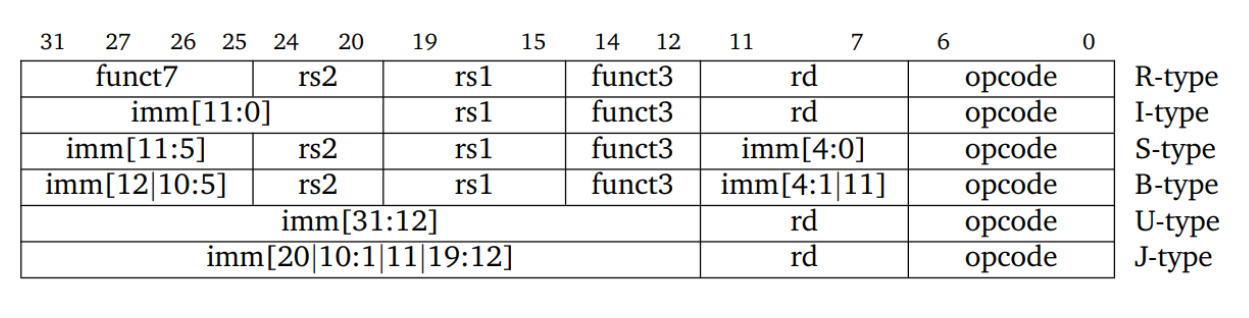
As you can see, we have added a new controller signal and expanded to a 4 to 1 multiplexer.  


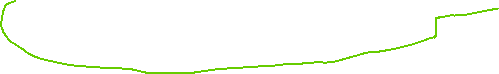


Figure : Modified Datapath

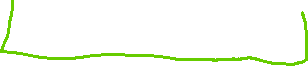


Figure : RISC-V Instruction Types

After designing the controller, we need to design the combinational controller since it’s single cycle RISC-V about which we are talking.



A diagram of a chemical formula

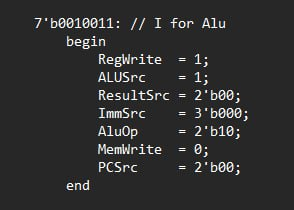
AI-generated content may be incorrect.

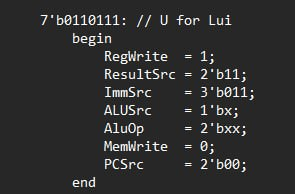


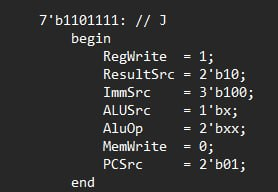
Figure : ALU Opcode Table

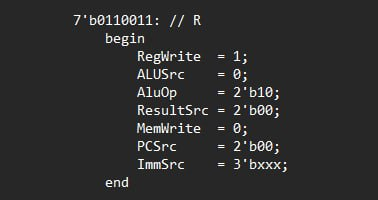
To implement the controller, first we need to code the ALU controller and the RISC-V controller and make an instance of it in the main controller.

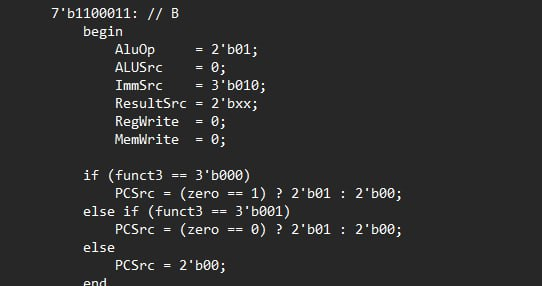
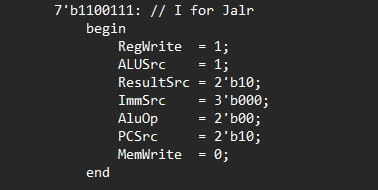
These are the logics obtained from our RISC-V controlling signal table:

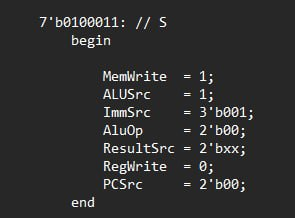












Now we need to implement the RISC-V assembly code for the desired program:

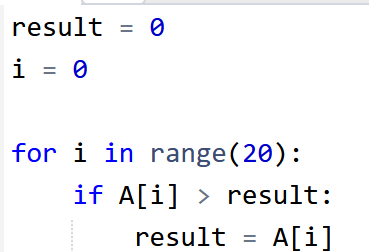
  




Figure : Algorithm

According to this algorithm we can write the RISC-V assembly code of this program as follows:

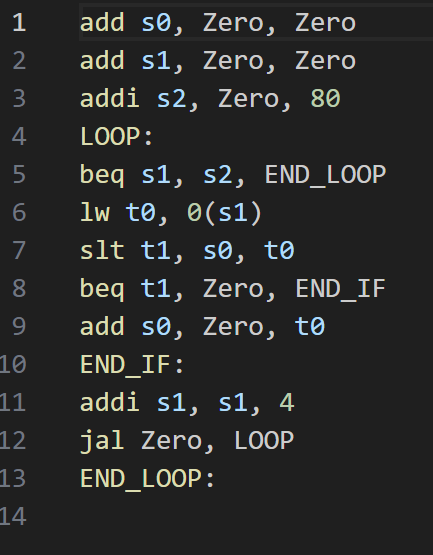


Figure : Assembly Code

The following is the resulting machine code of this program in hex format:

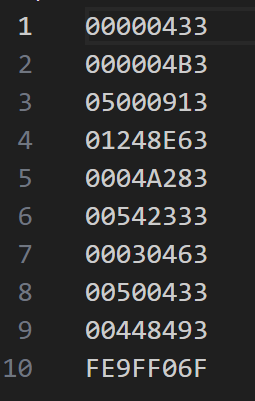


Figure : Hex Machine Code

We need to specify the 20 numbers in our program in “data.mem” file in 80 lines each representing a byte (8 bits).

