How fast do clocks run in today's computers?

Suppose we have a 2 GHz system.

2 GHz = 2 x 10^9 cycles per second
1 cycle = .5 x 10^{-9} seconds
= .5 nanoseconds (ns)
or 500 picoseconds (ps)

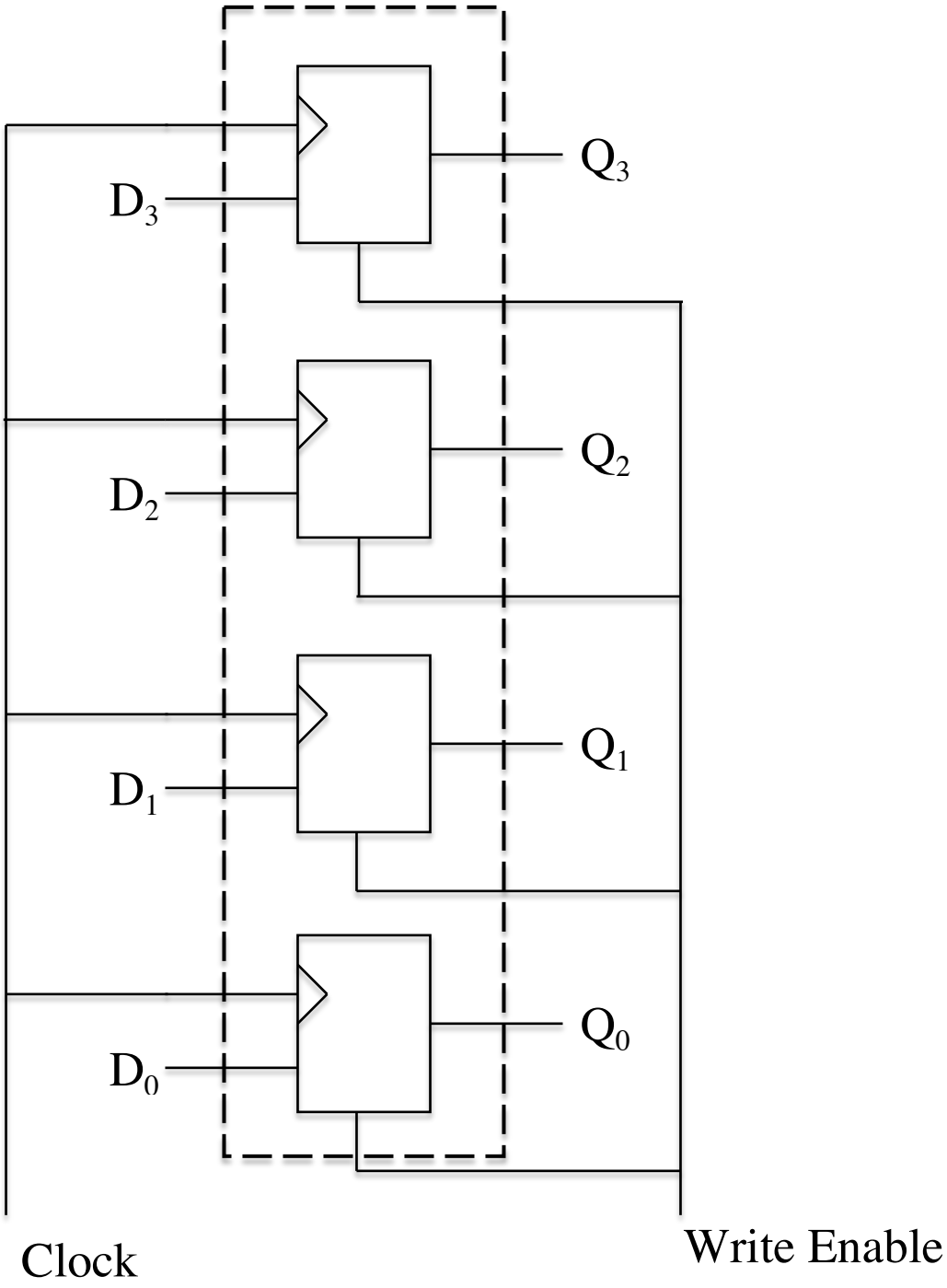
Terminology for subdivisions:
http://en.wikipedia.org/wiki/SI

