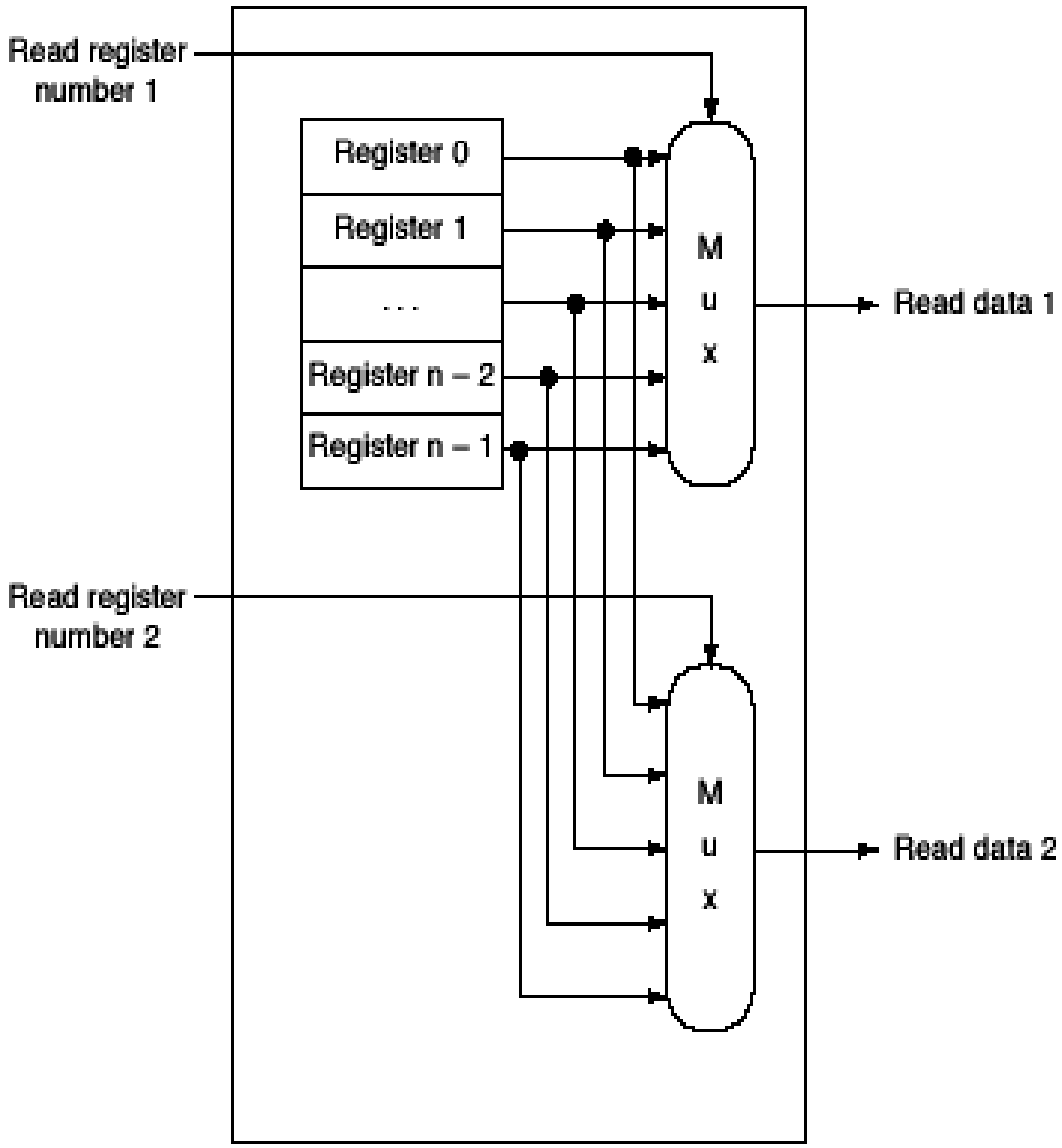


FIGURE B.8.8 The implementation of two read ports for a register file with *n* registers can be done with a pair of *n*-to-1 multiplexors each 32 bits wide. The register read number signal is used as the multiplexor selector signal. Figure B.8.9 shows how the write port is implemented.

