Register File

Inside of every computer is a Central Processing Unit, or CPU. Inside of that will be a register file. A register file is a piece of hardware that stores data temporarily in registers for the processor to use. The data in these registers can be added together, multiplied, stored onto a computer’s hard drive, etc. The same could be done with memory from the hard drive, but registers provide more speed in computing, so the CPU can get its tasks done faster.

Computers deal with data in zeros and ones. A zero or a one is called a “bit” of data. The register file described in this document uses 32 bits of data, or 32 zeros and ones in succession. Bits can also be used as Boolean, or true and false, values. A zero is false, and a one is true. This can be used to compute logic in the register file.

A register file consists of three major parts. An array of registers, a decoder, and two multiplexors are all required. Each component is described in more detail below.

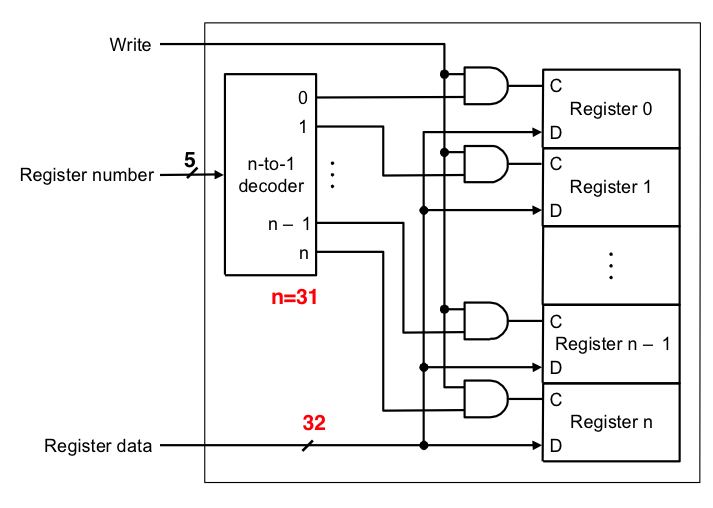


Figure 1

C – Write Enable ports

D – Data Input ports

**Registers**

First of all, the register file contains a register array. In a standard register file, there are 32 registers in the array. Each register will contain two inputs: a data input, and a write enable input. The data input takes in 32 bits of data, and each data input port is connected to the data input port for the entire register file (See Fig. 1). The write enable port determines whether or not the data input is actually written to the register. This value is true if both the write enable port for the register file is true and the register is the one specified by the register address, determined by the decoder. The half rounded rectangles seen in Fig. 1 are AND gates, outputting 1 only if the two inputs are 1.

**Decoder**

This decoder takes in 5 bits for a register address, and outputs a 1 in the slot corresponding to that address. A decoder basically converts a 5 bit binary value into its decimal equivalent and sets that corresponding output to true. For example, 00010 in binary is 2 in decimal. Therefore, output 2 of the decoder will be 1, and all the other will be zero. This allows the data to be written to register 2 and only register 2, provided the write enable for the register file is 1.

**Multiplexors**

The other half of the register file consists of two multiplexors. A multiplexor, or mux, takes a bunch of n-bit values (n is 32 in this case) and outputs just one of them, depending on what address is used. This register file uses two multiplexors, so that basic computer instructions can be executed. The outputs of the multiplexors extend beyond the scope of the register file so other components can use the data. For example, if a computer wanted to add data together, it would make sense to have more than one read data port (and consequently, more than one multiplexor) to add multiple numbers together. There are only two used because you can write the partial sum to another register to be ready for another addition.

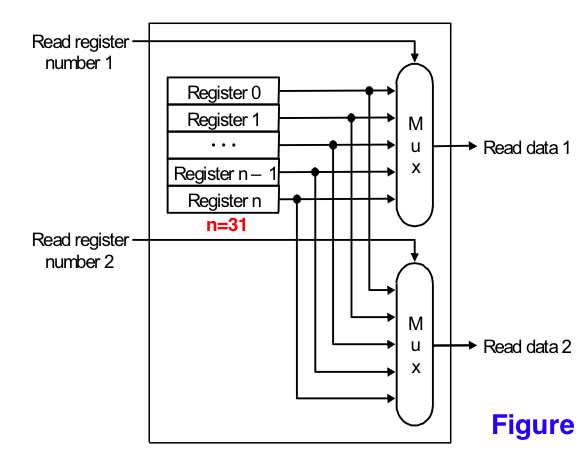


Figure 2

Register files are one of the key components of a CPU. Without them, it would take much longer for simple calculations to be run due to the CPU having to read the data from the hard drive every time it needed something. The CPU will specify a register to write to, and two to read from. The decoder will enable the correct register to be written to, along with the write enable signal. The multiplexors with choose the correct registers to read from based on the read addresses. Simultaneously, data is written to the write register, and the two read registers output their data to the rest of the CPU.

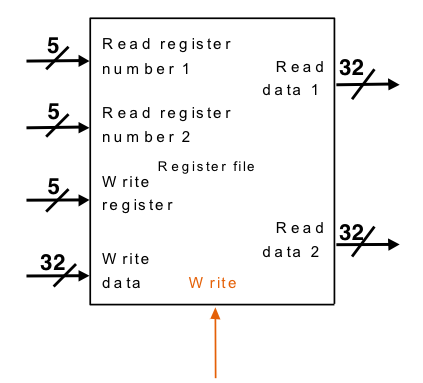


Figure 3

**Technical Details**

|  |  |
| --- | --- |
| 32 | 32-bit registers |
| 2 | 32-bit multiplexors |
| 1 | 5 to 32 decoder |
| 2 | 5-bit Read address ports |
| 1 | 5-bit Write address port |
| 2 | 32-bit read data ports |
| 1 | 32-bit write data port |
| 1 | 1-bit write enable port |