Name ____

CS/COE 0447 Example Problems for Exam 2
Fall 2010

1. This time, consider a non-leaf function `lisa`. This function has no arugments or return value. The return address is on the stack at offset 12 and the activation record is 12 bytes. Give a sequence of three MIPS instructions that cause a function return.

2. Suppose the stack pointer (`$sp`) has the value 0xFF0000020 and the activation record has two halfword fields. Give a single instruction that will load the second field in the activation record into register `$t0`.

For the next questions, consider the program code below (with line numbers):

```
                       # assume $sp = 0xFFFF0020
 0                     li    $s0,1
 1                     li    $s1,2
 2                     jal   _bart
 3                     j     quit
 4     _bart:          addi     $sp,$sp,-8
 5                     sw       $s0,0($sp)
 6                     sw       $ra,4($sp)
 7                     jal      _homer
 8                     lw       $ra,4($sp)
 9                     lw       $s0,0($sp)
10                     addi     $sp,$sp,8
11                     jr       $ra
12     _homer:         add      $sp,$sp,-4
13                     sw       $s1,0($sp)
14                     addi     $s1,$0,10
15                     move     $v0,$s1
16                     lw       $s1,0($sp)
17                     addi     $sp,$sp,4
18                     jr       $ra
19     quit:           ....
```

3. Give the value of `$sp` on each line from the code:

Line 5: `$sp`'s value is ________________________________

Line 8: `$sp`'s value is ________________________________

Line 13: `$sp`'s value is ________________________________

Line 19: `$sp`'s value is ________________________________

Name ____

4. Assume memory is all 0s. Fill in the table below to show the memory contents (as words) after the code above executes (i.e., when line 19 is reached):

Address	**Value at this Address**
0xFFFF0028	
0xFFFF0024	
0xFFFF0020	
0xFFFF001C	
0xFFFF0018	
0xFFFF0014	
0xFFFF0010	
0xFFFF000C	

5. Show the steps to multiply the 4-bit numbers 3 and 5 with the "fast shift-add multipler". Use the table below. List the multiplicand and product in binary. In the field "step", write "ADD" when the multiplicand is added. Write "SHIFT" to indicate when the product is shifted. In the iteration "Start" write the initial values for the mutiplicand and product. You may not need all steps (rows) in the table.

Iter.	**Multiplicand (M)**	**Product**	**Step**
Start			
1			
2			
3			
4			
5			
6			
7			

Name ____

6. Show the steps to multiply an 6-bit number 17 and 3 with Booth's algorithm. Use the table below. List the multiplicand and product in binary. In the field "step", write "ADD", "SUB", or "NO OP" to indicate which operation is done on each iteration.

Iter.	Multiplicand (M)	Product	Step
Start			
1			
2			
3			
4			
5			
6			
7			

7. Explain how the "fast shift-add multiply" improves over the original "slow shift-add multiply". Be sure to indicate what hardware changes make the "fast version" faster than the "slow version".

8. Suppose we want to do the computation S = A + B. A and B are positive 2's complement 8-bit binary numbers. Give a boolean expression that indicates whether there was an overflow when these numbers are added. To represent a certain bit *i* in *A*, *B* or *S*, use A_i, B_i or S_i. E.g., bit position 3 in *A* is A_3. Assume the bits are numbered 0 to 7 (right to left).

9. Give the negation in one's complement binary representation (5 bits) for the decimal numbers:

5d	Negation (in one's complement binary) ______________________
10d	Negation (in one's complement binary) ______________________
-15d	Negation (in one's complement binary) ______________________

Name ____

10. Give the negation in two's complement binary representation (5 bits) for the decimal numbers:

11d	Negation (in two's complement binary)	____________________
15d	Negation (in two's complement binary)	____________________
-13d	Negation (in two's complement binary)	____________________

11. Give Booth's encoding for the 8-bit numbers:

-19d	Booth's encoding	________________________________
27d	Booth's encoding	________________________________
62d	Booth's encoding	________________________________

12. Give *two reasons* to use Booth's algorithm (encoding) to improve the multiplication hardware.

13. Floating point numbers represent a "richer" set of values than integer numbers. Nevertheless, processors support integer numbers and programs frequently use them. What primary advantage does integer numbers and operations offer over floating point numbers and operations?

integer operations are significantly faster, programs frequently use discrete values thus, using integer for common operations/values offers a big performance benefit.

14. Using 1-bit adders, draw the circuit for a 4-bit ripple-carry addition unit.

15. Using 1-bit adders and 1-bit inverters (i.e., the *not* of a bit), draw a circuit for a 4-bit ripple-carry subtract unit.

16. Consider the boolean equation: A'BC + ABC + A'B'C. Give the truth table representation for this boolean equation.

17. Suppose you want to design a hardware circuit that has two inputs A and B, and one output O. The output O has the value 1 when exactly one input (A/B) is a 1. Give the truth table for this circuit design.

18. For question 15, what is the boolean equation for the truth table?

19. Here is a truth table. What is the boolean equation for O?

INPUTS			**OUTPUT**
A	**B**	**C**	**O**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0

1	1	0	0
1	1	1	0

20. Let's consider a 4-bit adder. If each 1-bit adder takes 2ns to compute an output (1-bit result and carry-out), how long does it take to compute the full 4-bit result?

21. Consider problem 17 again. This time, let's compute the time for subtraction (A-B). Suppose the 1-bit inverter (used to complement the B input of each 1-bit adder) takes 1ns. How long does it take to compute an answer with this subtraction unit? (*Be careful*: Think about whether the invert operations can be done simultaneously.)

22. Suppose you have a program P. This program executes 1000 regular instructions, 100 floating point multiply instructions, and 5 floating point square root instructions. Assume a regular instruction takes 10ns, a floating point multiply takes 100ns and and floating point square root takes 1000ns. What is the total amount of time (in ns) that the program takes to execute (the "execution time")?

23. Consider the program in question 18. Let's suppose we can improve the floating-point multiply to take 80ns or we can improve the floating-point square root to take 800ns. We cannot do both improvements. Compute the execution time with both improvements. Which improvement would you do?

24. Now, consider the fastest improvement from 19 and the original situation from problem 18. What is the speedup of the improvement versus the original case?