Cairo University

Faculty of Computers and Artificial Intelligence

CS322

Computer Architecture and Organization

Assignment-3

Instructor: Dr. Mohammad El-Ramly

Team members :

Names	Emails	IDs
Mohamed Khalid Mohamed	**m.tawaty96@gmail.com**	20170230
Abdelrahman Nasr Abdelsalam	**abdelrahman.nasr137@gmail.com**	20170343
Bahaa El-Deen Osama Sayed	**BEOsamaCS98@gmail.com**	20170078

Set 1: 20170230

Problem 1:

Data:

a. $r1 : g

b. $r2 : h

```
blt $r1 , $r2 , ifelse

add $r1 , $r2 , $r2

b       fin     #go to (jump)

ifelse:

        bgt $r1 , $r2 , else

sub $r1 , $r1 , $r2

b       fin

else:

        mult $r1 , $r1 , $r2

fin:
```

Problem 2:

Data:

- **a. $t0 : i**
- **b. $t1 : size**
- **c. $t6 : Upper case**
- **d. $t7 : Lower case**
- **e. Latter A & Z are $s1 & $s2**
- **f. Base address of array is $s9**

```
Addi $t0 , $t0 , 0
for:
        bgt $t0, $t1 , End
        sll $t2 , $t2, 2          # i*4
add $t5, $t0, $s9
lw $t4, 0($t5)
ble $t4, $s1, L1        #t4 <= s1
bge $t4, $s2, L1
addi $t6, $6 , 1        #Upper++
j       for
L1:
addi $t7, $t7, 1
j       for
End:
```

Problem 3:

Data:

a. $t0 : i

b. $t1 : a

c. $t2 : j

d. $t3 : b

e. Base address of C in $s5

```
addi $t0, $t0, 0
addi $t2, $t2, 0
L2:
        Bgt $t0, $t1, L1        #if (I < a)
Addi $t5, $t0, 1        # i++
Bgt $t2, $t3, L2        # if(j < b)
Addi $t2, $t2, 1        #j++
Addi $t4, $t5, 2
Mult $t6, $t4, $t0
Sll $t7, $t6, 2
Add $s0, $t7, $s5
Lw $t8, 0($s)
Sub $t8, $t0, $t2
L1:
```

2- **Data:**

a. $v0 : t1

```
Main:

Li $a0, 10       #Argument

Jal      sumodd        #jumb and link

Jal      printf

Sunodd:

Addi $t1, $t1, 0        #i=0

Addi $t2, $t2, 0        #result =0

For:

Bge $t1, $a0, L1        #i  < n (for)

Addi $t1, $t1, 1        # i++

Addi $t0, $t0, 2        #t0 = 2

Rem $t4, $t1, $t0       # remander t4 = I % 2

Addi $t5, $t5, 1        # t5 = 1

Bne $t4, $t5, for       #if (I % 2 == i)

Add $t6, $t2, $t1       # result+=i

Add $v0, $v0, $t6       #jump

J        for

LI:

Jr    $ra        # return
```

Problem 4:

Data:

a. $v0 : t1

```
Main:

Li $a0, 10      #Argument

Jal     sumodd          #jumb and link

Jal     printf

Sunodd:

Addi $t1, $t1, 0        #i=0

Addi $t2, $t2, 0        #result =0

For:

Bge $t1, $a0, L1        #i  < n (for)

Addi $t1, $t1, 1        # i++

Addi $t0, $t0, 2        #t0 = 2

Rem $t4, $t1, $t0       # remander t4 = I % 2

Addi $t5, $t5, 1        # t5 = 1

Bne $t4, $t5, for       #if (I % 2 == i)

Add $t6, $t2, $t1       # result+=i

Add $v0, $v0, $t6       #jump

J       for

LI:

Jr    $ra       # return
```

Problem 5:

Data:

```
.data

# 0 01111111 00000000000000000000000 = 1
        num1: .word 0x3f800000

        # 0 10000001 00000000000000000000000 = 4
        num2: .word 0x40800000

mmask:  .word 0x007FFFFF

emask:  .word 0x7F800000

leading_bit:   .word 0x00800000              # 1

overflow_bit:   .word 0x01000000

.text

main:
                # loading the two numbers
                 la $t0, num1
                la $t1, num2
                 lw $s0, 0($t0)
                 lw $s1, 0 ($t1)

flpmultiplaying:

lw $t4,mmask                    # load mantissa mask

        and $t0,$s0,$t4         # extract mantissa from $s0 (a)

 and $t1,$s1,$t4        # extract mantissa from $s1 (b)

 lw $t4, leading_bit            # load implicit leading 1

 or $t0,$t0,$t4         # add the implicit leading 1 to mantissa

        or $t1,$t1,$t4          # add the implicit leading 1 to mantissa

 lw $t4,emask           # load exponent mask

 and $t2,$s0,$t4        # extract exponent from $s0 (a)

 srl $t2,$t2,23         # shift exponent right
```

```
 and $t3,$s1,$t4        # extract exponent from $s1 (b)
 srl $t3,$t3,23         # shift exponent right
match:
beq $t2,$t3,multsig   # check whether the exponents match
            bgeu $t2,$t3,shiftb    # determine which exponent is larger
shifta: sub $t4,$t3,$t2        # calculate difference in exponents
 srav $t0,$t0,$t4              # shift a by calculated difference
 add $t2,$t2,$t4               # update a's exponent
 j      multsig                # skip to the mult
            shiftb:
sub $t4,$t2,$t3         # calculate difference in exponents
      srav $t1,$t1,$t4                # shift b by calculated difference
 add $t3,$t3,$t4               # update b's exponent (not necessary)
multsig:
mult $t5,$t0,$t1        # multiplying the mantissas
norm:
 lw $t4, overflow_bit          # load mask for bit 24 (overflow bit)
 and $t4,$t5,$t4               # mask bit 24
      beq $t4,$0,done                 # right shift not needed because bit 24=0
 srl $t5,$t5,1                 # shift right once by 1 bit
 addi $t2,$t2,1                 # increment exponent
done:
lw $t4,mmask                  # load mask
      and $t5,$t5,$t4                 # mask mantissa
 sll $t2,$t2,23                # shift exponent into place
```

```
    lw $t4,emask            # load mask
and $t2,$t2,$t4             # mask exponent
    or $v0,$t5,$t2          # place mantissa and exponent into $v0
jr $ra                      # return to caller
```

Set 2: 20170343

Problem 1:

```
# g = $s0
# h = $s1

.text
.globl main

main:
	bgt $s0, $s1, else # g <= h
	ble $s0, $0, else # g > 0
	move $s0, $s1 # g = h

exit:
	li $v0, 10
	syscall

else:
	move $s1, $s0 # h = g
	j exit
```

Problem 2:

```
.data
        string: .asciiz "Utility"
        vowels: .asciiz "AIEOUaieou"

.text
.globl main

main:
        la      $t0, string
        la      $s6, vowels
        li      $s7, 0 # counter

string_loop:
        lb   $s0, ($t0) # $s0 = i-th character
        addi $t0, $t0, 1 # move pointer to the next character
        beq  $s0, $0, exit # character == null
        move $t1, $s6 # reset the addressof vowels

vowels_loop:
        lb      $s1, ($t1) # $s1 = i-th vowel
        addi $t1, $t1, 1 # move pointer to the next vowel
        beq  $s1, $0, string_loop # done of all vowels
        bne  $s1, $s0, vowels_loop # character isn't a vowel
        addi $s7, $s7, 1 # increment counter
        j    vowels_loop # check for next vowel

exit:
        li $v0, 10
        syscall
```

Problem 3:

```
.data
        c: .word 1, 2, 3

.text
.globl main

main:
        li      $t0, 3 # a = 3
        li      $t1, 0 # i = 0
        la      $s0, c # $s0 = &c[0]
        move $t3, $s0 # loads &c[0]

first_loop:
        bge  $t1, $t0, exit # i < a
        move $t2, $0 # j = 0
        lw      $s1, ($t3) # load c[i] value

second_loop:
        bge     $t2, $t1, intermediate # j < i
        add $s1, $s1, $t2 # c[i] += j
        addi $t2, $t2, 1 # j++
        j   second_loop

intermediate:
        sw      $s1, ($t3) # store to c[i]
        addi $t3, $t3, 4 # move pointer to next word
        addi $t1, $t1, 1 # i++
        j   first_loop


exit:
        li $v0, 10
        syscall
```