How to make a 4-bit register from D flip-flops?
Think about the high-level behavior of a 4-bit register:

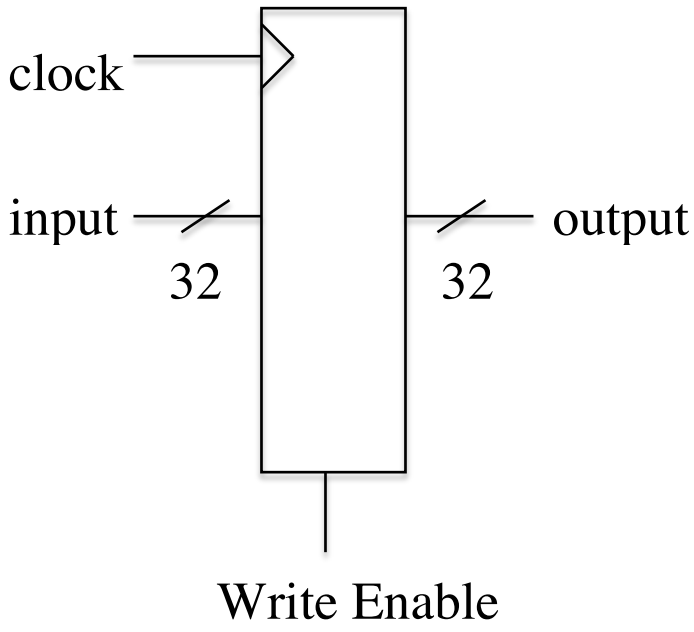
1) contents of register can be read all the time
   (at output pins)
2) to write a register, assert (or set to 1) the write-enable pin.
3) data at register input pins is written into register

Use one D flip-flop for each bit of register.
Output Q of flip-flops -> output of register
Enable of flip-flops -> write-enable of register
Input D of flip-flops -> input of register

See also Logisim example: 3_4BitReg


Q_3
D_3
Q_2
D_2
Q_1
D_1
Q_0
D_0
Clock
Write Enable

Single 32-bit register (ex. PC):

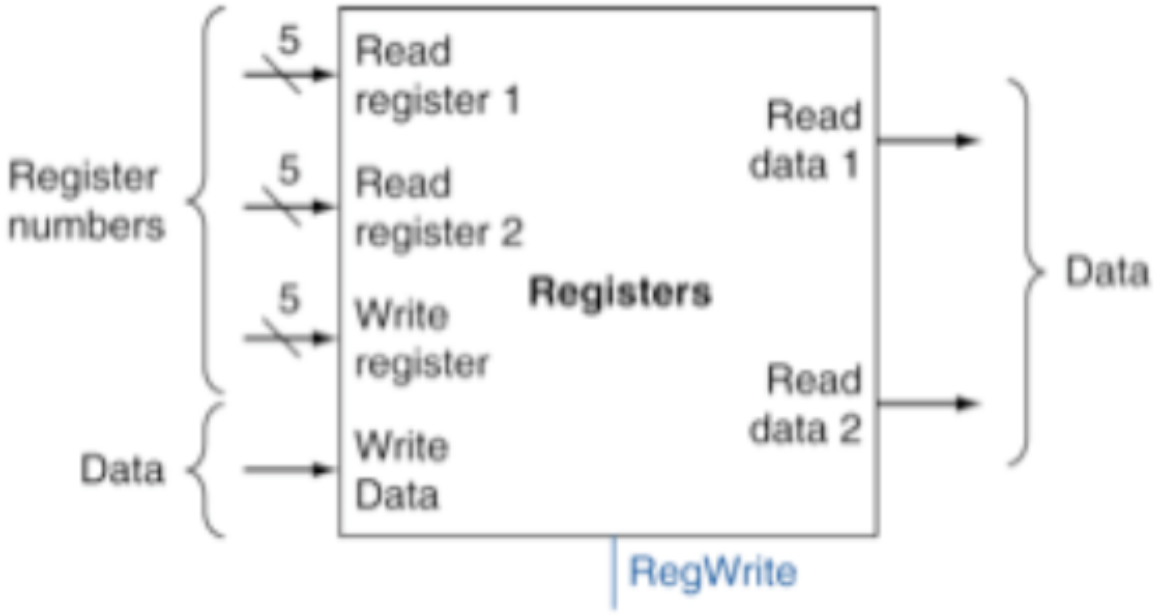


MIPS has 32 integer registers, organized in a *register file*.

