Computer Organization Test 2

Question 1. Consider the following "doSomething" subprogram that utilizes a function "calculateSomething" and a "printResult" subprogram.

```
procedure doSomething (integer start, integer end, integer Z)
    local integer variables: count, result, sum
    sum = 0
    for count = start to end do
        result = calculateSomething(Z, sum, count)
        sum = sum + result
        printResult(count, sum)
    end for
end doSomething
```

a) (3 points) Using the MIPS register conventions ($a0-$a3, $t0-$t9, $s0-$s7, $v0-$v1, $sp, $ra), what registers would be used to pass each of the following parameters into doSomething:

start	end	Z
$a0	$a1	$a2

b) (3 points) Using the MIPS register conventions, which of these parameters ("start", "end", "Z") should be moved into $s-registers? end to $s1 and Z to $s2, but not start since it only used to initialize the local variable count.

c) (3 points) Using the MIPS register conventions, what registers should be used for each of the local variables:

count	result	sum
$s0	$v0	$s3

d) (26 points) For the registers indicated above, write the assemble language code for the complete subprogram doSomething. (You do not need to write the calculateSomething function or the printResult subprogram code, just include the code to call them.)

```
# end is in $s1
# Z is in $s2
# count is in $s0
# sum is in $s3
# result is in $v0

doSomething:
    sub $sp, $sp, 20
    sw $ra, 4($sp)
    sw $s0, 8($sp)
    sw $s1, 12($sp)
    sw $s2, 16($sp)
    sw $s3, 20($sp)
    move $s1, $a1
    move $s2, $a2
    li $s3, 0
for_initialize:
    move $s0, $a0
for_loop:
    bgt $s0, $s1, end_for
    move $a0, $s2
    move $a1, $s3
    move $a2, $s0
    jal calculateSomething
    add $s3, $s3, $v0
    move $a0, $s3
    move $a1, $s3
    jal printResult
    addi $s0, $s0, 1
    j for_loop
end_for:
    lw $ra, 4($sp)
    lw $s0, 8($sp)
    lw $s1, 12($sp)
    lw $s2, 16($sp)
    lw $s3, 20($sp)
    addi $sp, $sp, 20
    jr $ra
end_doSomething:
```

Question 2. (29 points) Translate the following high-level language code segment to MIPS assembly language. Use the registers indicated in the code.

```
$3 = 5
while $3 < $4 do
    if ($3 >= $2) OR ($2 >= 50) then
        $2 = $2 + $3
    else if ($4 < $5) AND ($2  < 30) then
       $2 = $2 - $4
    else
        $5 = $5 + 10
    end if
    $3 = $3 * 2
end while
```

```
    li  $3, 5
while:
    bge $3, $4, end_while
if:
    bge $3, $2, then
    blt   $2, 50, else_if
then:
    add $2, $2, $3
    j end_if
else_if:
    bge $4, $5, else
    bge  $2, 30, else
    sub $2, $2, $4
    j end_if
else:
    addi $5, $5, 10
end_if:
    mul $3, $3, 2
    j while
end_while:
```

Question 3. (6 points) Suppose you have the following .data area in MIPS assembly language:

```
                .data
array:          .word  10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
```

For each of the following assembly language segments, what value is loaded into register $t2?

a) li $t0, 3
la $t1, array
mul $t3, $t0, 4
add $t1, $t1, $t3
lw $t2, 0($t1)
	b) la $t1, array
	lw $t2, 16($t1)
		c) li $t0, 5
		la $t1, array
		sll $t3, $t0, 2 # shift left logical
		add $t1, $t1, $t3
		lw $t2, 8($t1)

13	14	17

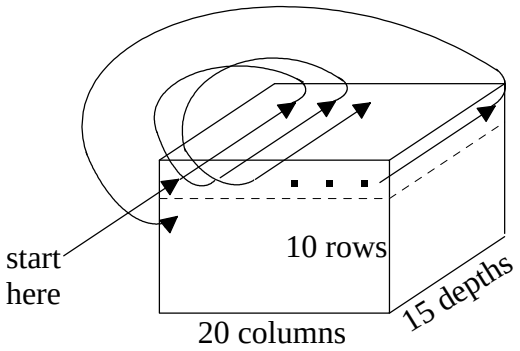
Question 5. (15 points) Complete the translation of the following high-level code segment to MIPS assembly language.

High-level Code Segment:	**MIPS Assembly Language to Complete:**
sum = 0	.data
for i = 0 to 10 do	array: .word 3, 10, 2, 4, 5, 5, 4, 20, 2, 3, 4
sum = sum + array[i]	sum: .word 0
array[i] = sum	.text
end for	.globl main
	main:
	li $t3, 0 # sum is in $t3
	la $t1, array # base addr. of array
	for:
	li $t0, 0 # i is in $t0
	for_compare:
	bgt $t0, 10, end_for
	lw $t2, 0($t1)
	add $t3, $t3, $t2
	sw $t3, 0($t1)
	addi $t1, $t1, 4
	addi $t0, $t0, 1
	j for_compare
	end_for:

b) After execution of the above code, what values will be in the array?

	0	1	2	3	4	5	6	7	8	9	10
array:	3	13	15	19	24	29	33	53	55	58	62

Question 6. (15 points) Assume that a 10 row x 20 column x 15 depth, three-dimensional array A is stored in memory as shown in the diagram. Write a formula to calculate the address of element A[r][c][d].



address of A[r][c][d] = base-addr. of A + r * 20 * 15 * 4 + c * 15 * 4 + d * 4

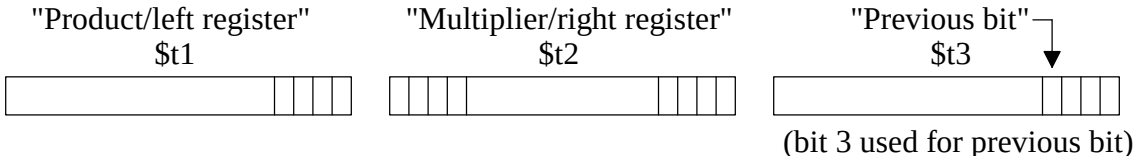
Question 7. (10 points) Consider speeding up Booth's algorithm by looking at the **4** least-significant bits of the multiplier and the previous bit. Complete the following partial table describing the value to be added to the "left register". Let M represent the value of the multiplicand.

Least-significant Bits of the Multiplier				Previous Bit	Amount added to left register
bit 3	bit 2	bit 1	bit 0		
0	0	0	0	0	0 x M
0	0	0	0	1	+1 x M
0	0	0	1	0	+1 x M
0	0	0	1	1	+2 x M
0	0	1	0	0	+2 x M
0	0	1	0	1	+3 x M
. . .					
1	1	1	0	0	-2 x M
1	1	1	0	1	-1 x M
1	1	1	1	0	-1 x M
1	1	1	1	1	0 x M

b) If we are multiplying two 32-bit numbers, how many times should we loop? 8

c) When the "left" and "right" registers are shifted, how many bit positions should we shift? 4

d)



Assume the above registers are used, write the MIPS assembly code to shift the "left" and "right" registers and update the previous bit.

(i) code to update the previous bit (bit 3 in register $t3)

```
andi  $t3, $t2, 8
```

(ii) code to update $t2

```
srl  $t2, $t2, 4
andi  $t4, $t1, 15
ror   $t4, $t4, 4
or    $t2, $t2, $t4
```

(iii) code to update $t1

```
sra   $t1, $t1, 4
```