Creating read ports

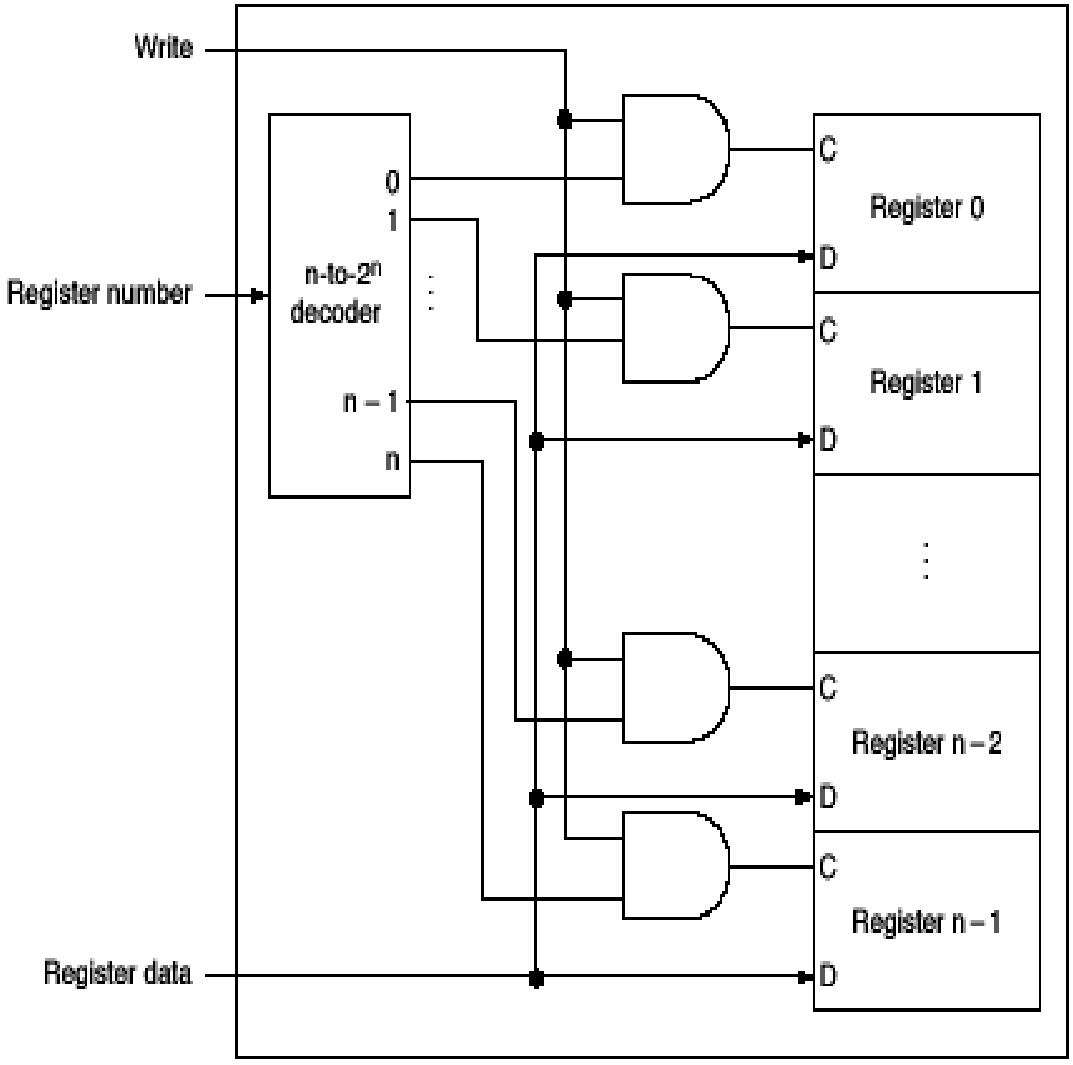
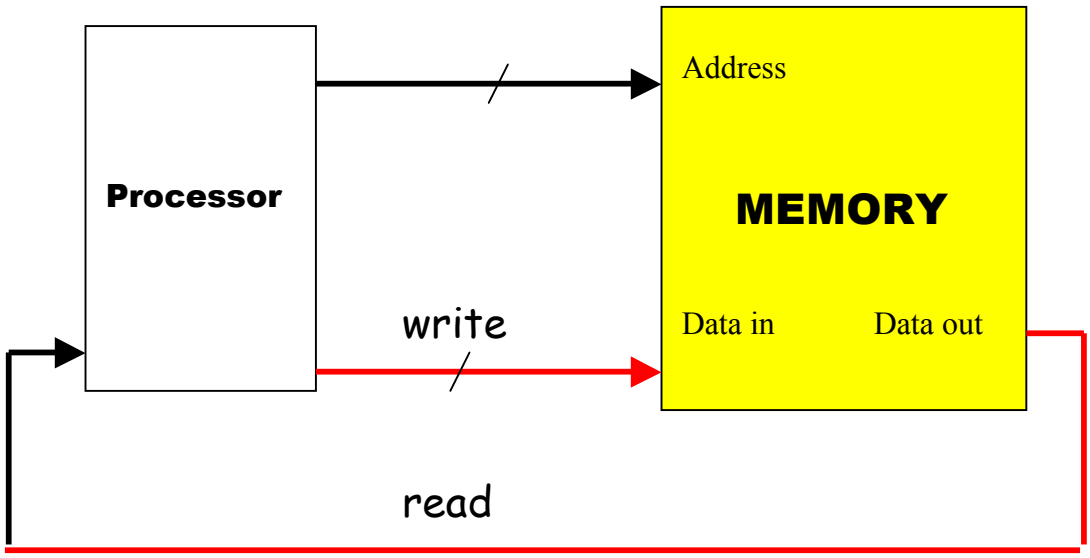
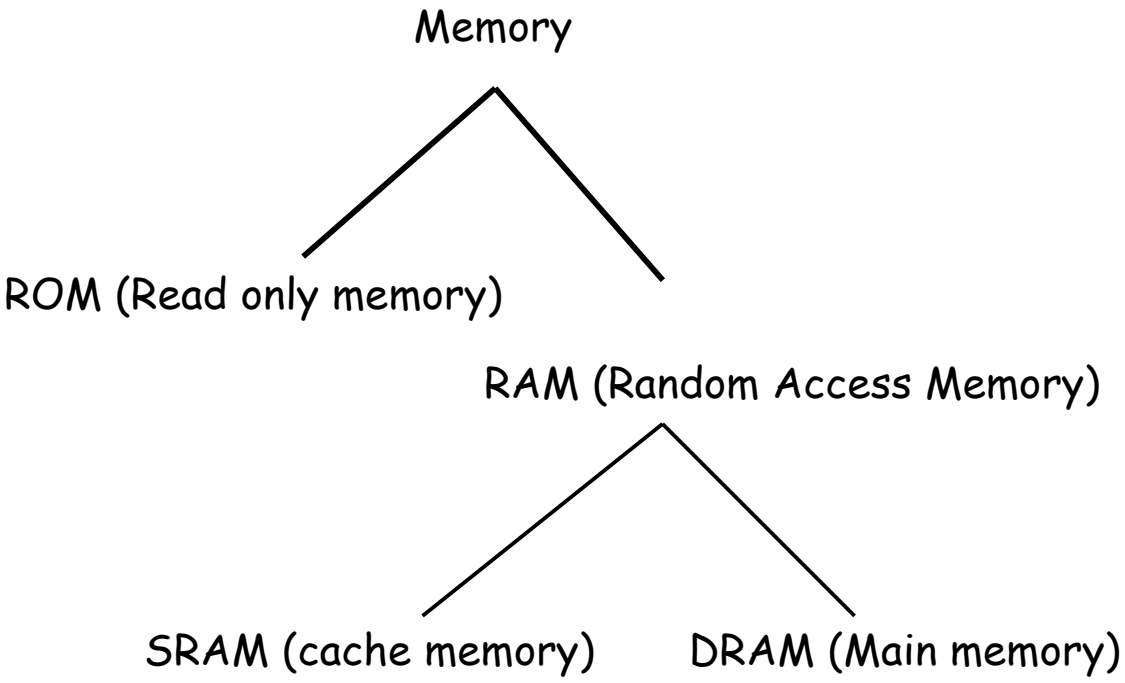


