Course Name: **Electronic Devices Lab**
Course Number and Section: 14:332:333:2A

Experiment: Lab Experiment 1 – Introduction to C

Lab Instructor: Jalal Abdulbaqi

Date Performed: 10/10/18

Date Submitted: 10/24/18

Submitted by: Vedanta Dhobley – 171002654

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 – B	A. Code
2. Local Variables – D	B. Static
3. Global Variables – B	C. Heap
4. Constants – A B D	D. Stack
5. Machine Instructions – A	
6. malloc() – C	
7. String Literals – B	

2. Write the code necessary to properly allocate memory (on the heap) in the following scenarios:
   a. An array arr of k integers:
   $$arr = (int\ *)malloc(sizeof(int) * k);$$
   b. A string str of length p:
   $$str = (char\ *)malloc(sizeof(char) * (p + 1));$$
   c. An n x m matrix mat of integers initialized to zeroes:
   $$mat = (int\ **)calloc\big(n, sizeof(int\ *)\big);$$
   $$for(int\ i = 0; i < m; i++)\{$$
   $$\quad mat[i] = (int\ *)calloc\big(m, sizeof(int)\big);$$
   $$\}$$

3. 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 snippets of RISC-V code do?
   a. Sets register t0 to arr[3]:
   $$lw\ t0{,}12(s0)$$
   b. Array element with pointer t2 incremented by 1:
   $$slli\ t1, t0{,}2$$
   $$add\ t2, s0, t1$$
   $$lw\ t3{,}0(t2)$$
   $$addi\ t3, t3{,}1$$
   $$sw\ t3{,}0(t2)$$
   c. Multiplies arr[0] by -1 and sets register t0 to its two's compliment:
   $$lw\ t0{,}0(s0)$$
   $$xori\ t0, t0{,}0xFFF$$
   $$addi\ t0, t0{,}1$$

4. What are the instructions to branch to label on each of the following conditions? The only branch instructions you may use are beq and bne.

$s0 < s1$	$s0 \leq s1$	$s0 > 1$
$slt\ t0, s0, s1$ $bne\ t0, 0, label$	$slt\ t0, s1, s0$ $beq\ t0, 0, label$	$sltiu\ t0, s0, 2$ $beq\ t0, 0, label$

5. Open the files lab3_ex5_c.c and lab3_ex5_assembly.s. The assembly code provided (.s file) is a translation of the given C program into RISC-V. Your task is to find/explain the following components of this assembly file.
   a. Register representing variable k
      t0 represents variable k and is set to 0 where each iteration increments it by 1.
   b. Registers behaving as pointers to destination arrays and source.
      t1 is set to source array address in main function therefore having it pointed to the source array just as t2 points to the destination array.
   c. Assembly code for loop found in C code.

$$
\begin{gathered}
slli\ t3, t0, 2 \\
add\ t4, t1, t3 \\
lw\ t5, 0(t4) \\
beq\ t5, x0, exit \\
add\ t6, t2, t3 \\
sw\ t5, 0(t6) \\
addi\ t0, t0, 1 \\
jal\ x0, loop
\end{gathered}
$$

      exit: Loop checks if $source[k] = 0$. Source[k] set to dest[k] incrementing k by 1 per iteration. After $beq\ t5, x0, exit$ this executes. Because we were returned the exit value the original equality must stand true.
   d. How pointers are manipulated in assembly.
      t1 increments by 0100_2 per iteration which after each the value is stored in t4 allowing t1 to retain its original value throughout the program. The value stored at the 0 offset in register t4 is stored in t5 so that the beq statement can be checked. The instruction set is similar for register t2 in which the destination address is stored. t2 is incremented by 0100_2 per iteration which after each it is stored in register t6 so that t2 can retain its original value throughout the program.

6. Translating between C and RISC-V Translate between the C and RISC-V code. You may want to use the RISC-V Reference Card for more information on the instruction set and syntax. 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. You may assume all registers are initialized to zero.

C	RISC-V
$//s0 \rightarrow a, s1 \rightarrow b$ $//s2 \rightarrow c, s33 \rightarrow z$ $int\ a = 4, b = 5, c = 6, z;$ $z = a + b + c + 10;$	$addi\ s0, x0, 4$ $addi\ s1, x0, 5$ $addi\ s2, x0, 6$ $add\ s3, s0, s1$ $add\ s3, s3, s2$ $addi\ s3, s3, 10$
$//\ s0 \rightarrow int * p = intArr;$ $//\ s1 \rightarrow a; * p = 0;$ $int\ a = 2;$ $p[1] = p[a] = a;$	$sw\ x0, 0(s0)$ $addi\ s1, x0, 2$ $sw\ s1, 4(s0)$ $slli\ t0, s1, 2$ $add\ t0, t0, s0$ $sw\ s1, 0(t0)$
$//\ s0 \rightarrow a, s1 \rightarrow b$ $int\ a = 5, b = 10;$ $if(a + a == b)\ \{$ $a = 0;$ $\}else\{$ $b = a - 1;$ $\}$	$addi\ s0, x0, 5$ $addi\ s1, x0, 10$ $add\ t0, s0, s0$ $bne\ t0, s1, else$ $xor\ s0, x0, x0$ $jal\ x0, exit$ $else:$ $addi\ s1, s0, -1$ $exit:$
$//\ s1 = 2\text{^}30$ $s1 = 1;$ $for(s0 = 0; s0 < 30; s++)\ \{$ $s1 *= 2;$ $\}$	$addi\ s0, x0, 0$ $addi\ s1, x0, 1$ $addi\ t0, x0, 30$ $loop:$ $beq\ s0, t0, exit$ $add\ s1, s1, s1$ $addi\ s0, s0, 1$ $jal\ x0, loop$ $exit:$
$//\ s0 \rightarrow n, s1 \rightarrow um$ $//\ assume\ n > 0\ to\ start$ $int\ sum;$ $for(sum = 0; n > 0; sum += n--);$	$addi\ s1, s1, 0$ $loop:$ $beq\ s0, x0, exit$ $add\ s1, s1, s0$ $add\ s0, s0, -1$ $jal\ x0, loop$ $exit:$

7. Implement a function factorial in RISC-V that has a single integer parameter n and returns n!. A stub of this function can be found in the file factorial. You will only need to add instructions under the factorial label, and the argument that is passed into the function is configured to be located at the label n. You may solve this problem using either recursion or iteration.

```
main:
   la t0,n
   lw a0,0(t0)
   jal ra,factorial
   addi a1,a0,0
   addi a0,x0,1
   ecall
   addi a0,x0,10
   ecall
factorial:
   li t1,1
   li t2,1
   bgt a0,t1,LABEL
   mul t2,t2,t1
LABEL:
   addi a0,t2,0
   jalr zero,ra,0
```