A typical design:

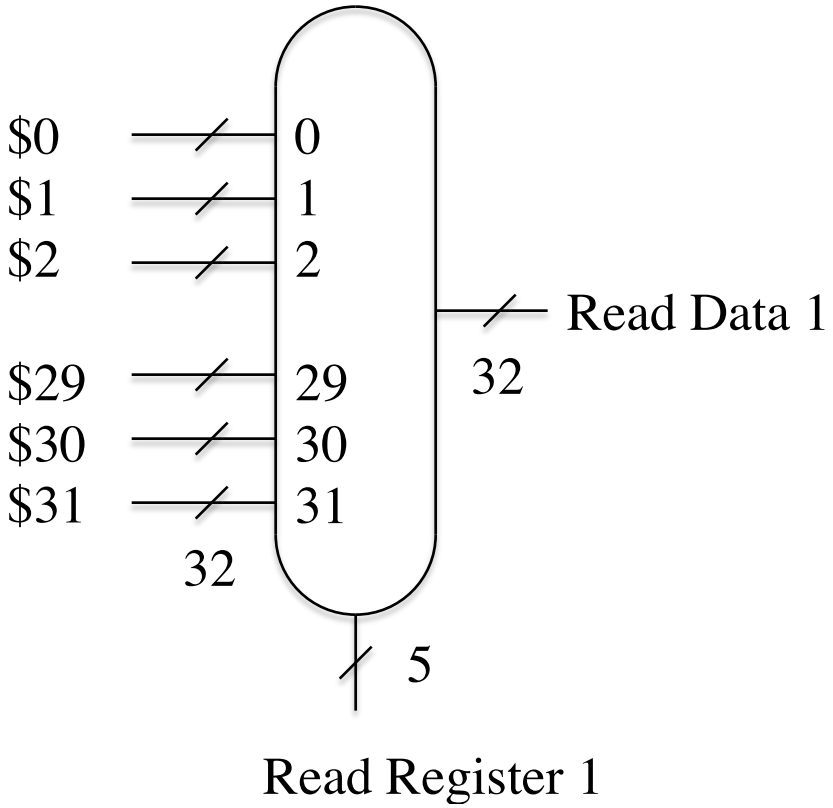
Can read two registers at the same time (Read Register 1, Read Register 2)

Can write one register at a time (Write Register)



To read register 1 (register 2 is similar):

5-bit register number used to select from 32 registers (use multiplexor)



Register file with two read ports:

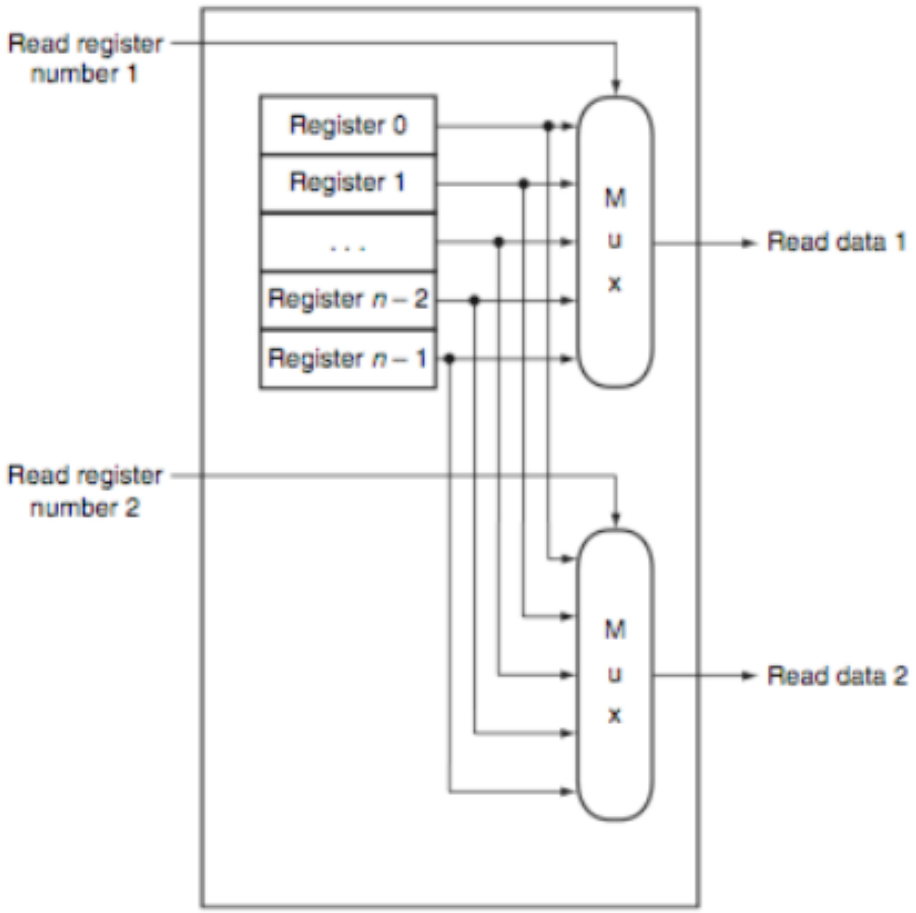


FIGURE C.8.8 The implementation of two read ports for a register file with *n* registers can be done with a pair of *n*-to-1 multiplexors, each 32 bits wide. The register read number signal is used as the multiplexor selector signal. Figure C.8.9 shows how the write port is implemented.

To write a register:

Write Data is connected to inputs of all 32 registers!

But set Write Enable of only *one* register

Usually, write (enable) is 0 (not writing)

When write 0->1, only one reg. is written!

