T-107-TOLH, Homework Assignment IV

October 1, 2015

"An algorithm must be seen to be believed."

-Donald Knuth

Instructions:

- This assignment will be answered through Skel.
- You *must* hand in this assignment through Skel or it will **not** be graded.

Handout instructions: Connect to Skel using your favorite `ssh` client and unpack the assignment into your home directory by running the following command:

```
[student15@skel ~]$ tar xzvf /labs/tolh15/hw4/$(whoami)/hw4.tar.gz
[student15@skel ~]$ cd tolh15-hw4
[student15@skel tolh15-hw4]$ ls
answers  assignment  problem1  problem2  problem3  problem4  problem5
[student15@skel tolh15-hw4]$
```

To hand in the assignment, you **must** run the "./assignment handin" command inside the "hw4" directory. This will archive a copy of your assignment into a file called "/labs/tolh15/.handin/hw4/$(whoami)/handin.tar.gz". If the handin file does not exist, then you will not get a grade for this assignment.

```
[student15@skel ~]$ cd tolh15-hw4
[student15@skel tolh15-hw4]$ ./assignment handin
[student15@skel tolh15-hw4]$ ls /labs/tolh15/.handin/hw4/$(whoami)/handin.tar.gz
/labs/tolh15/.handin/hw4/student15/handin.tar.gz
```

To see when you last handed inn the assignment, then run the "./assignment check" command.

```
[student15@skel tolh15-hw4]$ ./assignment check
rutool check -c tolh15 -p hw4
Last handin: 2015-09-16 16:13:37
```

Before you start this homework assignment, you must first set your preferred editor using the "./assignment set editor" command. The classical choices are nano, vim, and emacs.

```
[student15@skel tolh15-hw4]$ ./assignment config set editor nano
# or
[student15@skel tolh15-hw4]$ ./assignment config set editor vim
# or
[student15@skel tolh15-hw4]$ ./assignment config set editor emacs
```

Every problem should be answered using the "./assignment" program which is located in the tolh15-hw4 directory. You can answer each question individually or all of them by running "./assignment" with the following parameters.

```
# Only answer problem1
[student15@skel tolh15-hw4]$ ./assignment problem1
# Answer all problems in the assignment
[student15@skel tolh15-hw4]$ ./assignment all
```

Question 1 (4 points)

Consider the assembly code on the left, and the C code on the right. Match the assembly code with the correct function.

(a)

```
fun:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax
    movl    12(%ebp), %edx
    cmpl    %edx, %eax
    cmovg   %edx, %eax
    popl    %ebp
    ret
```

_____

1.

```
int fun(int a, int b){
    unsigned ua = (unsigned) a;
    if (ua < b)
        return b;
    else
        return ua;
}
```

2.

```
int fun(int a, int b){
    if (b < a)
        return b;
    else
        return a;
}
```

3.

```
int fun(int a, int b){
    if (a < b)
        return a;
    else
        return b;
}
```

Question 2 (9 points)

(a)

```
bar:
    pushl    %ebp
    movl    %esp, %ebp
    movl    12(%ebp), %ecx
    imull    8(%ebp), %ecx
    movl    $13, %eax
    cltd
    idivl    %ecx
    popl    %ebp
    ret
```

(b)

```
bar:
    pushl    %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax
    movl    12(%ebp), %edx
    movl    %eax, %ecx
    shrl    $31, %ecx
    addl    %ecx, %eax
    sarl    %eax
    leal    (%eax,%eax,2), %ecx
    leal    (%edx,%edx,8), %eax
    leal    3(%ecx,%eax), %eax
    popl    %ebp
    ret
```

(c)

```
bar:
    pushl    %ebp
    movl    %esp, %ebp
    movl    12(%ebp), %edx
    leal    3(%edx), %eax
    testl    %edx, %edx
    cmovns    %edx, %eax
    sarl    $2, %eax
    addl    8(%ebp), %eax
    sall    $2, %eax
    popl    %ebp
    ret
```

