CS164: Written Assignment 5
(On Code Generation)

Assigned: Thursday, Oct 14, 2004
Due: Thursday, Oct 21, 2004, at the beginning of class.

Grading and Submission

Your answers must be brief and easy to understand. Your grade (credit / no credit) will depend partly on how easy it is for us to understand and verify your answer. If the written assignment contains multiple questions (or parts of questions), **you must answer all questions to receive a pass grade.** Minor mistakes are acceptable. Submit your written assignments either in the classroom (before the lecture) or in 283 Soda. *No late homeworks are accepted.* **Please indicate your login name and Section number.**

1 Reverse-engineering compiler-generated code

This question asks you to understand assembly code generated by a compiler. Answering this question will help you (a) get familiar with the x86 assembler; (b) compare the code generation scheme presented in the lectures with a slightly optimized one.

The question: Write a Java program that would produce the x86 assembly code on the following page.

FYI, this code was generated by the GNU C compiler gcc on Solaris / x86. The gcc compiler had all optimizations turned off. Note that while this assembly code was generated from a C program, there exists a Java program that would correspond to this assembly code.

You can find a manual for the Intel x86 assembly language at

`http://www.intel.com/design/intarch/manuals/243191.htm`

Don't print this document, as it's several hundred pages long. Instead, use the Find feature of Acrobat Reader to locate the description of the unfamiliar instructions.

```
        .text
.globl main
        .type   main,@function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $24, %esp
        andl    $-16, %esp
        movl    $0, %eax
        subl    %eax, %esp
        movl    $10, -4(%ebp)
        cmpl    $2, -4(%ebp)
        jle     .L2
        movl    $3, 4(%esp)
        movl    -4(%ebp), %eax
        movl    %eax, (%esp)
        call    foo
        jmp     .L3
.L2:
        movl    $2, 4(%esp)
        movl    -4(%ebp), %eax
        movl    %eax, (%esp)
        call    foo
        movl    %eax, %edx
        leal    -4(%ebp), %eax
        addl    %edx, (%eax)
.L3:
        leave
        ret
.Lfe1:
        .size   main,.Lfe1-main
.globl foo
        .type   foo,@function
foo:
        pushl   %ebp
        movl    %esp, %ebp
        movl    8(%ebp), %eax
        imull   12(%ebp), %eax
        popl    %ebp
        ret
.Lfe2:
        .size   foo,.Lfe2-foo
```