Samuel Zurowski

1) The only unsuccessful part was the store as I did not load the data correctly into the MDR but since you told me it was only worth two points, I did not fix it. I could have easily fixed it.

2)

```
Function: void StateMachine::start()
Purpose: to start the state machine after the instructions are put in memory (after
the object is created).

Function: void StateMachine::checkInfo()
Purpose: to actually start the current states info to do the states a function just to
separate the code up a bit. But it is what runs each step in the state machine and
figues instruction type and updates everything in the memory, cpu, and regfile when it
is needed. Based on op code it checks the type in a switch and runs those functions.

Function: void StateMachine::branchInstr(int op)
Input: the code of the alu for the branch instr
Purpose: to run the state machine for the branch instruction and based on the zflag
update the pc.

Function: void StateMachine::jump()
Purpose: to do the jump instructions state machine which updates the pc based on the
offset of the jump instruction

Function: void StateMachine::checkRType()
Purpose: if the instruction is an rType check the func of the encoding and based on
it, do a specific r instruction type.

Function: void StateMachine::iType(int op)
Input: op which determines what the ALUop is going to used.
Purpose: to run the I type instructions which runs the states and updates the reg
file.

Function: void StateMachine::stateOne()
Purpose: to run the state machine of step 1 for every instruction as it is the same.
Fetchs instruction from ir.

Function: void StateMachine::stateTwo()
Purpose: to run the state machine of step 2 for every instruction as it is the same.
Updates the pc by incrementing by 4. And loads a & b.

Function: void StateMachine::incrementPC()
Purpose: increments the PC by 4 and loads it into pc.
```

Function: void StateMachine::setAoe(bool val)
Input: bool as if a can be loaded.
Purpose: put on the s1 bus the rs1 value from current instruction.

Function: void StateMachine::setBoe(bool val)
Input: bool as if b can be loaded.
Purpose: put on the s2 bus the value of rs2

Function: void StateMachine::rType(int aluCode)
Input: alucode for what the alu is going to do
Purpose: run the state machine for the rtype instructions updates the reg file.

Function: void StateMachine::updateReg(long reg)
Input: op code for the reg (should of changed it to int but…)
Purpose: based on the parameter it puts into the register the value of c.

Function: void StateMachine::setAluOP(int op)
Input: op code for what the alu is going to do
Purpose: based on the alu, put on the dest bus the results. Also updates the zflag.

Function: void StateMachine::setImm(string s)
Input: binary string of the immediate.
Purpose: makes the vale of the immediate signed.

Function: void StateMachine::setOffset(string s)
Input: binary string of the offset value
Purpose: makes the offset signed and sets the offset val.

Function: void StateMachine::printInfo()
Purpose: prints the info about the memory, registers, buses, and reg buffers.

Function: void StateMachine::loadMemory()
Purpose: loads from the hex instructions from the input file to memory.

Function: void StateMachine::setS2OP(int op)
Input: sets the s2 buses data based on the op code.
Purpose: sets the data on the s2 bus

Function: void StateMachine::PCMARSelect(bool op)
Input: based on bool sets the addr to mar or pc.
Purpose: set the addr bus to pc or the mar.

Function: void StateMachine::readMemory(bool read, int op)
Input: based on the bool to read and and op for the memsize
Purpose: reads and get the size of memory.

Function: string StateMachine::getMemSize(long addr, int type)
Input: the address that is being used and the size of memory

Purpose: sets the data to the result of the function of the binary string.

Function: void StateMachine::loadInstr(int op)
Input: op code of the load instruction for sizing.
Purpose: to read the memory and store in reg file based on op address size.

Function: void StateMachine::storeInstr(int op)
Input: the op code for the size being stored.
Purpose: to store to memory the new data of the instruction.

Function: void StateMachine::memWrite(bool write)
Input: bool for if they can write
Purpose: to parse the data to the memory but split it based on the sizing

Function: void StateMachine::mdroeS2(bool val)
Input: bool if allowed to do mdroes2
Purpose: to put on the s2 bus mdr.

Function: void StateMachine::mdrLoad(bool val)
Input: bool if allowed to load mdr
Purpose: puts on mdr the data

Function: void StateMachine::marLoad(bool val)
Input: bool to allow mar to load
Purpose: put on mar the dest.

Function: void StateMachine::IRLoad()
Purpose: to get all the data from IR, func, op, rs1 & rs2, offset, imm, and others.

Function: int getAluOP()
Purpose: returns the alu op

Function: void setMemOP(int op)
Purpose: sets the memop for word halfword etc.

Function: int getMemOP()
Purpose: gets the memop code.

Function: int getS2OP()
Purpose: gets the s2op code.

Function: void setREGSelect(int op)
Purpose: selects which reg to save to.

Function: int getREGSelect()
Purpose: gets the reg that is selected.

Function: void setPCLoad()

Purpose: To set the state.pc to the value of dest

Function: string getIR()
Purpose: get the value stored in IR (binary representation in string)

Function: void setIR(string s)
Purpose: to set the value of ir.

Functions:
void setOP(string s)
void setRS1(string s)
void setRS2(string s)
void setRd(string s)
void setFunc(string s)

PURPOSE: THESE ARE ALL SETTERS
INPUT: THE string in binary to set the value to.

Functions:
void setS1(long val) { s1 = val; }
void setS2(long val) { s2 = val; }

Purpose: sets the values of the s1 & s2 buses to the value entered in.

Function: int getPC() { return state.pc; }
Purpose: gets the value of the pc.

Functions:
void ALoad(bool val) { if(val) a = regFile[rs1]; }
void BLoad(bool val) { if(val) b = regFile[rs2]; }
void CLoad(bool val) { if(val) c = dest; }

Purpose: loads into these variables the values if true.
Param: bool if it can do these loads.

Functions:
void iroeS1(bool val) { if(val) s1 = stol(getIR(), nullptr, 2); }
void iroeS2(bool val){ if(val) s2 = stol(getIR(), nullptr, 2); }
Purpose: sets the values of the s1 and s2 buses(setter)
Param: if it can do this a bool.