CS 250 Spring 2017 Homework 08
Due 11:58pm Wednesday, March 29, 2017
Submit your typewritten file in PDF format to Blackboard.

1. Let store indirect be defined as

   STOREI (regX, offset, regY) # store the contents of regX into the memory
   # location pointed to by the address found in
   # memory at location offset+regY

   Devise a way to implement STOREI as a macro written in the instruction set architecture (ISA) of Figure 6.1. For full credit, your implementation of STOREI may use no more storage locations and no different storage locations that does a single store instruction and must be expressed in the form of a macro.

   Add offset and regY into regB of the store operation and then move regX to regA of the store operation. Then call the store operation.

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 50% 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 50% measurement to compute speedup with Amdahl's Law.
   a. What is the speedup we have obtained from fast mode?
   $$\frac{0.5}{\frac{0.5}{10}} = 10$$

   b. What percentage of the original execution time has been converted to fast mode?
   0.25

3. Assume that a computer has a physical memory organized into 64-bit words. Give the word address and offset within the word for each of the following byte addresses: 0, 9, 27, 31, 120, and 256.
   0 – 0x0
   9 – 0x1 + 1
   27 – 0x3 + 3
   31 – 0x3 + 7
   120 – 0xF
   256 – 0x20
4. Consider the following code snippet and Figure 11.6 from our text.

```
for (i = 0; i <= 1023; i = i + j) {
        array[i] =  array[i] + 7;
}
```

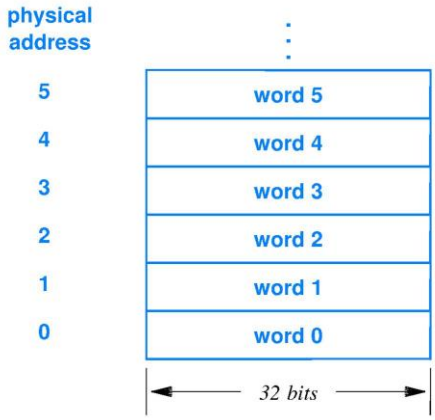

Figure 11.6 Physical memory addressing on a computer where a word is thirty-two bits. We think of the memory as an array of words.

Assume that the Figure 11.6 memory is built from a single memory chip that can be read during one CPU clock cycle but then requires 3 more CPU clock cycles before the chip is again ready to be accessed by the CPU. Assume that the constant 7 in the code snippet and the current values of variables i and j are available in CPU registers at all times. Finally, assume that the CPU stalls until array[i] can be read from data memory.

a. Describe the stalls that the CPU will experience due only to data memory read access of elements of array[i] as a function of the value of j for 1 <= j <= 1024. Ignore data memory write accesses; ignore all instruction fetch stalls (such as one due to a control hazard). With respect to execution time, what are the best values of j and the worst values?

   The CPU will stall every time the inner block of the for loop is executed because the inner block must always read and then written to.
   Best: j = 1023, the for loop executes only once causing 3 stalls.
   //Worst: j = 1, the for loop executes 1024 times causing 6*1024 stalls.

b. Now assume all is the same as in part (a) of this question except that the memory system configuration is now as shown in Figure 11.13.
   There will be 5*3 stalls for the read operation and 6*3 stalls for the write operation of each loop iteration.

   Best: j >= 1023, the for loop executes only once causing 33 stalls.

Worst: j = 1, the for loop executes 1024 times causing 36*1024 stalls.

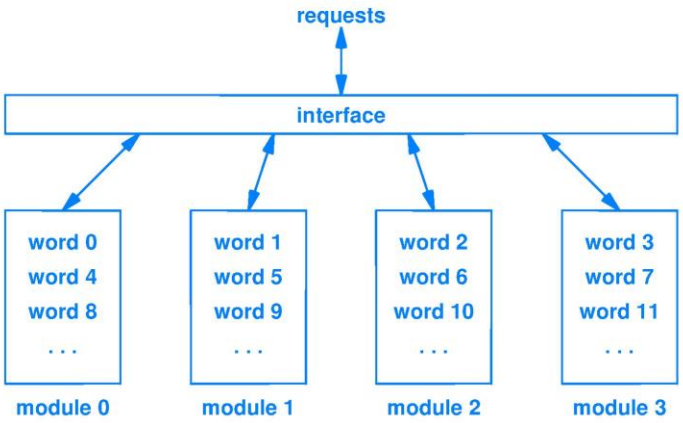

Figure 11.13 Illustration of 4-way interleaving that illustrates how successive words of memory are placed into memory modules to optimize performance.

Describe the stall cycles that the CPU will experience due only to data memory read access of elements of array[i] as a function of the value of j for 1 <= j <= 1024. Ignore data memory write accesses; ignore all instruction fetch stalls, such as one due to a control hazard. With respect to execution time what are the best values of j and the worst values?

5. The CPU time equation is as follows, CPI means Clock cycles per instruction: CPU Time = (Instructions/Program) * (CPI) * (Seconds/Clock cycle). For each of the three factors in the CPU Time equation, answer the following questions: (1) Can loop unrolling ALONE improve this factor, worsen this factor, or cannot affect this factor? (2) If loop unrolling either improves or worsens the factor, how does this occur?

   Loop unrolling would improve this factor because it decreases the number of instructions that gets executed because it reduces the number of times the processor needs to branch.

6. 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 (Instructions/Program) division can be improved because the instruction count will decrease.