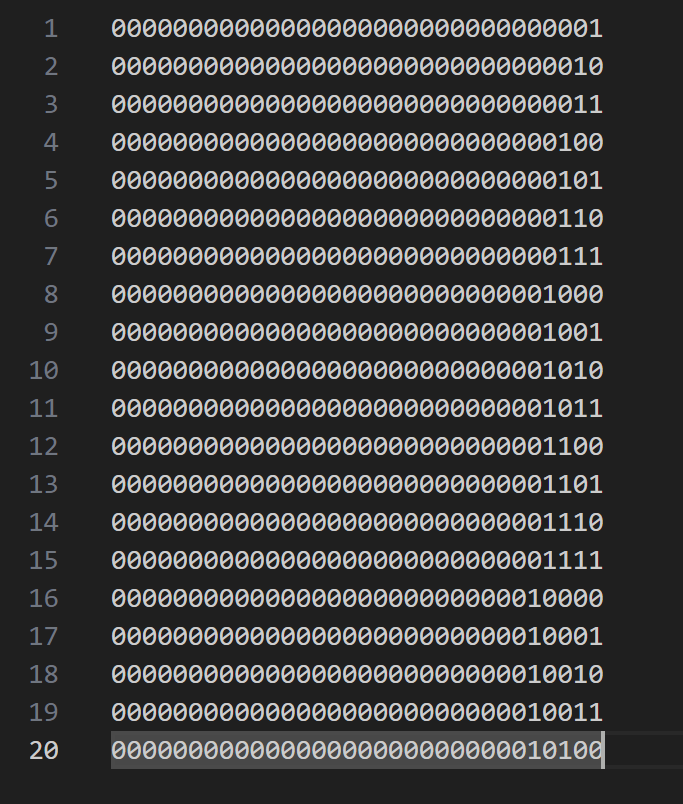
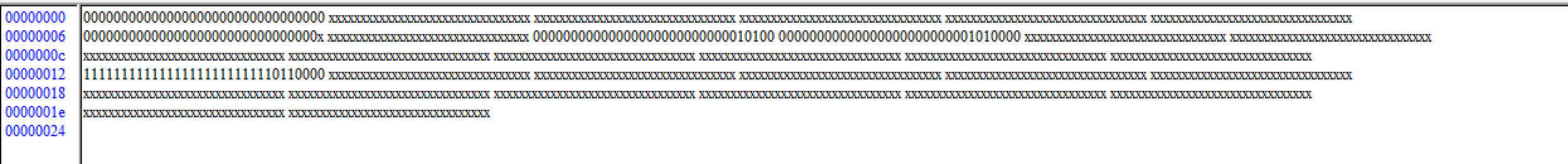


Figure : The twenty 32bit numbers given

A screenshot of a computer screen

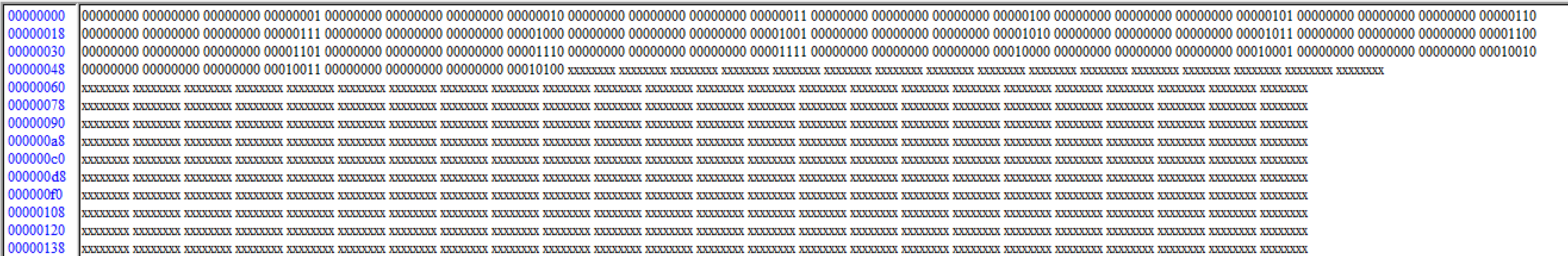
AI-generated content may be incorrect.

Figure : Twenty numbers in 80 lines of 8bit numbers.





As you can see after running the program we have the number 20 which is the maximum of given numbers stored in x12 register or the s0 register which is just the Result in our pseudo code.



This is the Data Memory which our numbers from 1 to 20 is stored.