Ira A. Fulton Schools of Engineering
School of Computing, Informatics and Decision Systems Engineering

SER 450 **COMPUTER ARCHITECTURE**

Your Name: **Jacob Hreshchyshyn** [illegible]

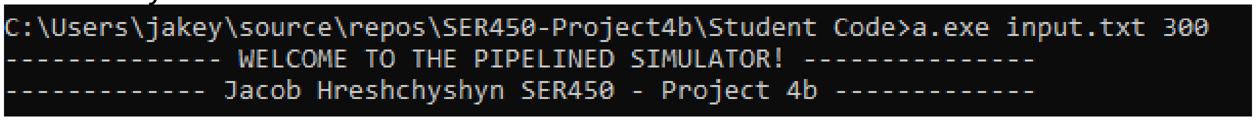
PROJECT 4B WORKSHEET

Step III: Gathering Data

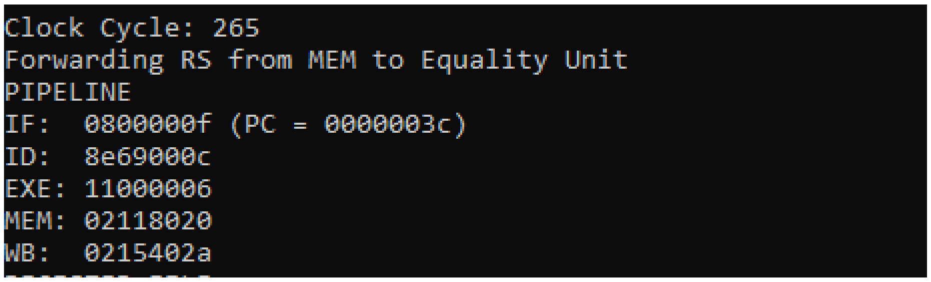
The management at SensoMIPS are evaluating two different product concepts for improved performance per Watt. The first is a pipelined version of the MIPS core. The second concept is a simulator that performs better optimization, including loop unrolling (which will further help pipelined performance). This first set of data gathering is related to just the pipelined architecture.

A. Using the newly completed pipelined simulator, find how many processor clocks are required to finish executing the program contained in “input.txt”. Program completion occurs when the instruction labeled “done:” completely traverses the pipeline for the first time. (you can find this label by viewing input.txt).

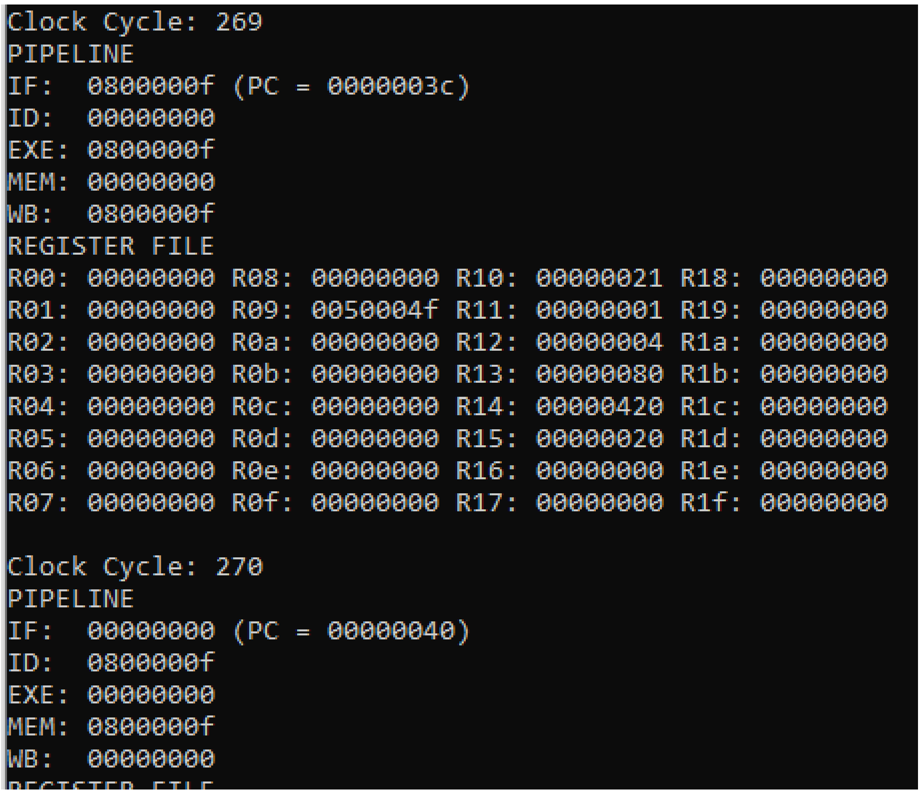
Include a screenshot of your program output (with banner visible) that shows your answer.



```
C:\Users\jakey\source\repos\SER450-Project4b\Student Code>a.exe input.txt 300
-------------- WELCOME TO THE PIPELINED SIMULATOR! ----------------
------------- Jacob Hreshchyshyn SER450 - Project 4b -------------
```



```
Clock Cycle: 265
Forwarding RS from MEM to Equality Unit
PIPELINE
IF:  0800000f (PC = 0000003c)
ID:  8e69000c
EXE: 11000006
MEM: 02118020
WB:  0215402a
```


```
Clock Cycle: 269
PIPELINE
IF:  0800000f (PC = 0000003c)
ID:  00000000
EXE: 0800000f
MEM: 00000000
WB:  0800000f
REGISTER FILE
R00: 00000000 R08: 00000000 R10: 00000021 R18: 00000000
R01: 00000000 R09: 0050004f R11: 00000001 R19: 00000000
R02: 00000000 R0a: 00000000 R12: 00000004 R1a: 00000000
R03: 00000000 R0b: 00000000 R13: 00000080 R1b: 00000000
R04: 00000000 R0c: 00000000 R14: 00000420 R1c: 00000000
R05: 00000000 R0d: 00000000 R15: 00000020 R1d: 00000000
R06: 00000000 R0e: 00000000 R16: 00000000 R1e: 00000000
R07: 00000000 R0f: 00000000 R17: 00000000 R1f: 00000000

Clock Cycle: 270
PIPELINE
IF:  00000000 (PC = 00000040)
ID:  0800000f
EXE: 00000000
MEM: 0800000f
WB:  00000000
```

