CS 223 Project

Selen Görgün

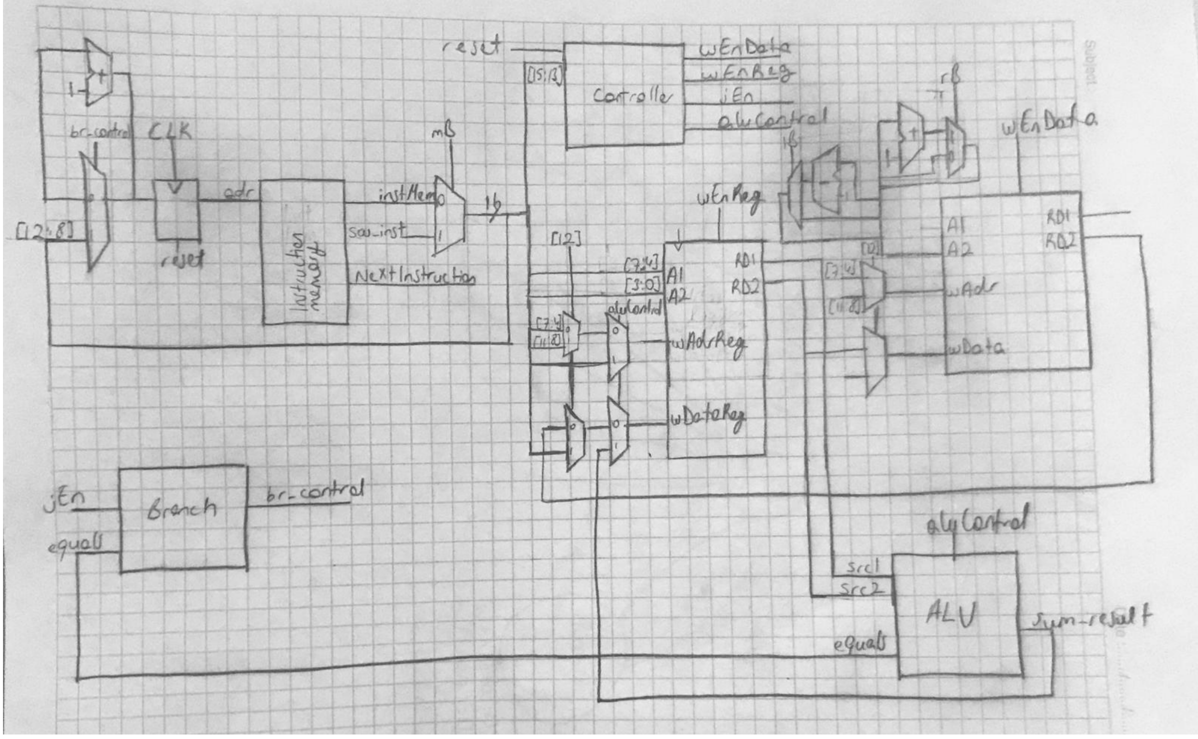
CS 223-006

21802674

Single Cycle Microprocessor

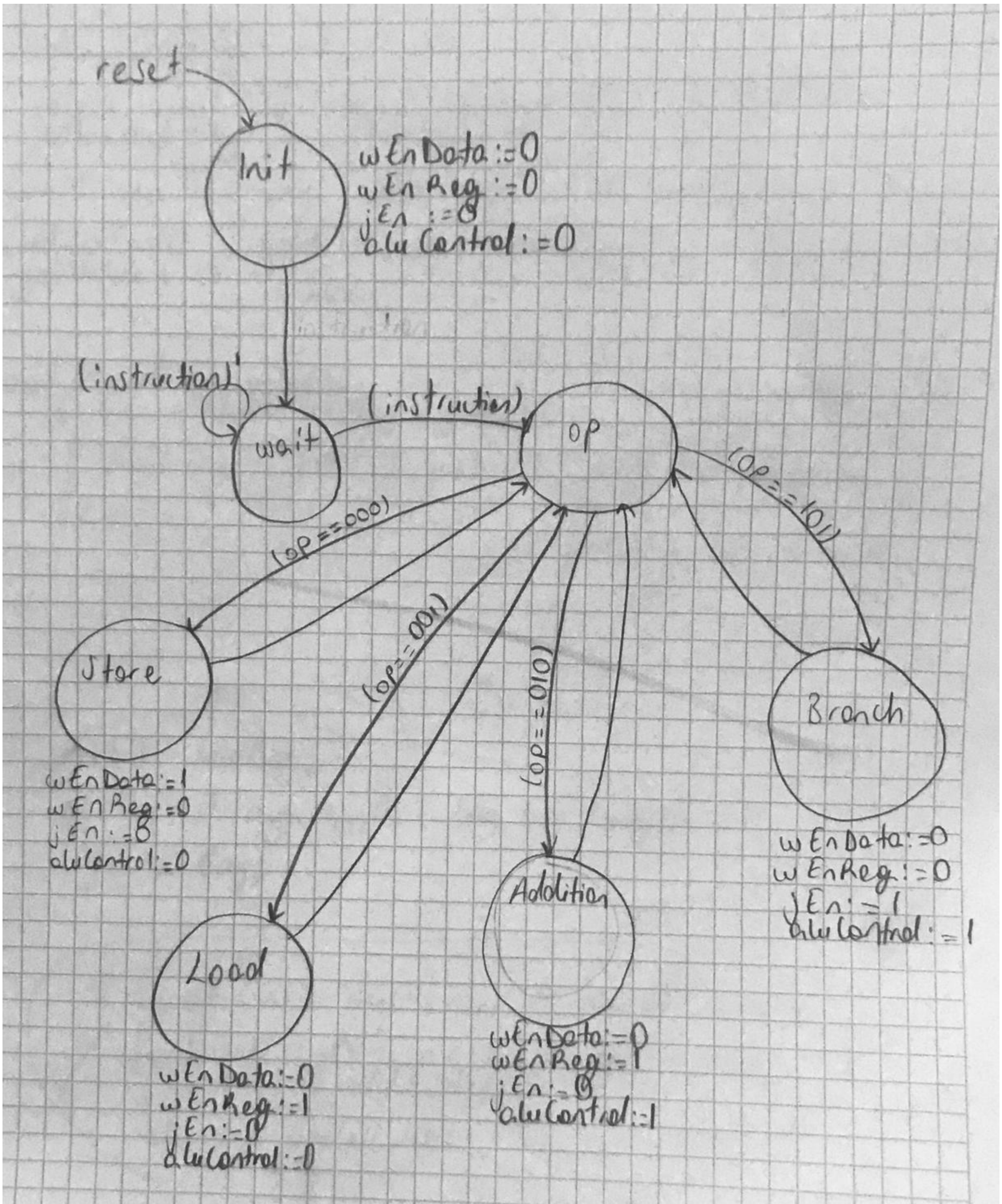
a) Block Diagram of the Single-cycle Processor

The aim of Single-Cycle microprocessor is performing some operations according to 16-bit instructions. In this design, microprocessor is able to perform four operations which are store, load, addition and branch. As seen on the block diagram, the instruction is being either being read from the instruction memory which is a rom with some hardcoded instructions or the user can enter the instruction on the switches. If the user does not enter an instruction in the switcthes, the program counter gives the address of the instruction to the instruction memory and instruction memory gives the instruction to the other modules to perform and also the next instruction to the leds. In the case of branch, the counter is not being incremented but the address taken from the current instruction is used to update the adress of the instructions in the instruction memory. After the instruction is decided on the instruction memory, the opcode is being sent to the controller. Controller decides which control outputs to send byb checking the opcode. In top module, the inputs that are going to be used by the modules are initilized by checking the immediate bit which is the 12th bit of the instruction. Branch module, decides if the branch operations is going to be performed or not by cheking if the datas are equal and the enable output from the controller is 1. Alu module performes 2 operations which are addition and comparition. All the operations are performed by pushing the buttons. The up button is reset button, the down button executes the next instruction, the right and left button is used to circulate thourusgh data memory which is going to be showed in the seven segment display. The middle button is used to execute the instruction taken from the switches.



