SINGLE CYCLE CPU DESIGN

Selin Kaya & Sinan Cem Yücel

Contents

CPU Design

Our single cycle CPU design consists of six important components.

- PC (Program Counter)
- Instruction Memory
- Register File
- Control Unit
- ALU (Arithmetic Logic Unit)
- Data Memory

All of these components are explained in detail.

PC (Program Counter)

The PC, program counter, consists of an input and an output. In each clock tick, the input is increased by 1, and the output gives out the address of the 32-bit instruction that is about to be executed.

Instruction Memory

The 32-bit instructions are fetched using the Instruction Memory. The first 6-bits of the instruction is always our opcode (operation code). Depending on the opcode our instruction type varies between *R*, *I* and *J* type instructions.

R-Type Instructions

The R-Type Instructions, which are mostly arithmetic operations consists of *op*, *rs*, *rt*, *rd*, *shamt* and *funct*.

R-Type Instructions	op	rs	rt	rd	shamt	funct
	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Our R-Type Instructions are:

- Addition (add)
- Subtraction (sub)
- Multiplication (mult)
- And Operation (and)
- Or Operation (or)
- Shift Left Logical (sll)
- Set Less Than (slt)
- Move from High (mfhi)
- Move from Low (mflo)
- Division (myIns → div)

I-Type Instructions

The I-Type Instructions, which are used in transferring and branching.

I-Type Instructions	op	rs	rt	immediate
	6 bits	5 bits	5 bits	16 bits

Our I-Type Instructions are:

- Addition Immediate (addi)
- Load Word (lw)
- Store Word (sw)
- Branch if Equal (beq)
- Branch Less Equals Zero (blez)

J-Type Instructions

The J-Type Instructions are used in jump instructions.

J-Type Instructions	op	target address
	6 bits	26 bits

Our J-Type Instructions are:

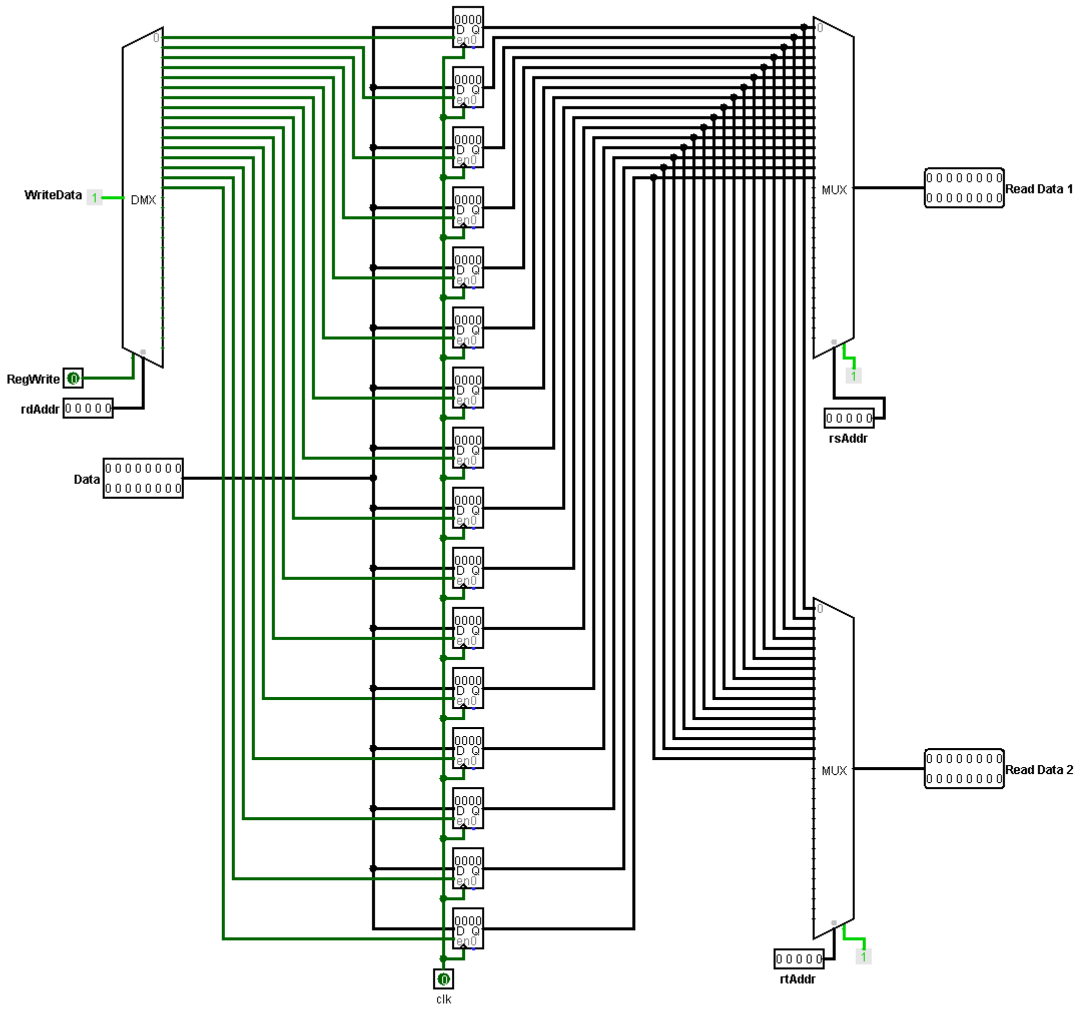
- Jump (j)

