**Max Score = 15 points**

CS 250 2018 Spring Homework 12

This assignment is due at 11:59:00 pm Thursday, April 26, 2018.

Insert your typewritten answers into this file. You may include images of neatly hand drawn diagrams when appropriate. To have this assignment graded, upload your file to Blackboard in either PDF or Word format. You may upload more than once to permit correction of errors. Late submissions will receive a score of zero (0).

You are responsible for ensuring that your upload (1) is to the location in Blackboard for this assignment, and (2) is the file that you intend to have graded for this assignment, and (3) is not marked “LATE” by Blackboard. You are encouraged to verify your upload was successful by downloading your file from Blackboard and examining that download.

Please be aware that Blackboard determines if an upload is late using an internal timer that measures time in units of seconds, but reports time to its users rounded down to the minute. This means that one second after Blackboard first shows time as 11:59pm any completing upload will be marked LATE because the internal program time is then after 11:59:00pm. Upload sufficiently before the stated deadline to avoid being late due to an upload that completes after 11:59:00pm.

1. Text exercise 21.12. A hashing function places values in random locations in an array called a hash table. A programmer finds that even when memory caching is turned off, storing and then looking up 50,000 values in an extremely large hash table (16 megabytes) has worse performance than using the same data in a smaller hash table (16 kilobytes). Explain why. **The reason it has worse performance is because there is a tradeoff between look up time and space. The larger number of spaces makes it so there is a greater number of locations to store each item and cause less collision. However, having more space means it takes longer to go through the entire table. In a smaller hash table there is less space and more possibilities of collision but it can look up values faster.**
2. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 70% of the time, measured as a percentage of the execution time *when the enhanced mode is in use*. Recall that Amdahl’s Law depends on the fraction of the original, *unenhanced* execution time that could make use of the enhanced mode. Thus, we cannot directly use this 70% measurement to compute speedup with Amdahl’s Law.
   1. What is the speedup we have obtained from fast mode?

**Speedup = UnenhancedT/EnhancedT**

**UnenhancedT = 70%EnhancedT + 10 \* 70% EnhancedT = 7.7 EnhancedT**

**Speedup = 7.7 UnenhancedT/EnhancedT = 7.7**

* 1. What percentage of the original execution time has been converted to fast mode?  
     **7.7 \* 10 – 10/ 7.7 \* 10 – 7.7**

**67/69.3**

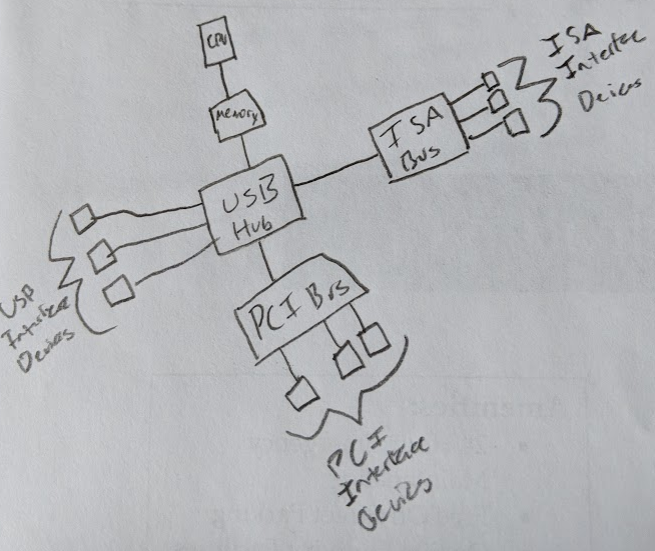
**96.68%**

1. Once a loop has been unrolled, which factor(s) of the CPU Time equation can be improved, worsened, or cannot be affected through the application of instruction scheduling? Explain.

**The unrolling of large loops can lead to increased code size and could therefore worsen it. However its primary uses is to increase speed by reducing or elimination instruction that control the loop which would therefore improve it. By adding instruction scheduling it will improve as it will allow for multiple instructions to be executed in parallel.**

**Loop unrolling improves CPI because it improves instruction level parallelism. However with instruction scheduling there is no improvement.**

**The clock cycle is worsened because loop unrolling increases the area of the program which negatively impacts the clock cycle time. This will become worse with instruction scheduling because multiple threads will execute together and increase time.**

1. Text exercise 22.3. A computer with a USB port contains hardware known as a USB hub that usually connects the external ports to a PCI bus. Modify the diagram in Figure 22.2 to show a USB hub.  
   
2. Text exercise 22.4. If a computer contains two buses connected by a transparent bridge, and the memory connects to one bus while the devices connect to the other, will the devices be able to communicate with memory? Explain. **Yes because the use of a bridge makes it possible for the two to communicate.**
3. The following information has been measured for the program **gcc**, the GNU Compiler Collection, running on a processor similar to ARM. This is real data on the dynamic frequency of instruction execution by the way. Do not be concerned that the frequencies do not total exactly to 100%.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Dynamic frequency | Instruction | Dynamic frequency | Instruction | Dynamic frequency |
| Load | 25.1% | Store | 13.2% | Add | 19.0% |
| Sub | 2.2% | Multiply | 0.1% | Compare | 6.1% |
| Load constant value | 2.5% | Conditional branch | 12.1% | Conditional move | 0.6% |
| Jump | 0.7% | Call | 0.6% | Return | 0.6% |
| Shift | 1.1% | AND | 4.6% | OR | 8.5% |
| XOR | 2.1% | Other logical | 0.4% |  |  |

Further, measurement on the processor reveals the following information.

|  |  |
| --- | --- |
| Instruction | Average clock cycles per instruction (CPI) |
| All ALU instructions | 1.0 |
| Loads and stores | 1.4 |
| Conditional branch if taken | 2.0 |
| Conditional branch if not taken | 1.5 |
| Jumps | 1.2 |

Assume that 60% of conditional branches are taken, and that “Conditional move” and “Other logical” instructions are handled by the ALU. Compute the effective CPI for this processor on the gcc program.

**Total ALU = 19 + 2.2 + .1 + 6.1 + 1.1 + 4.6 + 8.5 + 2.1 + .4 + .6 = 44.7%**

**Total Load/Store = 25.1 + 13.2 + 2.5 = 40.8%**

**Total taken = 60 \* 12.1 = 7.26%**

**Total not taken = 40 \* 12.1 = 4.84%**

**Total jump = .7 + .6 + .6 = 1.9%**

**Effective CPI = .447 \* 1 + .408 \* 1.4 + .0726 \* 2 + .0484 \* 1.5 + .019 \* 1.2 = 1.2588**

1. Consider the CPU time equation, where CPI means clock cycles per instruction:  
    CPU Time = (Instructions/Program) \* (CPI) \* (Seconds/Clock cycle).  
   Fill in the following table with any of – Improves, Worsens, No Effect, Unknown Effect – for each factor of the CPU time equation

|  |  |  |  |
| --- | --- | --- | --- |
| **Technique** | **Instr./Program** | **CPI** | **Sec/Clock** |
| Loop blocking | **Improve** | **No Effect** | **Worsen** |
| Loop unrolling alone; not in combination with instruction scheduling | **improve** | **improve** | **No effect** |
| Hardware forwarding | **Improve** | **Improve** | **Improve** |
| Adding a pipeline stage by splitting the slowest stage into two stages | **Improve** | **No Effect** | **Improve** |