FIGURE B.8.9 The write port for a register file is implemented with a decoder that is used with the write signal to generate the *C* input to the registers. All three inputs (the register number, the data, and the write signal) will have set-up and hold-time constraints that ensure that the correct data is written into the register file.

Creating write port

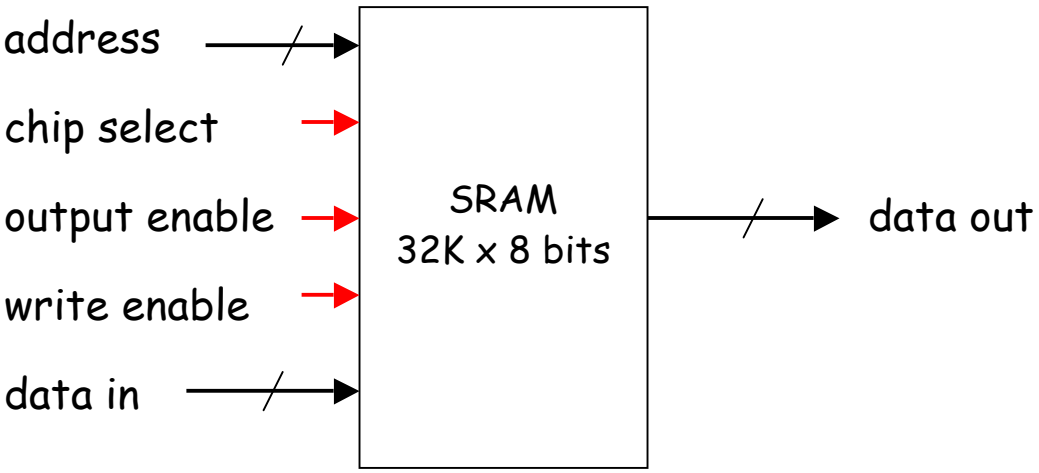
Main Memory


Processor
Address
MEMORY
Data in
Data out
write
read


Memory
ROM (Read only memory)
RAM (Random Access Memory)
SRAM (cache memory)
DRAM (Main memory)

Typical sizes of SRAM are

(32 or 64 or 128 or 256 or 512K) x (1 or 2 or 4 or 8 bits)



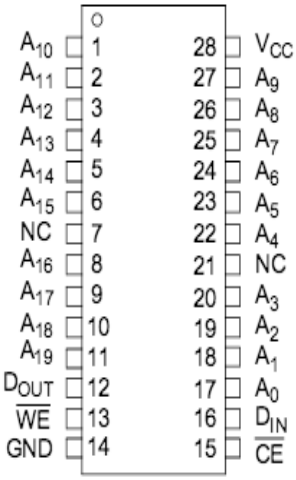
How many lines are there in **address**, **data in** and **data out**?

Example

CY7C107D is a commercially available (1M x 1 bit) SRAM

Pin Configuration

Figure 1. 28-pin SOJ pinout (Top View) [2]



Selection Guide

Description	CY7C107D-10 CY7C1007D-10	Unit
Maximum access time	10	ns
Maximum operating current	80	mA
Maximum CMOS standby current, I_{SB2}	3	mA

