

**Question 1 (25 points):** The program in the next page was written in an attempt to simulate a Turing machine. The program violates the MIPS calling conventions, by design, for increased efficiency: register `$s0` is used to store a global variable and thus is not be saved/restored according to the conventions . Please answer the following questions:

1. **(5 points)** If the line that contains the comment `# print $a0` were to be replaced with a system call to print the value of `$a0`, which value would be printed?

The value printed would be one:

```

        $a0 <-- 0
----- 'lrlh'
right:
        $a0 <-- 1
----- 'rlh'
left:
        $a0 <-- 0
----- 'lh'
right:
        $a0 <-- 1
----- 'h'
left:
        (does not change $a0)

```

2. **(5 points)** There are two jump instructions that are commented as `end of recursion`. Only one of these two instructions is executed. What is the address of the instruction that will be executed?

The instruction at address `0x 8000 0030` will be executed, the other will not.

3. **(15 points)** Assume that when execution starts `$sp = 0x 1000 0000`. In the figure below indicate the address to which `$sp` will be pointing when the `end-of-recursion` instruction that you identified above is executed. Write the entire content of the stack when that instruction is executed.

	Address	Memory Word
	0x 1000 0014	
	0x 1000 0010	
	0x 1000 000C	
	0x 1000 0008	
	0x 1000 0004	
	0x 1000 0000	
	0x 0FFF FFFC	0x 8000 0018
	0x 0FFF FFF8	0x 8000 0078
	0x 0FFF FFF4	0x 8000 0040
\$SP →	0x 0FFF FFF0	0x 8000 0078
	0x 0FFF FFEC	

```

.data
0x4000 0000 tape: .string 'lrlh'
.text
0x8000 0000     la      $s0, tape
0x8000 0004     li      $a1, 'l'
0x8000 0008     li      $a2, 'r'
0x8000 000C     li      $a3, 'h'
0x8000 0010     add     $a0, $zero, $zero
0x8000 0014     jal     right
# print $a0
0x8000 0018     lw      $ra, 0($sp)
...
left:
0x8000 001C     addi     $sp, $sp, -4
0x8000 0020     sw      $ra, 0($sp)
0x8000 0024     lb      $t0, 0($s0)
0x8000 0028     addi     $s0, $s0, 1
0x8000 002C     bne     $t0, $a3, test_direction_l    # if $t0 != 'h' do not halt
0x8000 0030     j       left_return                  # halt (end of recursion)
test_direction_l:
0x8000 0034     addi     $a0, $a0, -1
0x8000 0038     beq     $t0, $a1, goto_left          # if $t0 == 'l' then go to left
0x8000 003C     jal     right                        # else go right
0x8000 0040     j       left_return
goto_left:
0x8000 0044     jal     left
left_return:
0x8000 0048     lw      $ra, 0($sp)
0x8000 004C     addi     $sp, $sp, 4
0x8000 0050     jr      $ra

right:
0x8000 0054     addi     $sp, $sp, -4
0x8000 0058     sw      $ra, 0($sp)
0x8000 005C     lb      $t0, 0($s0)
0x8000 0060     addi     $s0, $s0, 1
0x8000 0064     bne     $t0, $a3, test_direction_r    # if $t0 != 'h' do not halt
0x8000 0068     j       right_return                # halt (end of recursion)
test_direction_r:
0x8000 006C     addi     $a0, $a0, 1
0x8000 0070     beq     $t0, $a2, goto_right        # if $t0 == 'r' then go to right
0x8000 0074     jal     left                        # else go left
0x8000 0078     j       right_return
goto_right:
0x8000 007C     jal     right
right_return:
0x8000 0080     lw      $ra, 0($sp)
0x8000 0084     addi     $sp, $sp, 4
0x8000 0088     jr      $ra

```