1.

```
int foo(int a, int b)
{
    return 13 * (a / b);
}
```

2.

```
int foo(int a, int b)
{
    return 4 * (a + b/4);
}
```

3.

```
int foo(int a, int b)
{
    return 3*(a/2) + 9*b + 3;
}
```

4.

```
int foo(int a, int b)
{
    return 4 * (a * b);
}
```

5.

```
int foo(int a, int b)
{
    return 13 / (a * b);
}
```

Question 3 (15 points)

Consider the source code below, where `M` and `N` are constants declared with `#define`.

```
#define M (secret)
#define N (secret)

int mat1[M][N];
int mat2[N][M];

void copy_element(int i, int j)
{
    mat1[i][j] = mat2[j][i];
}
```

This generates the following assembly code:

```
copy_element:                          // ____________________________________
    pushl    %ebp                      // ____________________________________
    movl    %esp, %ebp                 // ____________________________________
    movl    8(%ebp), %eax              // ____________________________________
    movl    12(%ebp), %ecx             // ____________________________________
    movl    %eax, %edx                 // ____________________________________
    sall    $4, %edx                   // ____________________________________
    addl    %ecx, %edx                 // ____________________________________
    leal    (%ecx,%ecx,4), %ecx        // ____________________________________
    addl    %ecx, %eax                 // ____________________________________
    movl    mat2(,%eax,4), %eax        // ____________________________________
    movl    %eax, mat1(,%edx,4)        // ____________________________________
    popl    %ebp                       // ____________________________________
    ret                                // ____________________________________
```

What are the values of N and M?

Question 4 (20 points)

Consider the following IA32 code for a procedure `foo()`:

```
foo:                                    // _____________________________________
    pushl    %ebp                       // _____________________________________
    movl    %esp, %ebp                  // _____________________________________
    subl    $16, %esp                   // _____________________________________
    movl    12(%ebp), %eax              // _____________________________________
    movl    %eax, -4(%ebp)              // _____________________________________
    movl    12(%ebp), %eax              // _____________________________________
    subl    8(%ebp), %eax               // _____________________________________
    movl    %eax, -8(%ebp)              // _____________________________________
    jmp    .L2                          // _____________________________________
.L3:                                    // _____________________________________
    movl    $-4, %eax                   // _____________________________________
    subl    -8(%ebp), %eax              // _____________________________________
    addl    %eax, -4(%ebp)              // _____________________________________
    movl    -4(%ebp), %eax              // _____________________________________
    cltd                                // _____________________________________
    idivl    8(%ebp)                    // _____________________________________
    movl    %eax, -4(%ebp)              // _____________________________________
    subl    $1, -8(%ebp)                // _____________________________________
.L2:                                    // _____________________________________
    cmpl    $0, -8(%ebp)                // _____________________________________
    js    .L3                           // _____________________________________
    movl    -4(%ebp), %eax              // _____________________________________
    leave                               // _____________________________________
    ret                                 // _____________________________________
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: you may only use symbolic variables a, n, val, and i from the source code in your expressions below—do *not* use register names.)

```
int foo(int x, int y) {
    int result = ____________;
    for (int i = ____________ ; ______________________________ ; i = ____________) {
        ____________________;
        ____________________;
    }
    return result;
}
```

Question 5 (15 points)

In this problem, you are to implement a couple of functions in x86 assembly. Use your favourite text editor on skel to work on the solution file, `solution32.s`.

You are provided a tool to test your solutions for correctness. It will test your implementation with random integers and inform you whether your output was correct or not.

```
[student15@skel problem8]$ make
[student15@skel problem8]$ ./asm 32
```

A reference implementation of these functions have been made available to you in the C programming language.

```
int add (int a, int b) {
    /* Compute the sum of the two integers ‘a’ and ‘b’. */
    return a + b;
}

int sub (int a, int b) {
    /* Subtract ‘b’ from ‘a’. */
    return a - b;
}

int sum (int a, int b, int c, int d, int e, int f, int g) {
    /* Compute the sum of the seven integers given as parameters. */
    return a + b + c + d + e + f + g;
}

int max (int a, int b) {
    /* Return the larger of the two integers ‘a’ and ‘b’. */
    if (a > b)
        return a;
    else
        return b;
}

int cmp (int a, int b) {
    /* Return -1, 0, or 1, if ‘a’ is less than, equal to,
     * or greater than ‘b’, respectively. */
    if (a > b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}
```