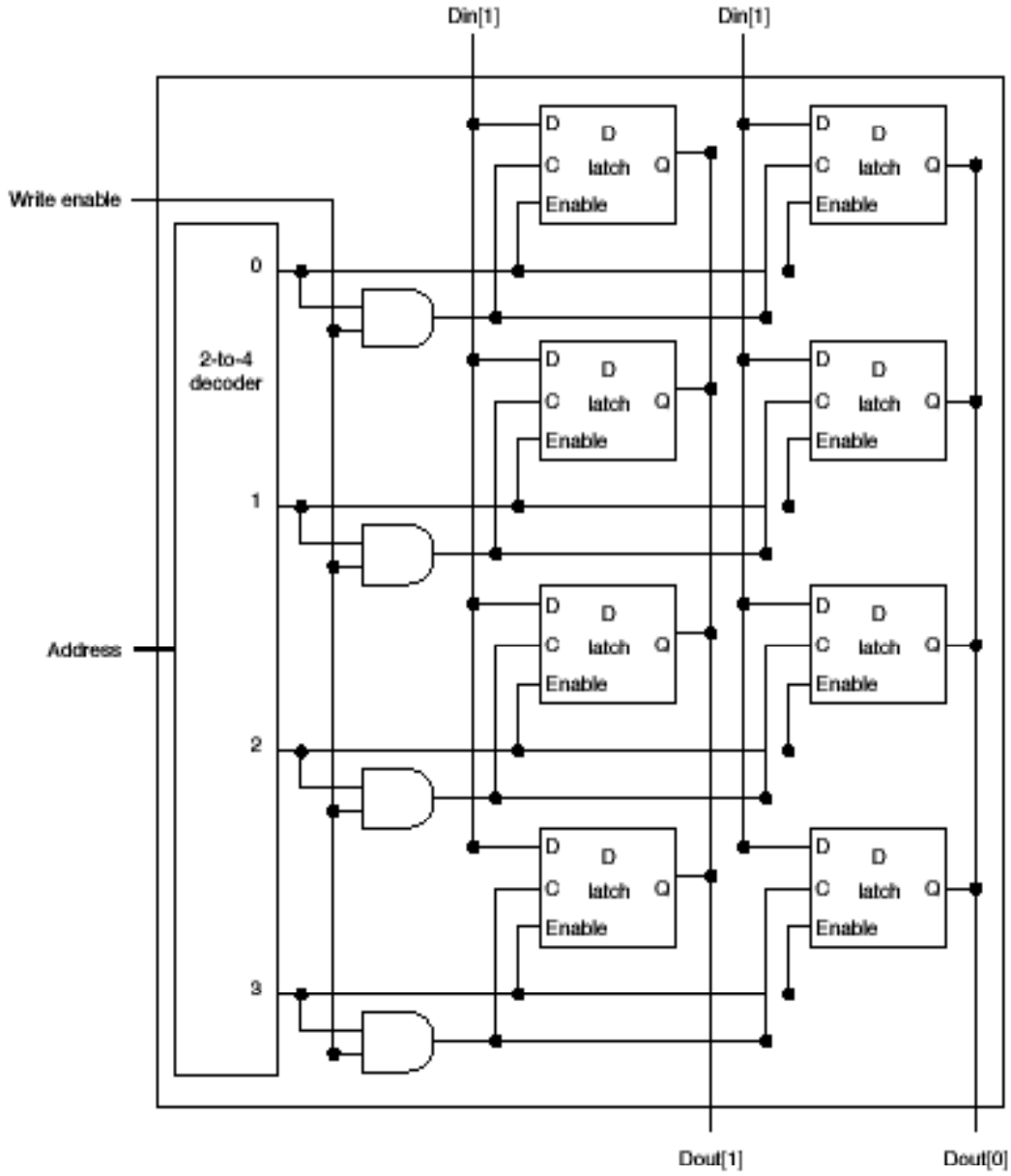
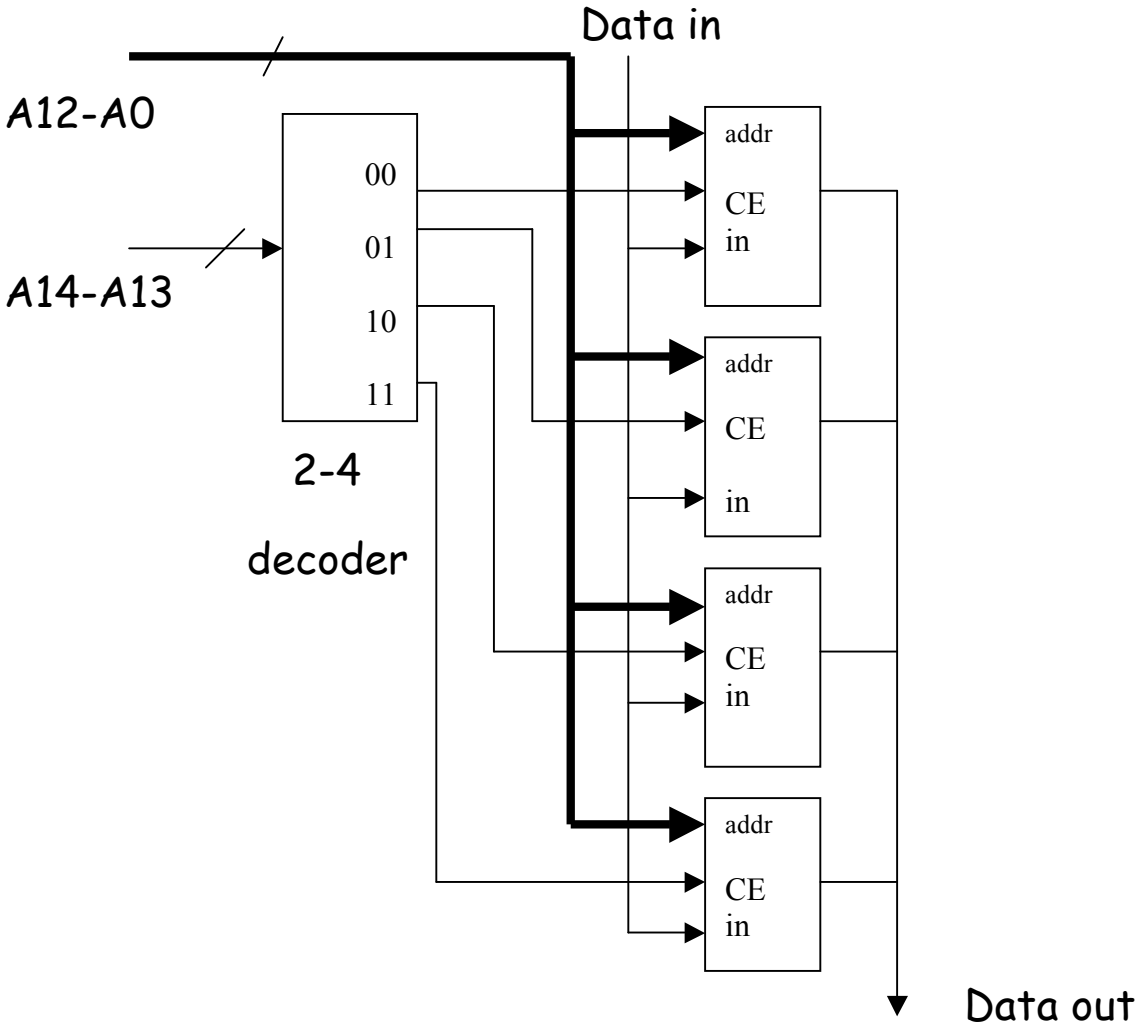


