CIS429/529 Computer System Architecture
Problem Set 1

Book: J. L. Hennessy and D. A. Patterson. Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 2011 (5th edition)

General guidelines: Always state your assumptions and clearly explain your answers. Please upload your solution document (PDF or TXT) to Blackboard.

100/100 points possible – Due Friday, October, in class or by 11:59 PM through Blackboard.

Problem 1
Exercise 1.8 (p. 65).

Problem 2
Exercise 1.14 (p. 66-67)

Problem 3
Exercise 1.15 (p. 67)

Problem 4
Exercise 1.18 (p. 67)

Problem 5
Let us begin by considering the following C code.

```
int b;  //a global variable

void multiplyByB(int a){
  int i, result;
  for(i = 0; i<b; i++){
    result=result+a;
  }
}
```

Using gcc and objdump on a Pentium III, we see that the above loop compiles to the following x86 instruction sequence. (On entry to this code, register `%ecx` contains `i`, register `%edx` contains `result` and register `%eax` contains `a`. `b` is stored in memory at location `0x08047580`.) A brief explanation of each instruction in the code is given in **Supplement** (Below).

```
            xor    %edx,%edx
            xor    %ecx,%ecx
 loop:      cmp    0x08047580,%ecx
            jl     L1
            jmp    done
 L1:        add    %eax,%edx
            inc    %ecx
            jmp    loop
 done:    ...
```

How many bytes is the program? For the above x86 assembly code, how many bytes of instructions need to be fetched if b = 10? Assuming 32-bit data values, how many bytes of data memory need to be fetched? Stored?

Supplement

x86 has a CISC-style instruction set with variable-length instructions. In the x86 architecture, each instruction is capable of performing one or more simpler instructions called micro-operations (μops). It also supports several complex addressing modes.

We introduce a (very small) subset of the x86 instruction set in the following table. (Interested readers are referred to the Intel' s website for full details.)

Instruction	Operation	OF	SF	Length
`add` R_{DEST}, R_{SRC}	$R_{SRC} = R_{SRC} + R_{DEST}$	M	M	2 bytes
`cmp imm32,` R_{SRC2}	`Temp =` R_{SRC2} `- MEM[imm32]`	M	M	6 bytes
`inc` R_{DEST}	$R_{DEST} = R_{DEST} + 1$	M	M	1 byte
`jmp label`	jump to the address specified by `label`			2 bytes
`jl label`	if `(SF≠OF)` jump to the address specified by `label`	T	T	2 bytes
`xor` R_{DEST}, R_{SRC}	$R_{DEST} = R_{DEST}$ `xor` R_{SRC}	O	M	2 bytes

Table H2-1: Simple x86 instruction set (x86jr)

Notice that the jump instruction `jl` (jump if less than) depends on SF and OF, which are status flags. Each instruction affects them in different ways based on the result of its computation: "M" indicates the instruction modifying (writing) the status flag, "T" the instruction testing (reading but not writing) it, and "O" the instruction resetting it. A blank (as in `jmp` instruction) means that the instruction does not affect the status flag. Some instructions, like the `cmp` instruction, perform a computation and set status flags, but do not return any result.

The meanings of the status flags are given in the following table:

Name	Purpose	Condition Reported
OF	Overflow	Result exceeds positive or negative limit of number range
SF	Sign	Result is negative (less than zero)

Table H2-2: Status flags