CPSC 213, Winter 2015, Term 2 — Extra Questions

Date: March 3, 2015; Instructor: Mike Feeley

Answer in the space provided. Show your work; use the backs of pages if needed. There are **7** questions on **5** pages, totaling **62** marks.

Q1	/ 8
Q2	/ 6
Q3	/ 8
Q4	/ 8
Q5	/ 12
Q6	/ 10
Q7	/ 10

STUDENT NUMBER: _____

NAME: _____

SIGNATURE: _____

1 (8 marks) **Loops and If.** The following assembly code computes `s = a[0]` where `a` is a global, static array of integers. Modify this code so that it computes the sum of all positive elements of the array where the size of the array is stored in a global int named `n`. Your solution should avoid unnecessary memory accesses where possible (e.g., inside of the loop). You may modify the code in place. Comment every line you add. Hint: notice that you have to add four things: (1) read the value of `n`, (2) turn part of this code into a loop, (3) exit the loop at the right time, and (4) only sum positive numbers; you might want to take these one at a time.

2 (6 marks) **Static Control Flow.** Give SM213 assembly code for the following C statements. Assume that `i` is a global variable of type `int`.

2a
```
if (i==0)
    i = 1;
else
    i = 2;
```

2b
```
while (i!=0)
    i -= 1;
```

3 (8 marks) **Dynamic Control Flow.** Give SM213 assembly code for the following C statements. Assume that `i` is a global variable of type `int`.

3a Using a jump table, the statement:

```
switch (i) {
    case 4:
        i = 0;
        break;
    case 6:
        i = 1;
        break;
    default:
        i = 2;
        break;
}
```

3b Where the global variable `int (*bar)(void)` was previously declared, the statement:

```
bar();
```

4 (8 marks) **Procedure Calls.** Give SM213 assembly for these statements. Assume the `i` is a global variable of type `int`, that `r5` stores the value of the stack pointer, and that arguments are passed on the stack.

4a
```
int foo (int i, int j) {
    return j;
}
```

4b
```
i = foo (1, 2);
```

5 (12 marks) Consider the following SM213 assembly code that implements a simple C procedure.

```
L0:  deca r5                 #
     st   r6, (r5)           #
     ld   4(r5), r1          #
     ld   8(r5), r2          #
     ld   $0, r3             #
L1:  bgt  r2, L2             #
     br   L3                 #
L2:  dec  r2                 #
     ld   (r1, r2, 4), r4    #
     deca r5                 #
     st   r4, (r5)           #
     gpc  $2, r6             #
     j    *16(r5)            #
     inca r5                 #
     beq  r0, L1             #
     add  r4, r3             #
     br   L1                 #
L3:  mov  r3, r0             #
     ld   (r5), r6           #
     inca r5                 #
     j    (r6)               #
```

5a Comment every line in a way that illustrates the connection to corresponding C statements.

5b Give an equivalent C procedure (i.e., a procedure that may have compiled to this assembly code).

```
ld $a, r0              # r0 = &a = &[0]

ld $0, r1              # r1 = temp_i = 0

ld $0, r2              # r2 = temp_s = 0

ld (r0, r1, 4), r3     # r3 = a[temp_i]

add r3, r2             # temp_s = temp_s + a[temp_i]

ld $s, r4              # r4 = &s

st r2, (r4)            # s = temp_s
```

6 (10 marks) **Writing Assembly Code**. Write SM213 assembly code that implements the following C program. Use labels for static addresses but do not include variable label declarations (i.e. "`.long`" lines). Show only the code for these two procedures. Do not implement a return from `callReplace()`; simply halt at the end of that procedure. Do not use the stack. Comment every line.

```
int* a;
int  searchFor, replaceWith, size, i;

void replace() {
    for (i=0; i<size; i++)
        if (a[i]==searchFor)
            a[i]=replaceWith;
}
```

```
void callReplace() {
    replace();
    // halt; do not return
}
```

7 (10 marks) Implement the following in SM213 assembly. Pass arguments on the stack. You can use a register for `c` instead of a local variable. **Comment every line.**

```
int countNotZero (int len, int* a) {
    int c=0;
    while (len>0) {
        len=len-1;
        if (a[len]!=0)
            c=c+1;
    }
    return c;
}
```