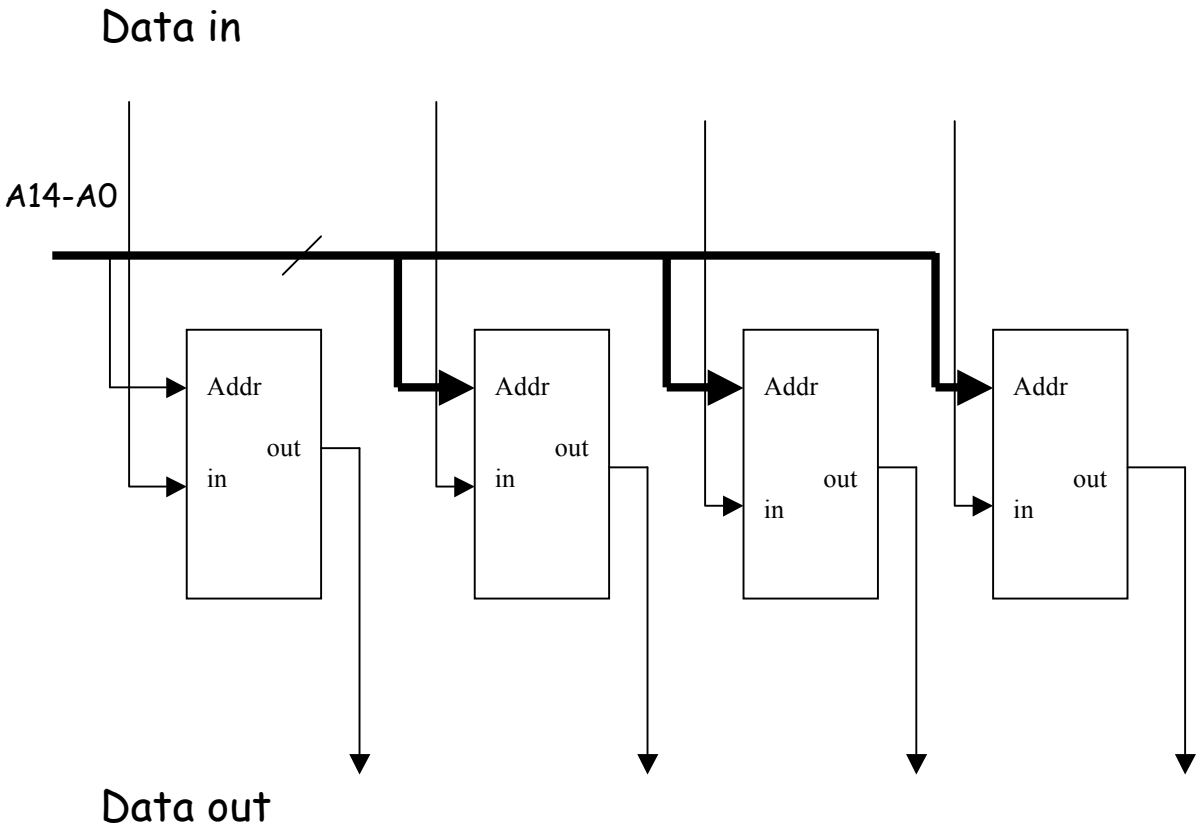
FIGURE B.9.3 The basic structure of a 4 ¥ 2 SRAM consists of a decoder that selects which pair of cells to activate. The activated cells use a three-state output connected to the vertical bit lines that supply the requested data. The address that selects the cell is sent on one of a set of horizontal address lines, called the word lines. For simplicity, the Output enable and Chip select signals have been omitted, but they could easily be added with a few AND gates.

32K x 1 bit RAM using 4 8K x 1 RAMs



For each chip, the write enable line is set to 1 during a write operation, and the output enable lines are set to 1 during a read operation.

32Kx 4 bit RAM using 32K x 1 bit RAMs

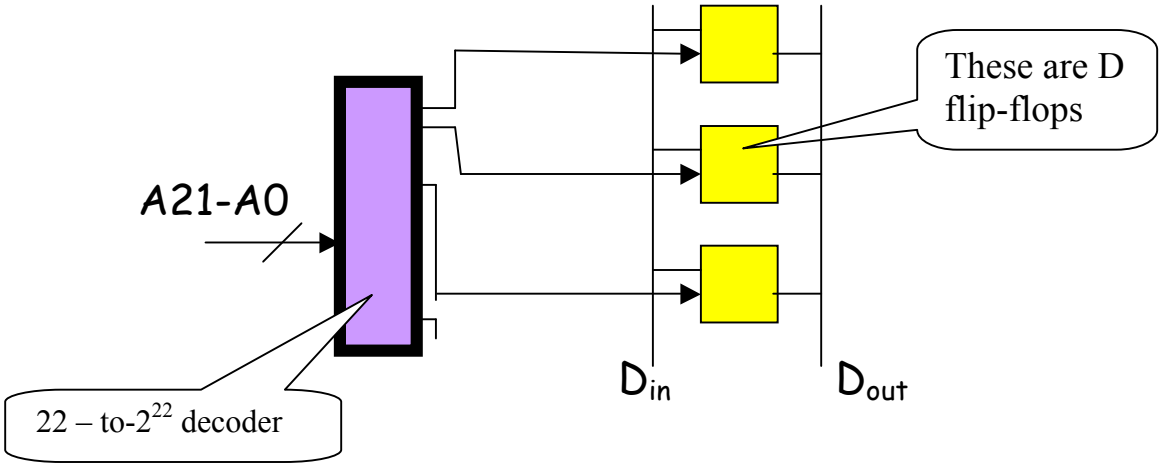


For each chip, the write enable line is set to 1 during a write operation, and the output enable lines are set to 1 during a read operation.

