

Homework#1 for ECE 152

Introduction (Chapter 1)

Due in class on Monday, January 28

All homework must be done in a group of 2 students. Each group should turn in one hard-copy in class. If your handwriting is unreadable, please type your homework.

1) [10 points] Sometimes software optimization can dramatically improve the performance of a computer system. Assume that a CPU can perform a multiplication ($X*Y$) in 10ns, an addition or subtraction in 1ns, and an exponentiation (X^Y) in 15ns. How long will it take for the CPU (without any software optimization) to calculate $c = a^2 + 2*a*b + b^2$? How would you optimize the equation so that it will take less time?

2) [10 points] A high level program can be translated by a compiler (and assembled) into any number of different, but functionally equivalent, machine language programs. (A simplistic and not particularly insightful example of this is that we can take the high level code $C=A+B$ and represent it with either `add C, A, B` or `add C, B, A`.) When you compile a program, you can tell the compiler how much effort it should put into trying to create code that will run faster. If you type `g++ -O0 -o myProgramUnopt myProgram.C`, you'll get unoptimized code. If you type `g++ -O3 -o myProgramOpt myProgram.C`, you'll get highly optimized code. Please perform this experiment on the program located on the ECE filesystem at `~sorin/ece152-public/myProgram.C`. Compile it both with and without optimizations. Compare the runtimes of each and write what you observe. (To time a program on a Unix machine, type "time myProgram", and then look at the number before the "u", as in the "0.40u" below. This number represents the time spent executing user code.)

```
sorin@carbon.ee.duke.edu [29] time myProgramUnopt
Finished
0.40u 0.02s 0:00.43 97.6%
```

3) [5 points] To see the differences between assembly languages, let's compile the same program as in Question 2 (but without assembling it) on computers with two different instruction set architectures (ISAs). To compile without assembling, use `g++ -S myProgram.C`, and it will produce `myProgram.s`. If you need help finding machines with different ISAs, I know that `login.ee.duke.edu` is a SPARC machine, and the `dsil` machines (e.g., `dsil3.ee.duke.edu`) are IA-32 (x86). What differences do you observe between these two assembly programs? I'm not expecting you to dissect the code, but rather look for big, obvious differences. To confirm that you did indeed do what I asked, please also include one line from each assembly program in your response to this question.

4) [5 points] Represent the integer $+149_{10}$ in 10-bit 2's complement.

5) [5 points] Represent the integer -149_{10} in 10-bit 2's complement.

6) [5 points] Represent “Go Duke!” in ASCII. Write your result in base 10. Pay careful attention to capital vs. lower-case letters as well as spaces.

7) [5 points] Consider the C-like code in Figure 1. What values do the last three lines print out? If it’s printing an address, tell me what variable’s address it is printing. Assume that the “print” statement “does what you’d expect” - it prints out the value of the address or variable after it.

8) [5 points] Processor P1 has a 1GHz clock, and processor P2 has a 2GHz clock. They have completely different instruction sets, but they both execute exactly 1 instruction per cycle. Which processor has better performance and why? You must explain your answer to get credit.

```
int p[20];
for (i=0; i<20; i++){
    p[i] = i;
}
int *mypointer = &p[0];
print *mypointer;
print p;
print p[3];
```

FIGURE 1. Code for Question 7