These three screenshots are meant to demonstrate me running the program and the number of clock cycles in which the program runs. This program runs differently from the way this will be submitted in that it prints a dump of the CPU at every clock cycle. This helps me determine where the instruction indicating loop completion appears and in which clock cycle the program ultimately ends. The second screenshot demonstrates that the done instruction, which is encoded by the hex value 0x0800000f, first appears in the pipeline on the 265th clock cycle (the counting of cycles is based on Professor Sandy's counting in the output_debug.txt file). It traverses the rest of the pipeline until it reaches the 269th clock cycle, which is the stage in which the instruction is written back to the register. On the 270th clock cycle, the instruction has completely traversed the pipeline. Later cycles will show other done instructions reaching the WB stage as well. If complete traversal of the pipeline means reaching the WB stage, then the number of clocks required to complete the program is 269. If complete traversal of the pipeline means passing the WB stage, then the number of clocks required to complete the program is 270.

B.

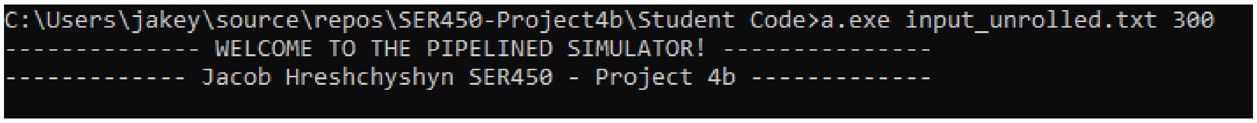
Compute the speedup of the pipelined processor if its clock cycle is 30% of the single-cycle clock cycle and the single cycle machine takes 270 cycles to complete.

For this, I will use 270 as the number of clock cycles the pipelined processor took to finish. Recall that the CPU time can be calculated by multiplying the number of clock cycles with the Clock Cycle Time. We are told that the pipelined processor clock cycle time is 30% of the single-cycle clock cycle time. Important to note is that this is not a 30% reduction. Rather, the cycle time is 0.3 * single-cycle time. Therefore, our speedup can be calculated using the following expression:

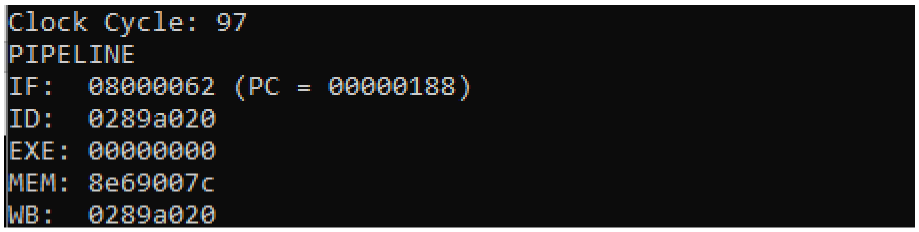
Speedup = CPU Time Single/CPU Time Pipelined = (270 cycles * Single Cycle Time)/(270 cycles * 0.3 * Single Cycle Time) = 1/0.3 = approximately 3.3 time speedup.

C.