b) State Diagram of the Controller

Control unit of the microprocessor decides which operation is going to be executed by checking the first 3 bits of the instruction which is called "opcode". According to my design, controller has 4 1-bit outputs: wEnData, wEnReg, jEn and aluControl. When the opcode is equal to 000, the processor is going to store data on data memory so only the wEnData output is 1. When the opcode is equal to 001, the processor is going to load data on register file so only the wEnReg output is 1. When the opcode is equal to 010, the processor is going to read to adresses from the register and then after adding the datas in the given adresses, it will write the result to register file. In this operation wEnReg and aluControl outputs are both 1 since the load and addition operations are executed in the same state. When the opcode is equal to 101, the processor is going to read two datas from the register file and if they are equal, the next instruction to be executed will be the instruction in the given address of the instruction memory. In this operation, both jEn and aluControl outputs are 1 since the comparision and jumping operations are executed in the same state.


reset
Init
wEnData:=0
wEnReg:=0
jEn :=0
aluControl:=0
(instruction)'
wait
(instruction)
op
(op==000)
Store
wEnData:=1
wEnReg:=0
jEn:=0
aluControl:=0
(op==001)
Load
wEnData:=0
wEnReg:=1
jEn:=0
aluControl:=0
(op==010)
Addition
wEnData:=0
wEnReg:=1
jEn:=0
aluControl:=1
(op==101)
Branch
wEnData:=0
wEnReg:=0
jEn:=1
aluControl:=1