Inside RAM

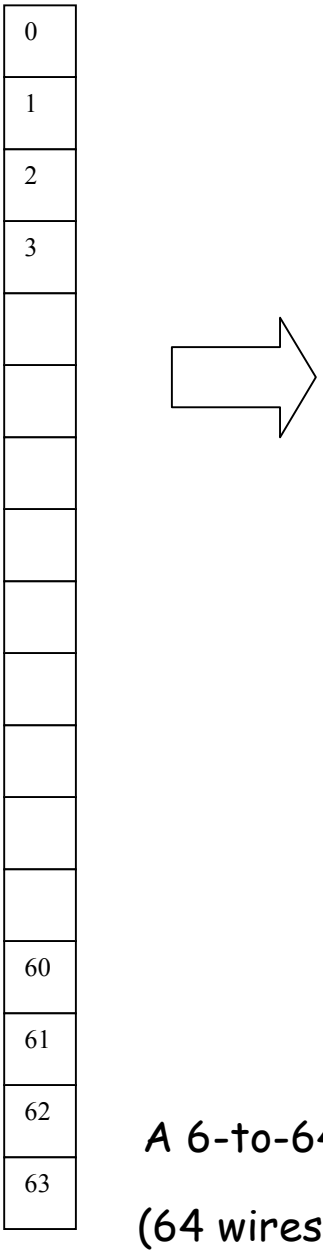
Two-level addressing

Consider a 4M x 1 bit RAM. It will apparently need a 22-to-4M decoder. 4 million output lines will make it complex! Complex! TwO-level addressing simplifies it.



In two-level addressing, the memory cells for any bit position are logically arranged as a two-dimensional array. 11 address lines are used to select a row, and 11 address lines are used to select a column. This reduces the number of output lines of the decoders from 4M to 2048 + 2048 = 4096 only.

Basic idea



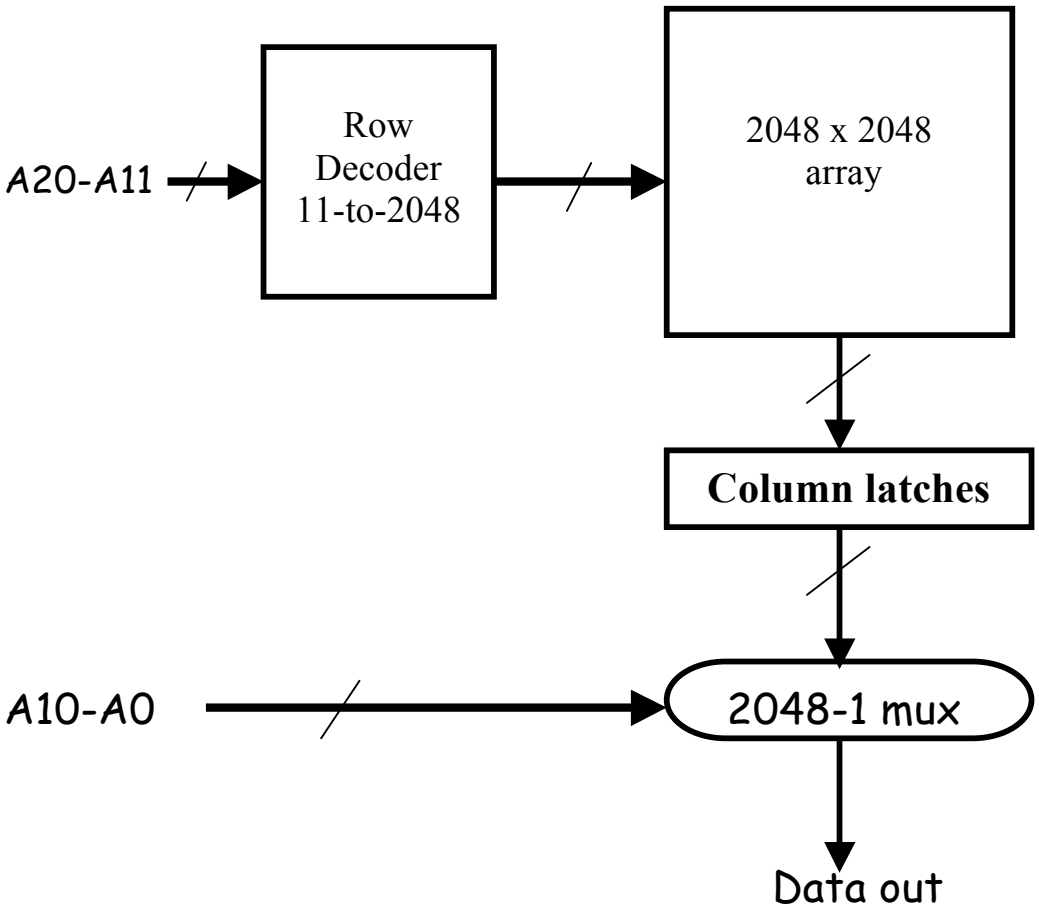
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

A 3-to-8 decoder for row
A 3-to-8 decoder for the column
(8+8=16 wires lead to the memory)

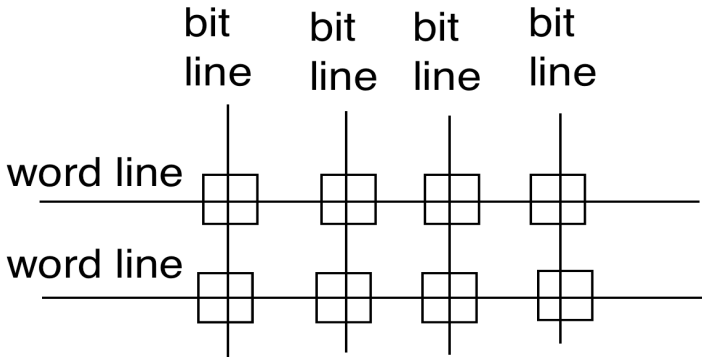
A 6-to-64 decoder will decode the address

(64 wires lead to the memory cells)

