CS61c Spring 2015 Discussion 2 – C Memory Management & MIPS

1 C Memory Management

1. Match the items on the left with the memory segment in which they are stored. Answers may be used more than once, and more than one answer may be required.

   1. Static variables
   2. Local variables
   3. Global variables
   4. Constants
   5. Machine Instructions
   6. Data
   7. malloc()
   8. String Literals
   9. Characters

   A. Code

   B. Static

   C. Heap

   D. Stack

2. In which memory segment do the following reside? What is wrong with the following C code?

```
#define C 2
const int val = 16;
int constant = 42;
char arr[] = "foo";
void foo(int arg){
    char *str = (char *) malloc (C*val);
    char *ptr = arr;
    if(bear) str = (void *) malloc (10);
}
```

```
arg  [         ]     str  [         ]

arr  [         ]     *str [         ]

ptr  [         ]     C    [         ]

*ptr [         ]     val  [         ]

           42 [         ]
```

3. Write code to prepend (add to the start) to a linked list, and to free/empty the entire list.
   `struct r_node { struct r_node* next; int value; }`

`void free_r(struct r_node** lst)`	`void prepend(struct r_node** lst, int val)`

Note: `*lst` *points to the first element of the list, or is* `NULL` *if the list is empty.*

2 MIPS Intro

1. Assume we have an array in memory that contains `int* arr = {1,2,3,4,5,6,0}`. Let the value of `arr` be a multiple of 4 and stored in register `$s0`. What do the following programs do?

```
a) lw $t0, 12($s0)
   add $t1, $t0, $s0
   sw $t0, 4($t1)

b) addiu $s1, $s0, 27
   lh $t0, -3($s1)

c) addiu $s1, $s0, 24
   lh $t0$, -3($s1)

d) addiu $t0, $0, 12
   sw $t0, 6($s0)

e) addiu $t0, $0, 8
   sw $t0, -4($s0)

f) addiu $s1, $s0, 10
   addiu $t0, $0, 6
   sw $t0, 2($s1)
```

2. In 1), what other instructions could be used in place of each load/store without alignment errors?

3. What are the instructions to branch to `label:` on each of the following conditions?

`$s0 < $s1`	`$s0 <= $s1`	`$s0 > 1`	`$s0 >= 1`

3 Translating between C and MIPS

Translate between the C and MIPS code. You may want to use the MIPS Green Sheet as a reference. In all of the C examples, we show you how the different variables map to registers – you don't have to worry about the stack or any memory-related issues.

C	MIPS

`// $s0 -> a, $s1 -> b`
`// $s2 -> c, $s3 -> z`
`int a = 4, b = 5, c = 6, z;`
`z = a + b + c + 10;`	

`// $s0 -> int * p = intArr;`
`// $s1 -> a;`
`*p = 0;`
`int a = 2;`
`p[1] = p[a] = a;`	

`// $s0 -> a, $s1 -> b`
 	
`int a = 5, b = 10;`
`if(a + a == b) {`
`    a = 0;`
`} else {`
`    b = a - 1;`
`}`	

	`    addiu $s0, $0, 0`
	`    addiu $s1, $0, 1`
	`    addiu $t0, $0, 30`
	`loop:`
	`    beq $s0, $t0, exit`
	`    addu $s1, $s1, $s1`
	`    addiu $s0, $s0, 1`
	`    j loop`
	`exit:`

`// $a0 -> n, $v0 -> sum`
`int sum;`
`for(sum=0;n>0;sum+=n--);`	