CPSC 213, Winter 2010, Term 1 — Quiz 2

Date: Oct 20, 2010; Instructor: Tamara Munzner

NAME: ______________________________ **STUDENT NUMBER:** ________________

1 Consider the following C code.

```
int a[10] = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55};
int j = *(a+(&a[4]-&a[1]) + *(&a[5]-3));
```

Does the execution of this code generate a runtime error? If not, what is the value of j after the code executes? **Show your work/calculations to justify your answer.**

2 Give the SM213 assembly code that a compiler might generate for the function foo in the following C code; include comments. Assume that the location in memory for the variable a is 0x1000. You do not need to write down any .address/.pos specifications, just the assembly language commands.

```
struct A {
    int i;
    int j;
};
```

```
struct A* a;
void foo () {
    a->i = a->j + 3;
}
```

SM213 assembly code with comments.

3 Compare the following two alternatives implementing a similar computation. Recall that ”`strncpy (s1, s2, n)`” copies string s2 into strings s1.

```
char* foo () {                          char* foo () {
    char* x = (char*) malloc (11);          char* x = (char*) malloc (11);
    strncpy (x, "Tra la la!", 11);          strncpy (x, "Tra la la!", 11);
    return x;                               free (x);
}                                           return x;
                                        }
void bar () {                           void bar () {
    char* y = foo ();                       char* y = foo ();
}                                       }
```

Both versions of this code have a different bug. Carefully describe each bug (give its name if you can) and then fix the code.

3a Describe the bug on left-hand side.

3b Describe the bug on right-hand side.

3c Re-write the code so that neither bug is present.

OpCode	Format	Semantics/RTL	Eg Machine	Eg Assembly

load immediate	`0d--`
	`vvvvvvvv`
		$r[d] \leftarrow v$	`0100`
			`00000100`
				`ld $0x1000,r1`

load base+dis	`1osd`	$r[d] \leftarrow m[o \times 4 + r[s]]$	`1123`	`ld 4(r2),r3`
load indexed	`2sid`	$r[d] \leftarrow m[r[i] \times 4 + r[s]]$	`2123`	`ld (r1,r2,4),r3`
store base+dis	`3sod`	$m[o \times 4 + r[d]] \leftarrow r[s]$	`3123`	`st r1,8(r3)`
store indexed	`4sdi`	$m[r[i] \times 4 + r[d]] \leftarrow r[s]$	`4123`	`st r1,(r2,r3,4)`
halt	`f000`		`f000`	`halt`
nop	`ff00`		`ff00`	`do nothing (nop)`
rr move	`60sd`	$r[d] \leftarrow r[s]$	`6012`	`mov r1, r2`
add	`61sd`	$r[d] \leftarrow r[d] + r[s]$	`6112`	`add r1, r2`
and	`62sd`	$r[d] \leftarrow r[d]\ \&\ r[s]$	`6212`	`and r1, r2`
inc	`63-d`	$r[d] \leftarrow r[d] + 1$	`6301`	`inc r1`
inc addr	`64-d`	$r[d] \leftarrow r[d] + 4$	`6401`	`inca r1`
dec	`65-d`	$r[d] \leftarrow r[d] - 1$	`6501`	`dec r1`
dec addr	`66-d`	$r[d] \leftarrow r[d] - 4$	`6601`	`deca r1`
not	`67-d`	$r[d] \leftarrow !r[d]$	`6701`	`not r1`

shift	`7dss`	$r[d] \leftarrow r[d] << s$	`7102`
			`71fe`
				`shl $2, r1`
				`shr $2, r1`

branch	`8-oo`	$\mathrm{pc} \leftarrow \mathrm{pc} + 2 \times o$	`1000:  8004`	`br 0x1008`
branch if equal	`9roo`	if $r[r] == 0,\ \mathrm{pc} \leftarrow \mathrm{pc} + 2 \times o$	`1000:  9104`	`beq r1, 0x1008`
branch if greater	`aroo`	if $r[r] > 0,\ \mathrm{pc} \leftarrow \mathrm{pc} + 2 \times o$	`1000:  a104`	`bgt r1, 0x1008`

jump	`b---`
	`aaaaaaaa`
		$\mathrm{pc} \leftarrow a$	`b000`
			`00001000`
				`jmp 0x1000`

get program counter	`6f-d`	$r[d] \leftarrow \mathrm{pc}$	`6f01`	`gpc r1`
jump indirect	`croo`	$\mathrm{pc} \leftarrow r[r] + 2 \times o$	`c102`	`jmp 8(r1)`
jump double ind, b+disp	`droo`	$\mathrm{pc} \leftarrow m[4 \times o + r[r]]$	`d102`	`jmp *8(r1)`
jump double ind, index	`eri-`	$\mathrm{pc} \leftarrow m[4 \times r[i] + r[r]]$	`e120`	`jmp *(r1,r2,4)`