Review 1

PROBLEM 1

1) Consider the following code that copies one string in the memory to another memory location
strcpy:

```
addi $sp, $sp, -4 # adjust stack for 1 item
sw $s0, 0($sp) # save $s0
add $s0, $zero, $zero # i = 0
L1:   add $t1, $s0, $a1 # addr of y[i] in $t1
lbu $t2, 0($t1) # $t2 = y[i]
add $t3, $s0, $a0 # addr of x[i] in $t3
sw $t2, 0($t3) # x[i] = y[i]
beq $t2, $zero, L2 # exit loop if y[i] == 0
addi $s0, $s0, 1 # i = i + 1
j L1 # next iteration of loop
L2:  lw $s0, 0($sp) # restore saved $s0
addi $sp, $sp, 4 # pop 1 item from stack
jr $ra # and return
```

   a) Is this a leaf or non-leaf procedure?
   b) Which instruction format is used in every instruction?
   c) Assuming that the string has three characters, how many times would every instruction be executed?
   d) Assuming a single clock cycle processor, how many clock cycles are needed to execute this code?
   e) In a pipelined processor, which instructions are prone to pipelining hazards? For each of these instructions (if any)
      i) Identify the type of hazard and indicate whether processor stalls can be avoided or not?
      ii) Which technique is used to avoid the hazard?

1) leaf procedure as it does not call external procedures or perform recursive calls.
2) I I R R I R I I I J I I R
3) 1 1 1 4 4 4 4 4 3 3 1 1 1
4) 32 cycles
5)

sw $s0, 0($sp)	prone to data hazard	solved by bypassing (forwarding)
add $t1, $s0, $a1	prone to data hazard	solved by bypassing (forwarding)
lbu $t2, 0($t1)	prone to data hazard	solved by bypassing (forwarding)
add $t3, $s0, $a0	prone to data hazard	solved by bypassing (forwarding)
sw $t2, 0($t3)		
beq $t2, $zero, L2	prone to control hazard	can be avoided with proper prediction

PROBLEM 2

1) Consider the following C statements
   int x1=500;
   float x2=500;
   float x3=500.25;
   Write down the corresponding binary value stored for x1, x2, and x3.

1) 500 = 111110100
2) 500 = 1.11110100 x 2^8
   S=0, EXPONENT = 135 = 10000111, Fraction = 1111 0100 0000 0000 0000 000
3) 500.25 = 111110100.01 = 1.1111010**01** x 2^8
   S=0, EXPONENT = 135 = 10000111, Fraction = 1111 0100 0100 0000 0000 000

Double precision format: 11-bit exponent and 52-bit fraction