

## EE 2310 Homework #10 – Branch Instructions, Loops, and Code Segments

Name: \_\_\_\_\_

Student Number: \_\_\_\_\_

Below are snapshots of MIPS registers, a data declaration, and a SPIM memory dump from the data section. All register and memory contents are shown in hexadecimal form. Given the data shown below, answer the following questions. **NOTE: Changes to a register or memory location made in one problem to not carry to any other problem.**

|          |
|----------|
| .data    |
| n: .word |
| p: .word |
| q: .word |
| r: .word |
| s: .word |
| t: .word |
| u: .word |
| v: .word |
| w: .word |
| x: .word |
| y: .word |
| z: .word |
| str:     |
| .asciiz  |
| "hello   |
| world\n" |

### MIPS Registers

|                        |                         |                         |                          |
|------------------------|-------------------------|-------------------------|--------------------------|
| R0 (r0):<br>0x00000000 | R8 (t0):<br>0x0f0f0f0f  | R16 (s0):<br>0x00000000 | R24 (t8):<br>0x00000000  |
| R1 (at):<br>0x10010000 | R9 (t1):<br>0x0000ffff  | R17 (s1):<br>0x00000000 | R25 (t9):<br>0x00000001  |
| R2 (v0):<br>0x0000000b | R10 (t2):<br>0x00000000 | R18 (s2):<br>0x00000058 | R26 (k0):<br>0x00000000  |
| R3 (v1):<br>0x00000000 | R11 (t3):<br>0x10010020 | R19 (s3):<br>0x00000000 | R27 (k1):<br>0x00000000  |
| R4 (a0):<br>0x00000058 | R12 (t4):<br>0x100100f0 | R20 (s4):<br>0x00400020 | R28 (gp):<br>0x10008000  |
| R5 (a1):<br>0x10010010 | R13 (t5):<br>0x10010030 | R21 (s5):<br>0x00000000 | R29 (sp):<br>0x7ffffeff0 |
| R6 (a2):<br>0x0000000c | R14 (t6):<br>0x80000080 | R22 (s6):<br>0x800c1001 | R30 (s8):<br>0x00000000  |
| R7 (a3):<br>0x00000010 | R15 (t7):<br>0xffff0000 | R23 (s7):<br>0x00000050 | R31 (ra):<br>0x00400070  |

### Data

[0x10000000]...[0x1000fffc] 0x00000000

|              |            |            |            |            |
|--------------|------------|------------|------------|------------|
| [0x10010000] | 0x5350494d | 0x20697320 | 0x61207573 | 0x6566756c |
| [0x10010010] | 0x2073696d | 0x756c6174 | 0x6f722066 | 0x6f72206c |
| [0x10010020] | 0x6561726e | 0x696e6720 | 0x4d495053 | 0x20617373 |
| [0x10010030] | 0x68656c6c | 0x6f20776f | 0x726c640a | 0x00000000 |
| [0x10010040] | 0x726f6772 | 0x616d6d69 | 0x6e672061 | 0x6e642063 |
| [0x10010050] | 0x6f6d7075 | 0x74657220 | 0x61726368 | 0x69746563 |

[0x10010060]...[0x10020000] 0x00000000

1. In: beqz \$s1, after is the branch to memory location “after” taken? \_\_\_\_\_
2. In: bgez \$s6, next is the branch to memory location “next” taken? \_\_\_\_\_
3. In: blez \$t6, next is the branch to memory location “next” taken? \_\_\_\_\_
4. In: bltzal \$t7, next what is saved in \$ra (The name of what is saved)? \_\_\_\_\_
5. In the sequence: lw \$t8, w; lw \$t2, 16(\$t3); bge \$t2, \$t8, next , is the branch to memory location “next” taken? \_\_\_\_\_
6. In: bgt \$v0, \$a2, next is the branch to memory location “next” taken? \_\_\_\_\_
7. In: beq \$a0, \$s2, next is the branch to memory location “next” taken? \_\_\_\_\_
8. In the sequence: lw \$t8, w; lw \$t9, r; ble \$t8, \$t9, next , is the branch to memory location “next” taken? \_\_\_\_\_
9. In the sequence: lw \$t8, z; lw \$t9, p; blt \$t8, \$t9, next , is the branch to memory location “next” taken? \_\_\_\_\_
10. In: bne \$a0, \$s2, next , is the branch to memory location “next” taken? \_\_\_\_\_
11. In: bnez \$v1, next , is the branch to memory location “next” taken? \_\_\_\_\_
12. In the sequence: and \$t2, \$t7, \$s4; beqz \$t2, next , is the branch to memory location “next” taken? \_\_\_\_\_

13. The code sequence on the right is a procedure that performs an analysis. One (and only 1) instruction is wrong, so first correct that instruction. Then answer the following questions:

- 13.1. What is the program doing?
  
- 13.2. What is the resulting number in \$t3?
  
- 13.3. What was wrong with the program?

|                       |
|-----------------------|
| main: la \$a0, str    |
| move \$t3, \$0        |
| j look                |
| li, \$v0, 10          |
| syscall               |
| look: lb \$t0, (\$a0) |
| beqz \$t0, done       |
| beq \$t0, 0x61, count |
| beq \$t0, 0x65, count |
| beq \$t0, 0x69, count |
| beq \$t0, 0x6f, count |
| beq \$t0, 0x75, count |
| adup: addi, \$a0, 1   |
| j look                |
| count: addi, \$t3, 1  |
| j adup                |
| done: jr \$ra         |

14. If the contents of \$a2 are designated as “x,” write a brief program to compute  $12x^2 - 864$  and output it to the simulated console.

- 14.1. The result should end up in \$a0 (for output).
  
  - 14.2. What is the decimal answer in \$a0?
- 

|       |
|-------|
| .text |
| main: |
|       |
|       |
|       |
|       |
|       |
|       |
|       |

15. Review the code sequence on the right. The code segment between the two portions shown, that is actually executed, is omitted. Answer the following questions:

- 15.1. What sort of code segment is omitted?
  
- 15.2. Why is \$ra stored and what does this imply about the omitted segment?

|                         |
|-------------------------|
| go: sub \$sp, \$sp, 12  |
| sw \$a0, 4 (\$sp)       |
| sw \$s0, 8 (\$sp)       |
| sw \$ra, 12 (\$sp)      |
| (code segment omitted)  |
| back: lw \$a0, 4 (\$sp) |
| lw \$s0, 8 (\$sp)       |
| lw \$ra, 12 (\$sp)      |
| add \$sp, \$sp, 12      |
| jr \$ra                 |