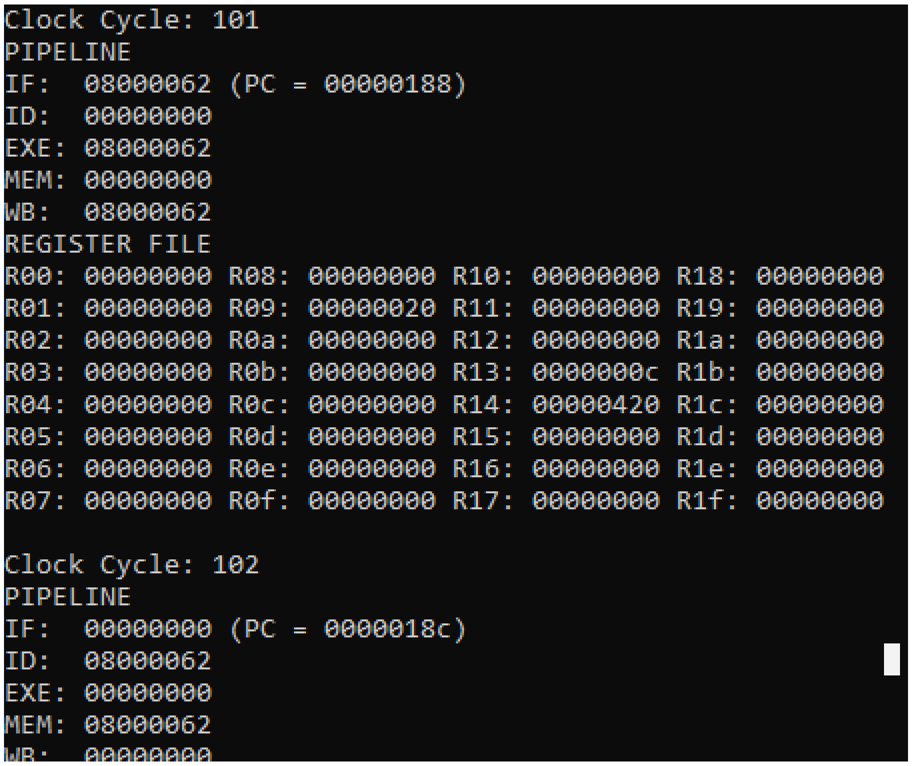
Now perform the same steps as A and B with the optimized compiler input file (input_unrolled.txt). Paste a screenshot of your program output (showing your banner message and compute the speedup of the loop-unrolled code running on the pipelined processor relative to the original single-cycle machine.



```
C:\Users\jakey\source\repos\SER450-Project4b\Student Code>a.exe input_unrolled.txt 300
-------------- WELCOME TO THE PIPELINED SIMULATOR! ---------------
------------- Jacob Hreshchyshyn SER450 - Project 4b -------------
```



```
Clock Cycle: 97
PIPELINE
IF:  08000062 (PC = 00000188)
ID:  0289a020
EXE: 00000000
MEM: 8e69007c
WB:  0289a020
```



```
Clock Cycle: 101
PIPELINE
IF:  08000062 (PC = 00000188)
ID:  00000000
EXE: 08000062
MEM: 00000000
WB:  08000062
REGISTER FILE
R00: 00000000 R08: 00000000 R10: 00000000 R18: 00000000
R01: 00000000 R09: 00000020 R11: 00000000 R19: 00000000
R02: 00000000 R0a: 00000000 R12: 00000000 R1a: 00000000
R03: 00000000 R0b: 00000000 R13: 0000000c R1b: 00000000
R04: 00000000 R0c: 00000000 R14: 00000420 R1c: 00000000
R05: 00000000 R0d: 00000000 R15: 00000000 R1d: 00000000
R06: 00000000 R0e: 00000000 R16: 00000000 R1e: 00000000
R07: 00000000 R0f: 00000000 R17: 00000000 R1f: 00000000

Clock Cycle: 102
PIPELINE
IF:  00000000 (PC = 0000018c)
ID:  08000062
EXE: 00000000
MEM: 08000062
WB:  00000000
```

In this case, the instruction that indicates program completion is encoded with the hex value 0x08000062. The first instance of this instruction in our pipeline is in clock cycle 97. It reaches the WB stage on cycle 101 and will have completely traversed the pipeline at cycle 102. We can find our speedup using the expression before:
Speedup = CPU Time Single/CPU Time Pipelined = (270 cycles * Single Cycle Time)/(102 cycles * 0.3 * Single Cycle Time) = 2.647058824/0.3 = approximately 8.8 time speedup.

Step IV: Business Impact

A.

