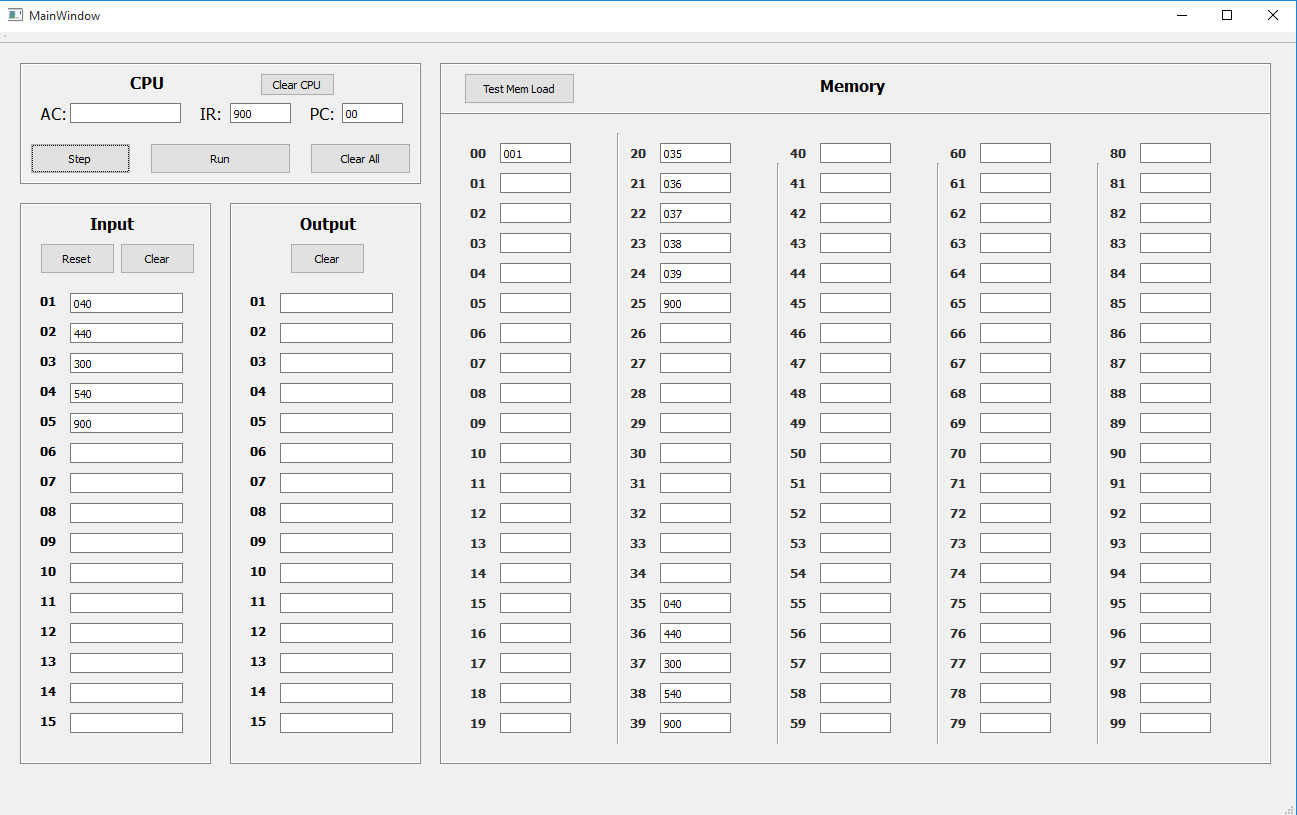
Before Running:

After Running:

Code Examples:

-User Input to Storage (This works similarly when inputing directly memory registers)

//Reading from text edit

void MainWindow::on\_input\_1\_textEdited(const QString &arg1)

{

computer.setInCard(0, arg1);

}

//Storing the value

void SimpleComputer::setInCard(int cardNum, QString value){

in[cardNum] = value;

}

-Running OpCode

//Run Buttton Click Code (Note TODO’s are for dialogue boxes for user notification of run finished or errors). Note that it updates window as soon as program run is finished

void MainWindow::on\_runBtn\_clicked()

{

if(!computer.halted()){

computer.runProgram();

this->UpdateMemoryRegisters();

this->UpdateCPU();

//TODO - DISPLAY FINISHED or ERROR msg

}

else{

//TODO - DISPLAY PROGRAM currently halted

}

}

//Code for simple computer running

void SimpleComputer::runProgram(){

do{

step();

}while(!isHalt);

}

//Code for step function on next pages.

void SimpleComputer::step(){

if(!isHalt){

//Read Instructions

QString instructions = mem[pCount];

//Load Instructions to iReg

iReg = instructions.toInt();

//Interpret and Execute Instructions

int opCode = (instructions.left(1)).toInt();

int destination = (instructions.right(2)).toInt();

switch(opCode){

case INP:

{

qDebug() << "Input to " << destination;

QString fromIn = getInCard(inIterator++);

mem[destination] = fromIn;

pCount++;

break;

}

case OUT:

{

qDebug() << "Output to " << destination;

QString toOut = mem[destination];

setOutCard(outIterator++, toOut);

pCount++;

break;

}

case ADD:

{

qDebug() << "Adding from " << destination;

accum += mem[destination].toInt();

pCount++;

break;

}

case SUB:

{

qDebug() << "Subtracting from" << destination;

accum -= mem[destination].toInt();

pCount++;

break;

}

case LDA:

{

qDebug() << "Loading from to " << destination;

int toLoad = mem[destination].toInt();

setAccum(toLoad);

pCount++;

break;

}

case STA:

{

qDebug() << "Loading from to " << destination;

QString toStore = getAccum();

mem[destination] = toStore;

pCount++;

break;

}

case JMP:

{

qDebug() << "Jumping to " << destination;

mem[99] = getPCount();

setPCount(destination);

break;

}

case TAC:

{

qDebug() << "Testing possible jump to" << destination;

if(accum < 0){

setPCount(destination);

qDebug() << "accum is negative: Jumping to " << destination;

}

break;

}

case SHF:

{

qDebug() << "Shifting values: " << destination;

int rShift = destination / 10;

int lShift = destination % 10;

//TODO - Shifting code

pCount++;

break;

}

case HLT:

{

qDebug() << "HALTING, PC = " << destination;

setPCount(destination);

isHalt = true;

break;

}

default: qDebug() << "ERROR - opCode undefined. OpCode: " << opCode << " destination: " << destination;

break;

}

}

}