The Instruction memory we have in our Single Cycle CPU consists of a RAM (random access memory), which takes in the given instruction address and gives out the instruction itself.

In order to try the given test codes, we had to use the feature of Logisim to load the text file into the RAM.

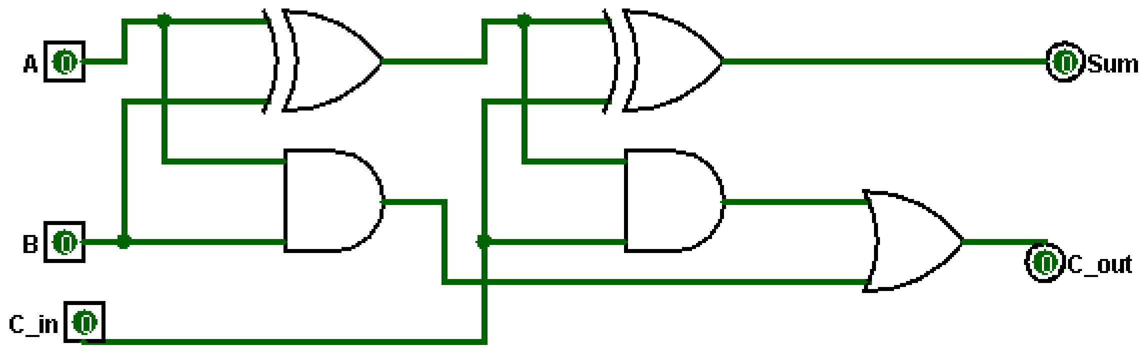
Register File

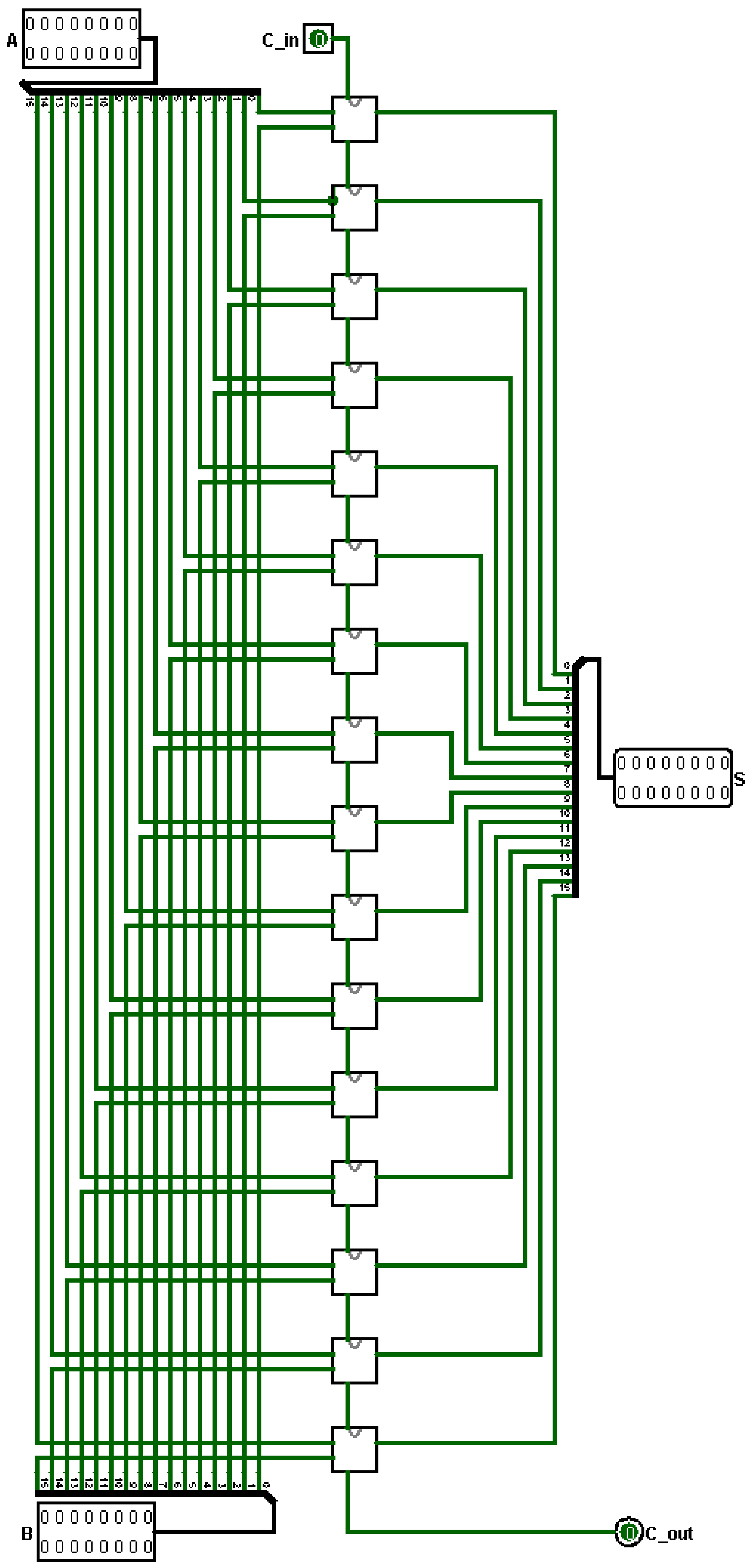
The Register File stores register data. It contains 16 register files of 16-bit size. The constant WriteData signal is lead to one of the 16 register files chosen by the rdAddr signal if the RegWrite signal is 1. The data signal is also fed to all 16 registers, which are enabled by the clock's falling edge. Also, on every clock tick, the data in the register corresponding to the rsAddr signal is output by the Read Data 1 signal. Similarly, the data in the register corresponding to the rtAddr signal is output by the Read Data 2 signal.



ALU

In our 16-bit ALU (arithmetic logic unit) we used full adders as our own adder design. Our full adder design is given below:

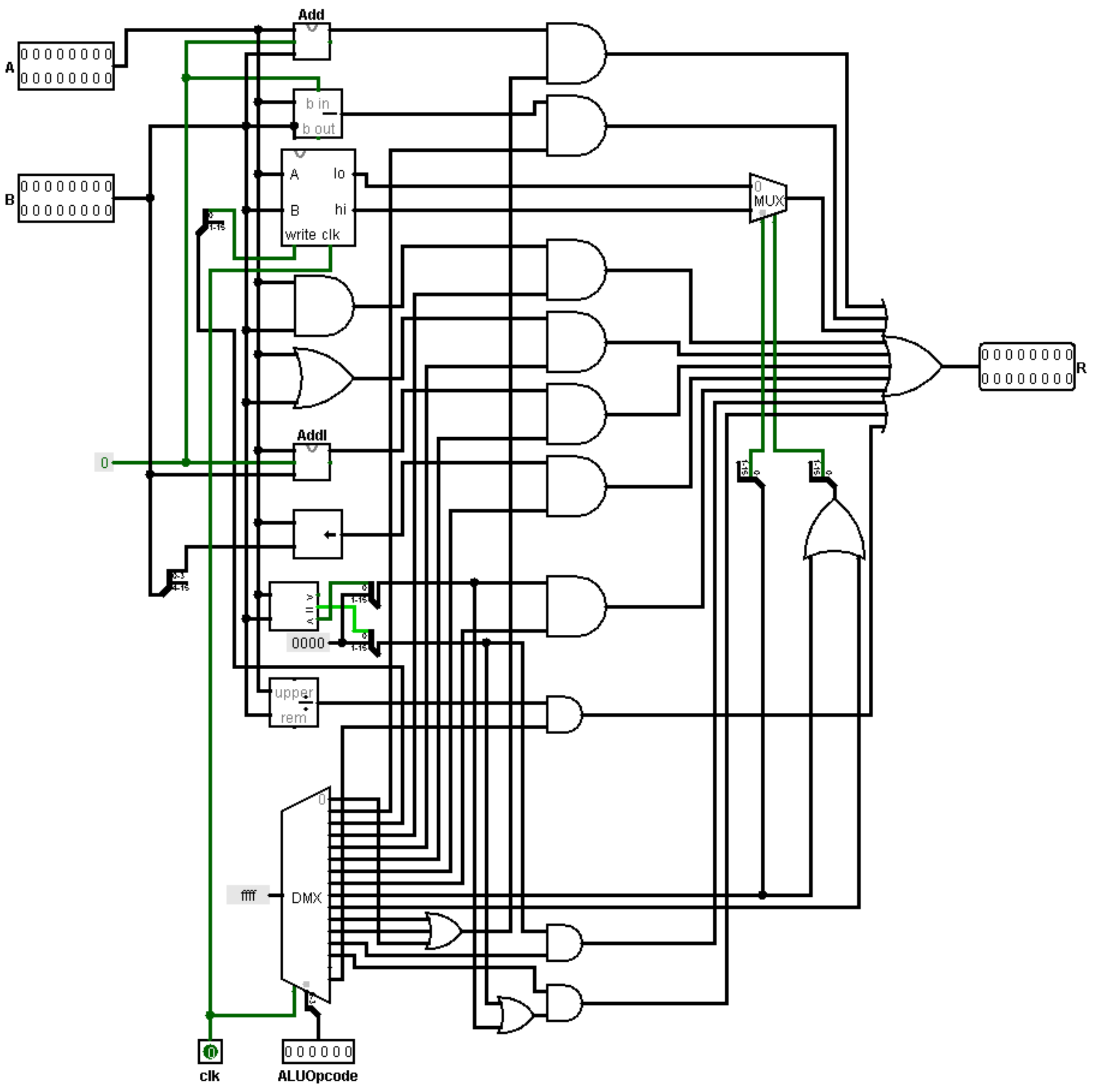




The data our ALU operates on is 16-bits, so we needed to create a bigger adder that can add two 16-bits data.

The adder we used in our 16-bit ALU is shown on the left side of the page.

