Fall ’14 CIS 314 Assignment 4 – 120/100 points – Due Wednesday, 11/26, 11:59 PM

This assignment will involve solving problems and writing C related to optimizing program performance. For ease of submission, please submit a .zip file containing a single solution document for non-coding exercises (.txt, .doc, or .pdf) and individual source files for coding exercises (see naming conventions below). Your code and answers need to be documented to the point that the graders can understand your thought process. Problems will be graded based on work shown, not your final answer; full credit will not be awarded if no work is shown!

For problems 1-3 below, use the following code to aid in testing:

```c
#define VECTOR_SIZE 3
typedef long long data_t;
typedef data_t *vec_ptr;

data_t * get_vec_start(vec_ptr v) {
    return v;
}

int vec_length(vec_ptr v) {
    return VECTOR_SIZE;
}
```

1. [30] B&O’H 5.15. Write your answers in your solutions document. Also write a main() function to test the inner4 procedure. Your main() method should also print the time elapsed during execution of the inner4 function. Name your source file 5.15.c.

2. [20] B&O’H 5.16a. Write your answer for part A in your solutions document (you do not need to answer part B). Also write a modified version of inner4 as specified by 5.16 and a main() function to test the modified inner4 procedure. Your main() method should also print the time elapsed during execution of the inner4 function. Name your source file 5.16.c.

3. [20] B&O’H 5.17a. Write your answer for part A in your solutions document (you do not need to answer part B). Also write a modified version of inner4 as specified by 5.17 and a main() function to test the modified inner4 procedure. Your main() method should also print the time elapsed during execution of the inner4 function. Name your source file 5.17.c.
4. [10] Try running parts 1-3 above using various values for VECTOR_SIZE. What happens to the execution times of the three algorithms running on your machine as VECTOR_SIZE increases? Write your observations in your solutions document.


6. [+20] (Extra credit). Write a Y86 program to iteratively calculate numbers in the Fibonacci sequence. Your program should define a function "Fib" that takes an integer argument n and returns the nth Fibonacci number. Your function should calculate this number iteratively, NOT recursively. Also have a main section that first calls Fib(2) and then Fib(15). The program should end with the results of these calls stored in %ebx and %eax, respectively. Make sure to follow conventions for your function, with respect to passing parameters and returning values, as well as how you handle the %ebp and %esp. Additionally, you should follow x86 caller- vs. callee saved conventions (see Sec. 3.7.3).

Upload .zip file to Blackboard (see Assignments section for submission link).