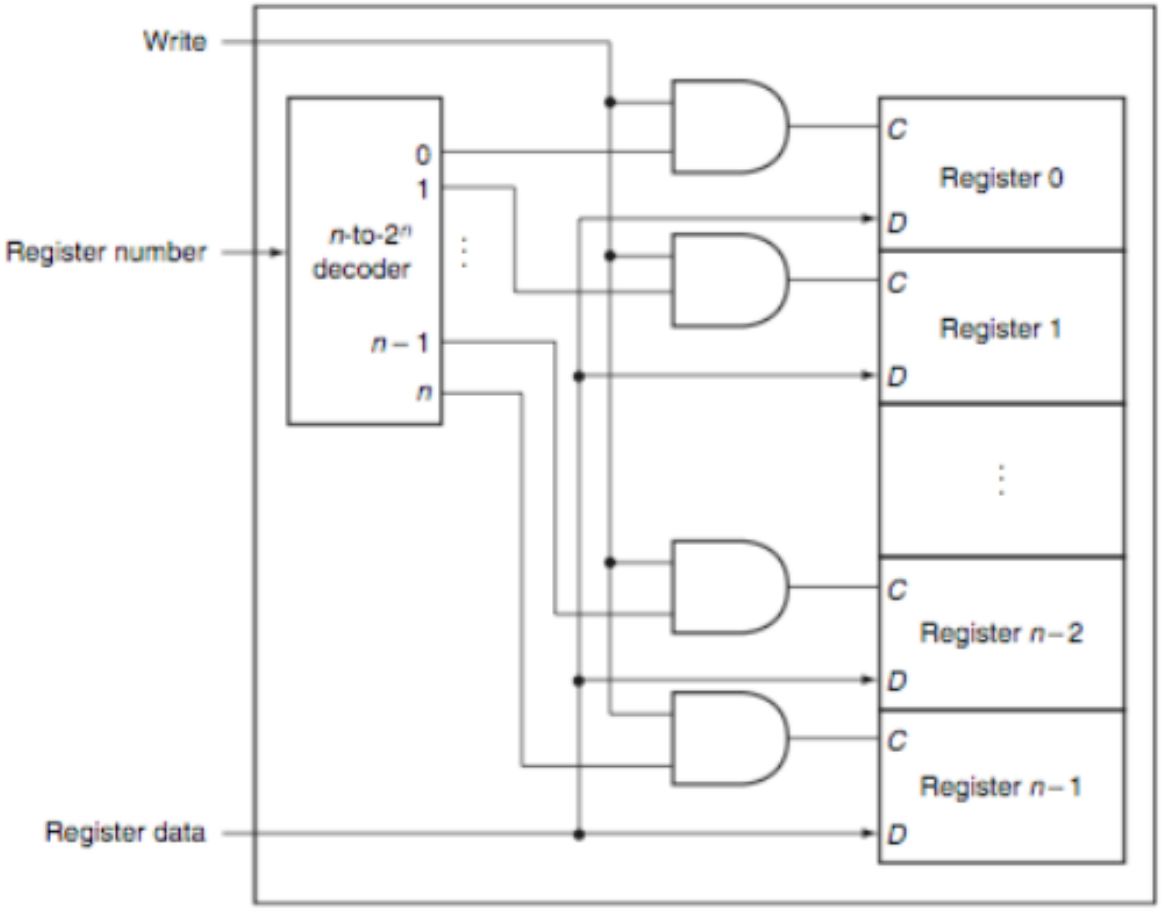


FIGURE C.8.9 The write port for a register file is implemented with a decoder that is used with the write signal to generate the *C* input to the registers. All three inputs (the register number, the data, and the write signal) will have setup and hold-time constraints that ensure that the correct data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

Memory Elements

Memory blocks are based on *Static Random Access Memory (SRAM)* or *Dynamic Random Access Memory (DRAM)* technology. (More on the differences later.)

A memory chip looks similar to a register file, with one read port:

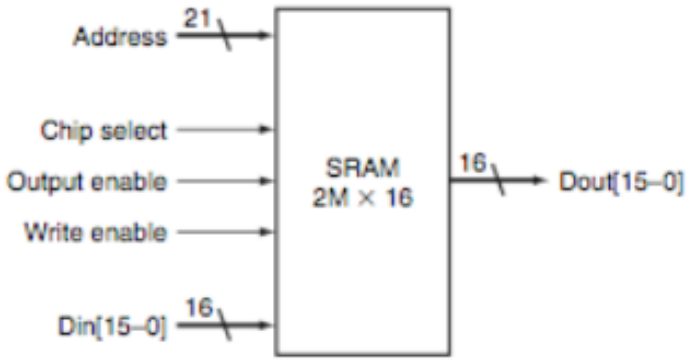


FIGURE C.9.1 A 32K × 8 SRAM showing the 21 address lines (32K = 2^{15}) and 16 data inputs, the 3 control lines, and the 16 data outputs.

This is a 2M x 16 SRAM chip, meaning:
1) the chip contains 2M (2 x 2^{20}) slots
hence, 21 address bits
2) each slot contains 16 bits

Chip select:

0 means this chip is not being used

1 means this chip is being used

Output enable:

0 means this chip is not sending data to output pins (like it's disconnected!)

1 means this chip is sending data to output pins (it's connected)

Write enable:

0 means contents of chip do not change

1 means data on Din pins get written to appropriate location

DRAM has similar pins.

In general, suppose number of address pins = A

Number of memory locations in chip = 2^A

Number of bits per location

= number of data input pins (Din)

= number of data output pins (Dout)

Main difference between SRAM/DRAM and register file:

> `each SRAM/DRAM chip contains many more locations than the register file`

Instead of huge multiplexor to select output (or read data), use *tri-state buffers* and a bus. (Don't worry about details; see Appendix C C-58 to C-66 if interested.)

Instead of huge decoder to select input, organize contents as 2-dimensional matrix of bits, use 2-step decoding. (Don't worry about details.)

Summary

Topics covered in this chapter:

Combinational logic basics
Flip-flops and registers
Memory elements

Next, we'll put them together to implement a simple MIPS CPU.