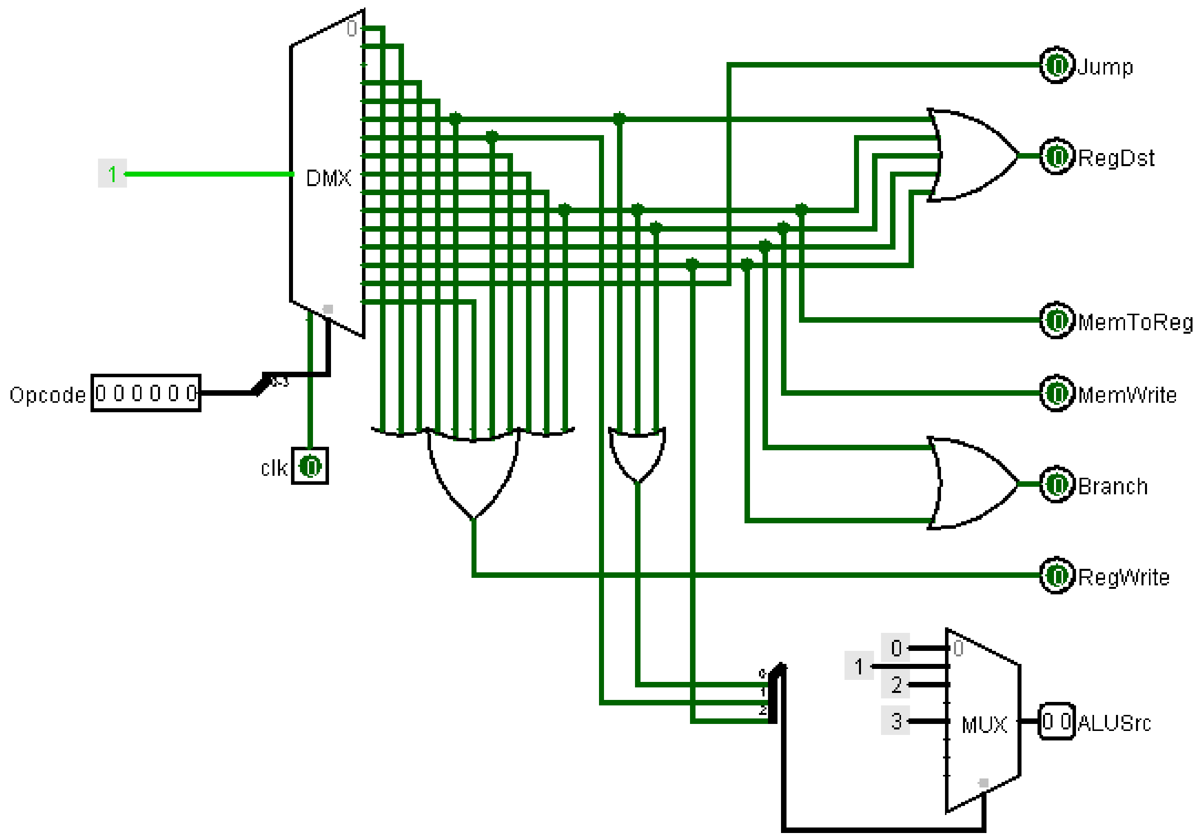
Our 16-bit ALU perform all the R-Type instructions and also our function, which is division.



Control Unit

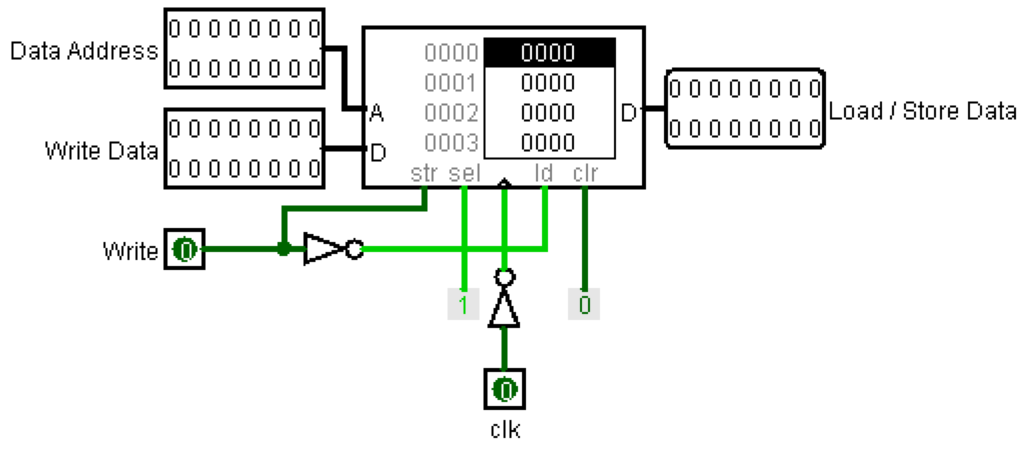
Signals are sent from the control unit depending on which features are used according to the given opcode.

- RegWrite is used to enable the demultiplexer in the register file.
- Branch goes into the multiplexer in our branch/jump path when any of the branching instructions are used.
- MemWrite is the signal which decides whether to write the data to the Data Memory.
- MemToReg is the signal which decides whether to send the data back to the ALU.
- RegDst chooses whether *rt* or *rd* will be used in the Write Register.
- Jump is 1, when our opcode performs the jumping instruction.



Data Memory

The data memory decides whether to store or write the data into the given address.



Data Address
00000000
00000000
Write Data
00000000
00000000
Write
A
D
0000 0000
0001 0000
0002 0000
0003 0000
str sel ld clr
D
00000000
00000000
Load / Store Data
1
0
clk