Scheme for two-level addressing

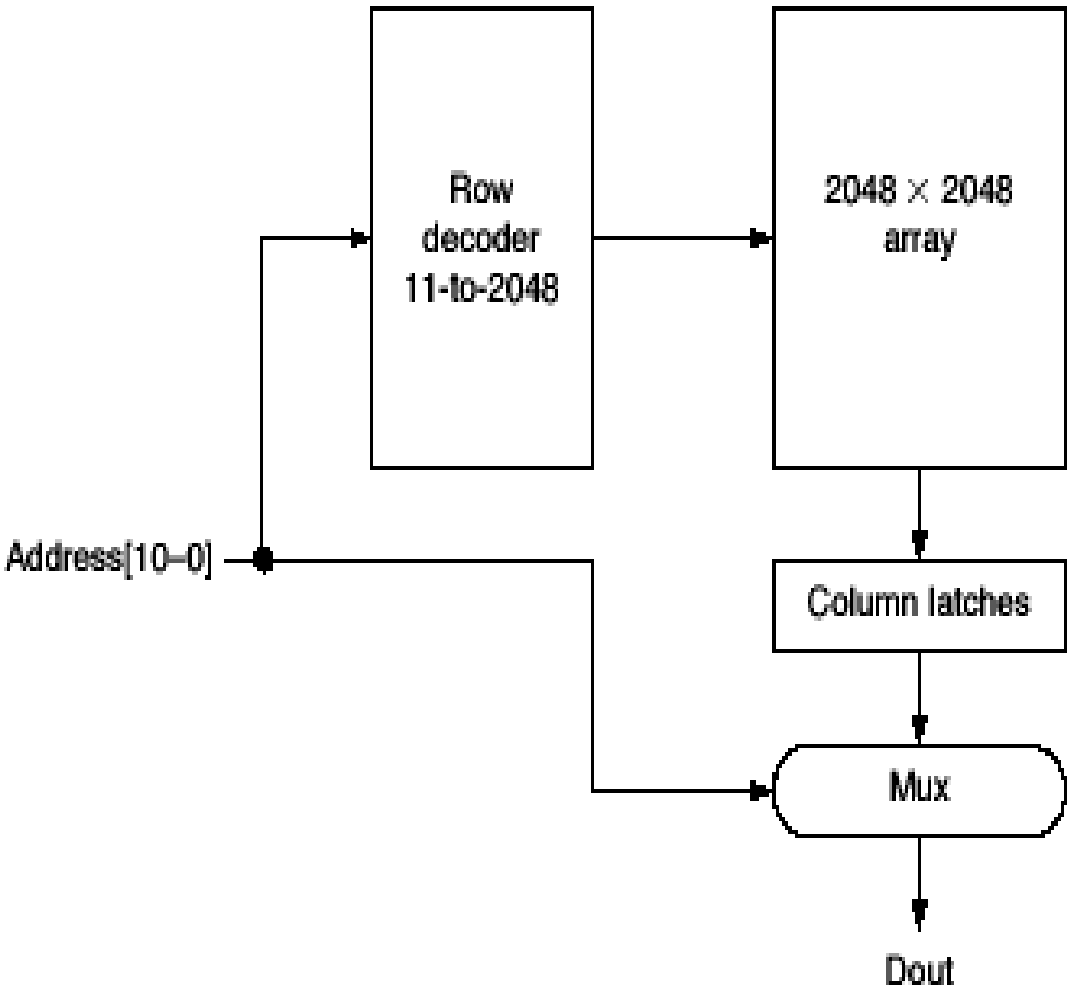


A 4M x 1 bit memory

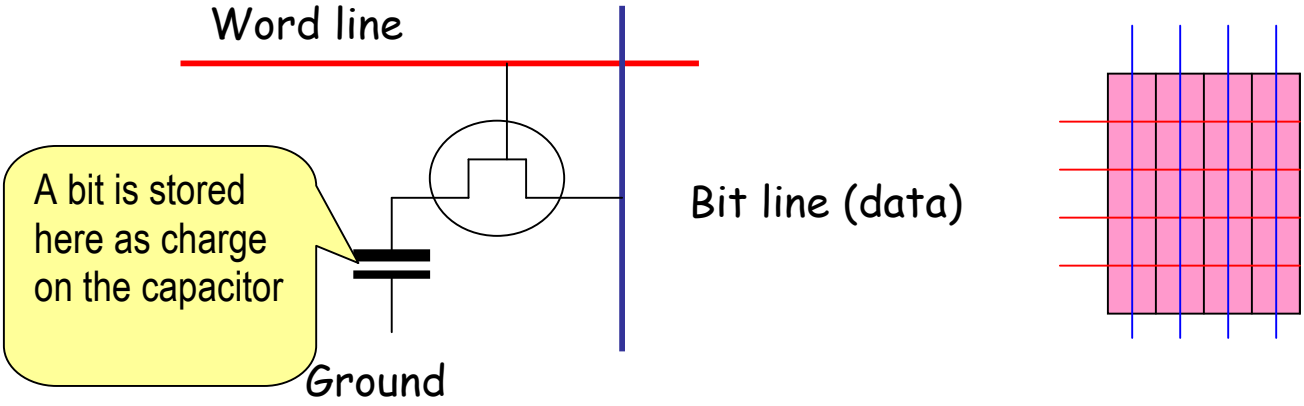


Modified scheme with shared address lines

To save pins, in 4M x1 bit DRAMs, 11 common lines are used for both A20-A11 and A10-A0. A pair of signals (RAS = Row Access Strobe and CAS = Column Access Strobe) notifies which part of the address is present at the input lines.



Dynamic RAM

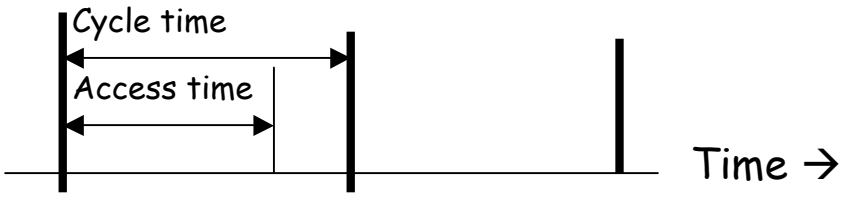


A "1" in the word line closes the switch, and the input data through the bit line charges the capacitor. During reading, the voltage on the capacitor is sensed through the bit line to determine if the content is a "1" or a "0".