The target customers for SensoMips are concerned about performance per Watt of power consumption. Power consumption of the current chip is 500milliWatts. If power consumption of the new chip is 500 milliwatts plus 5milliWatts for every percent of additional clock frequency, what is the power consumption of the new chip? (lower power consumption is better)

We can find the additional clock frequency by remembering that a clock of the pipelined processor runs at just 30% of the single-cycle processor. This means that, if the single-cycle processor has a clock speed of 1 second (which means that its frequency is 1 Hz), then the pipelined processor will have a clock speed of 0.3 * 1 second = 0.3 seconds (which means that it has a frequency of 1/0.3 seconds = 3.3 Hz). We need to find the percentage of additional clock frequency, which we can calculate by finding the percentage of improvement. We can use the following expression:
// Please ignore the above work. I am going to keep the above as proof of attempting the problem in hopes of receiving partial credit if my decided reasoning is considered incorrect.

From project 4a, the percentage of additional clock frequency was calculated by dividing the old clock period by the new clock period. We already know that the clock period of the new processor is 30% that of the old processor. We can therefore calculate the percentage increase using the following expression:

Old Clock Speed/(Old Clock Speed * 0.3) = 1/0.3 = approximately 3.3. We then subtract this term by 1 to find our increase:

3.3 – 1 = approximately 2.3.

Using the full terms, this means that we have a 233% increase in clock frequency.

We can then find the power consumption of the new chip by adding 500 milliwatts to 5 milliwatts * the percentage we found:

500 milliwatts + 5 milliwatts * 233 = 1665 milliwatts

B.

What is the performance per watt of the new processor concept relative to the current generation with and without the optimized compiler? (higher performance per Watt is better)

From project 4a, the performance per watt of the new processor concept relative to the current generation can be found using the following expression:

Performance Speedup / (power_new/power_old)

We found that the Performance Speedup without the optimized compiler was 3.3.

We also found that the Performance Speedup with the optimized compiler was 8.8.

We also found that the power consumption of the new processor was 1665 milliwatts.

We know that the power consumption of the old processor was 500 milliwatts.

We now plug and chug for performance per Watt of the unoptimized compiler:

3.3/(1665 milliwatts/500 milliwatts) = 3.3/3.3 = approximately 1. We see no performance per Watt improvement using the new processor with the unoptimized compiler.

We can plug and chug for performance per Watt of the optimized compiler:

8.8/(1665 milliwatts/500 milliwatts) = 8.8/3.3 = approximately 2.6. This is a clear improvement.

C.

The compiler optimizations do not need to be run on a pipelined processor. Furthermore, if the compiler were used on the single-cycle machine, it would not need to insert NOP instructions to avoid Load-Use hazards. If there are 32 of these NOPs in the code, how many instructions would be required to run the optimized code to completion on the single cycle machine?

By counting the instructions in the input_unrolled.txt file, which represents the running of the optimized code, and removing the nop instructions in the code, you are left with 67 instructions, including the done instruction.

What is the speedup and performance per Watt of this solution?

Using this information about the number of instructions needed to complete the optimized code execution, we also know that the single-cycle processor will take 67 cycles to execute. Therefore, our speedup would be the following:

CPU Time Old Single/CPU Time New Single = (270 Cycles * Single Cycle Time)/(67 Cycles * Single Cycle Time) = 270 Cycles/67 Cycles = approximately 4 time speedup.

To find the performance per Watt, recall that it can be represented by the following:

Performance Speedup / (power_new/power_old)

Our performance Speedup is 4.

Our power_old is 500 milliwatts.

Our power_new is still 500 milliwatts because we have not changed chips.

Therefore, our performance per Watt is the following:

4/(500 milliwatts/500 milliwatts) = 4/1 = 4.

D.

Based on your analysis, what is your recommendation to SensoMIPS?

While high performance is desirable, the main factor in determining what would be the best solution for SensoMIPS is the performance per Watt since this is most relevant to the investment that would be made to obtain worthwhile performance increases. Using the unoptimized compiler with the pipelined processor, we saw no improvement in performance per Watt. While this may not have any obvious negatives, there exists a cost in developing the new pipelined processor. Since the performance per Watt is only 1, it is not worth the investment using the unoptimized compiler. On the other hand, the optimized compiler gives us a performance per Watt of 2.6. This demonstrates that there is more bang for our buck in this solution. However, there still exists a cost in developing the pipelined processor. Lastly, we have the existing single-cycle processor running our optimized code. Because of the huge reductions in instructions and nops, we have a performance per Watt of 4. This has the most bang for our buck and does not require the need to develop a new processor. Simply running the optimized compiler on the single-cycle processor leads to massive performance gains and is more cost-effective in the long run. Therefore, my recommendation is that SensoMIPS use the single-cycle processor with the optimized compiler.