1. Assume program execution time consists of 75% CPU time and 25%   
memory time that do not overlap. Which is a better enhancement for the overall   
performance and why?   
(a) Improve the CPU speed by 50%   
(b) Reducing the memory access time by 5 times

T = Tcpu + Tmem

Tcpu/T = 0.75

Tmem/T = 0.25

a) Tnew = Tcpu/1.5 + Tmem = 0.75T/1.5 + 0.25T = 0.75T

decreased by 25%

speedup = Told /Tnew = T/0.75T = 1.33

b) Tnew = Tcpu + Tmem/5 = 0.75T + 0.25T/5 = 0.8T

decreased by 20%

speedup = Told/Tnew = T/.08T = 1.25

option a is faster

2.

For the simulations I made c++ program in Visual Studio. The program has three functions in the source.cpp file. There are no command line arguments but there are quite a few user inputs in the c++ console. Once the source.cpp file is running it will pull up a console and ask the user to input the type of simulation the user wants to run. The input values for the three simulations are "single cycle", "multi cycle", and "pipelined". If any of these strings are inputted at the start of the program, then corresponding simulation will run. Below I have examples of the three simulations.

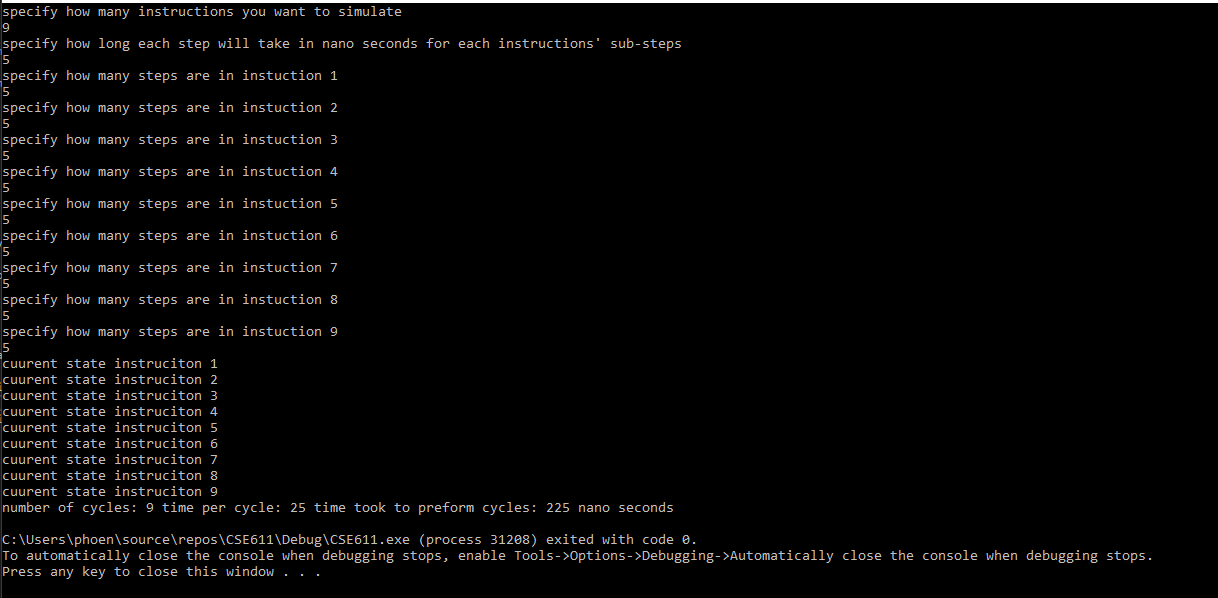


Figure 1: simulation 1 single cycle

In figure 1 above I ran the single cycle simulation. The simulation asks for some inputs from the user to test different scenarios. The first thing it will ask for is the number of instructions being run for the simulation. This number will determine how many instructions are used to calculate the number of cycles in the simulation. None of the inputted value, including this one, have been unit tested so the program will error if anything other then an integer is passed in. The next value it asks the user for is the time in which each sub step will take in each instruction. For simplicity all steps for every instruction runs at the same amount of time so what ever is entered here will be the same for all steps in all instructions. Next will ask how many steps are in each instruction individually. For single cycle it uses the largest instruction as the time needed to run one cycle so after each instruction has been given its steps the simulation runs and counts how many cycles where needed along with how much overall time it took to run the instructions. For single cycle the time taken is much higher than the other two simulations using the same values seen in the example above. For the example above I used 9 instructions with a time of 5 nano seconds per step in the instruction and then 5 steps per instruction.



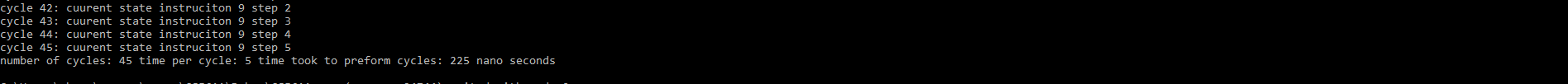


Figure 2: multi cycle simulator

The multi cycle simulator works similar to the single cycle by asking for the same information as before, the number of instructions, time taken for each step in the instructions, and number of steps for each instruction. The simulator then runs and shows which step it is on in each cycle and calculates the time it took to run all of its cycles. In this example of that I ran for this simulator the time is the same as the single cycle but that is only because each instruction has the same number of steps. If I reduce the number of steps for some of the instructions the number of cycles and time to run the cycles will decrease. However, there will still be a substantial more cycles run than in single cycle. The values used in this example where 9 instructions, 5 nano seconds, and 5 steps per instruction.

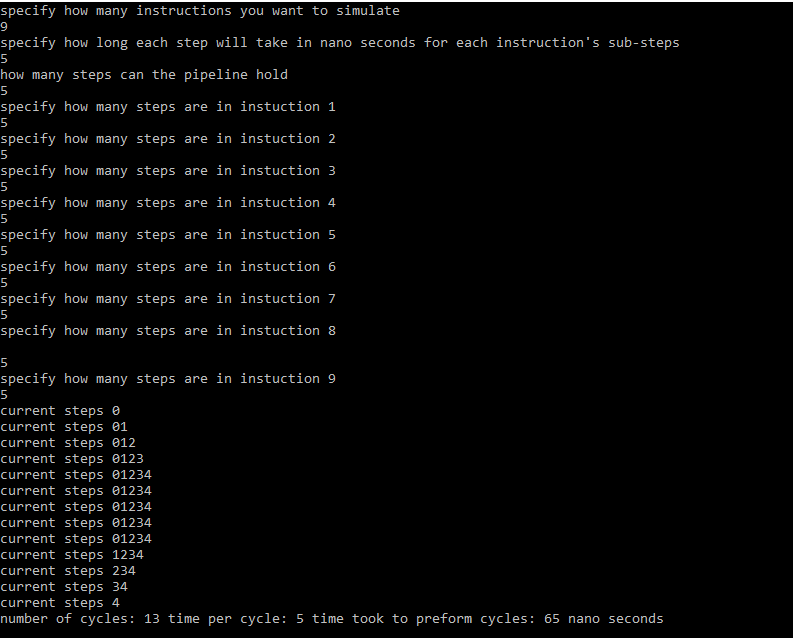


Figure 3: simulator 3 pipelined simulator

In simulator 3 the amount of time and cycles substantially decreased with the same tested inputs. Now however, there is a new input in the third user input called the pipeline size. This is how many steps the pipeline can hold at once and because of this that means there can be no instruction with a number of steps greater than the pipeline size. For the pipeline size I used a size of 5 and the same sizes for the precious user inputs in simulator 1 and 2.