To safeguard the leakage of the capacitor, it is periodically refreshed (An entire row in one cycle).

Very high packing density, but slow (45-65 ns access time) DRAM is a good choice for the main memory.

Cycle time



Smallest time between consecutive R/W operations.

Speeding up DRAMs

Page mode DRAM

Row address does not change – only the column address changes. Reduces the access time (25 ns) within a row.

SDRAM (Synchronous DRAM)

Series of bits from a row are automatically transferred in a burst – explicit column addresses are not specified. This saves times and leads to faster memory access.

ROM and PLA and FPGA

ROM = Read Only Memory

PLA = Programmable Logic Array

FPGA = Field Programmable Gate Array

Here a ROM that is a solid-state device (not an electro-mechanical device like a CD-ROM). Its contents can be randomly accessed, very similar to RAM, with the exception that it contains fixed codes/data that cannot be erased or overwritten by the programmer.

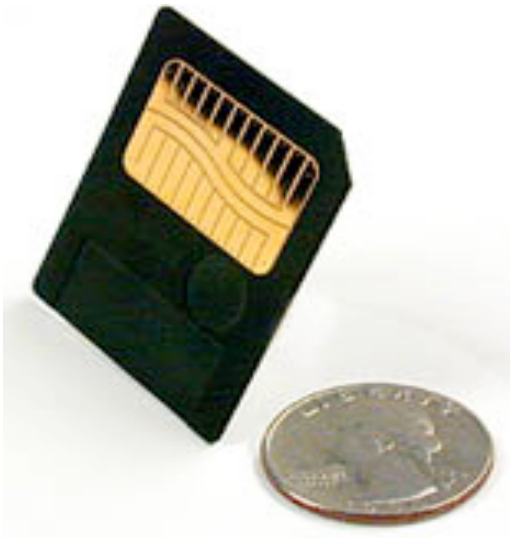
Needs special equipment to write data into it. Some can be programmed only once, and often done at the factory. In the programmable variety of ROMs **(PROM),** the contents can be erased (by ultraviolet radiation or electrical pulses as **in EEPROM**) and reprogrammed.

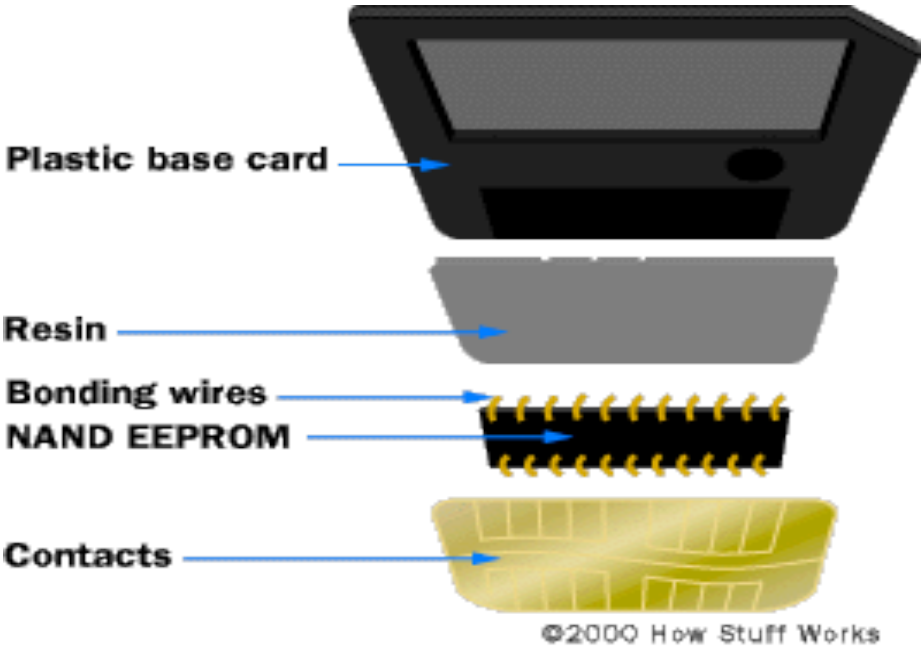
Flash Memory

Flash Memory is a form of EEPROM (Electrically Erasable Programmable ROM). Extensively used in USB thumb drives, memory sticks etc. **Smart Media** and **Compact Flash** are used as electronic films in digital cameras.

Flash memory works much faster than traditional EEPROMs because instead of erasing one byte at a time, it erases a block or the entire chip, and then rewrites it.

Smart Media Cards



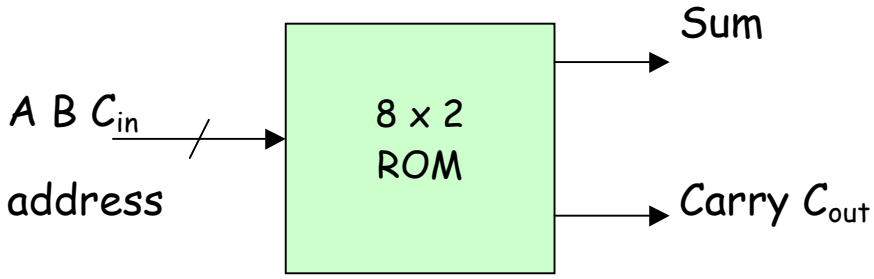


[Acknowledgment: Howstuffworks website]

Logic Design using ROMs

ROMs can also be used in logic design, and it opens up interesting possibilities. The design of combinational circuits can be reduced to a simple table look-up

Example. Design a full adder using a ROM



The content of the ROM are given in the table below:

Address	Sum	Carry
000	0	0
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	1

FPGA (Field Programmable Gate Array)

FPGA is a semiconductor device containing programmable logic components and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions.

In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. It is a useful building block for prototyping hardware design.