Problem 4:

```
$v0 # !isEven(n)
	andi $v.data
	even: .asciiz "even"
	odd: .asciiz "odd"

.text
.globl main

main:
	li	$s1, 10
	move $a0, $s1
	jal  isOdd
	move $t1, $v0
	beq  $t1, $0, print_even # t1 ? "odd" : "even"
	la   $a0, odd
	li	$v0, 4
	syscall
	j	exit

isOdd:
	addi $sp, $sp, -4 # push ra
	sw	$ra, 0($sp)
	jal  isEven
	not  $v0, 0, $v0, 0x1 # mask to git LSB
	lw	$ra, 0($sp) # pop ra
	add  $sp, $sp, 4
	jr	$ra

isEven:
	li   $t0, 2
	div  $a0, $t0
	mfhi $t0 # $t0 = $a0 % 2
	seq  $v0, $t0, $0 # $t0 == 0
	jr   $ra

print_even:
	la   $a0, even
	li	$v0, 4
	syscall

exit:
	li $v0, 10
	syscall
```

Problem 5:

```
.data
        # 0 01111111 00000000000000000000000 = 1
        num1: .word 0x3f800000

        # 0 10000001 00000000000000000000000 = 4
        num2: .word 0x40800000

        # 0 00000000 11111111111111111111111 = man_mask
        # 0 11111111 00000000000000000000000 = exp_mask
        # 0 00000001 00000000000000000000000 = leading1
        # 0 00000010 00000000000000000000000 = overflow
         man_mask: .word 0x007FFFFF
        exp_mask: .word 0x7F800000
        leading1: .word 0x00800000
        overflow: .word 0x01000000

.text
.globl main

main:
        # loading the two numbers
        la $t0, num1
        la $t1, num2
        lw $s0, ($t0)
        lw $s1, ($t1)

        # getting mantisas
   lw  $t4, man_mask
   and $t0, $s0, $t4
   and $t1, $s1, $t4

   # prepending leading 1 to mantisas
   lw $t4, leading1
   or $t0, $t0, $t4
   or $t1, $t1, $t4

   # getting exponent
   lw  $t4, exp_mask
   and $t2, $s0, $t4
   srl $t2, $t2, 23
   and $t3, $s1, $t4
   srl $t3, $t3, 23

   # which exponent is larger
```

```
   beq  $t2, $t3, add_mantisas # add mantisas directly if equal
   bgeu $t2, $t3, shift_num2 # shift num2 mantisa (if larger)

         # shift num1 mantisa (if larger)
   sub  $t4, $t3, $t2
   srav $t0, $t0, $t4
   add  $t2, $t2, $t4
   j    add_mantisas

shift_num2:
   sub  $t4, $t2, $t3
   srav $t1, $t1, $t4
   add  $t3, $t3, $t4

add_mantisas:
         add $t5, $t0, $t1

         # normalize and adjust exponent if necessary
   lw   $t4, overflow
   and  $t4, $t5, $t4
   beq  $t4, $0, assemble
   srl  $t5, $t5, 1
   addi $t2, $t2, 1

assemble:
         # assemble back into the floating-point format
   lw  $t4, man_mask
   and $t5, $t5, $t4
   sll $t2, $t2, 23
   lw  $t4, exp_mask
   and $t2, $t2, $t4
   or  $s2, $t5, $t2 # save result to $s2 (expected 0x40a00000 = 5)

exit:
         li $v0, 10
         syscall
```

Set 3: 20170078

Problem 1:

```
.data

	#if (g >= h)
		#  g++;
	#else
		#  g--;

# s0 = g
# s1 = h

.text

main:
     slt $t0, $s0, $s1 # (g >= h) (Using DeMorgan's Law)
     bne $t0, $0, exit
     addi $s0, $s0, 1 # g++
     j done
     exit: addi $s1, $s1, -1 # h--
     done:
     #End Program :
     li $v0,10
     syscall
```

Problem 2:

```
.data
        input: .space  256
        output:.space 256
 .text

main:
        # User Enter The Input
        li      $v0, 8
        la      $a0, input
        li      $a1, 256
        syscall

        li      $v0, 4
        la      $a0, input
        syscall

        jal     strlen          # (Jump and link) to strlen function, saves return
        address to $ra


        add     $t1, $zero, $v0  # Copy some of our parameters for our reverse func
        add     $t2, $zero, $a0  # We need to save our input string to $t2, it gets
        add     $a0, $zero, $v0  # butchered by the syscall.
        li      $v0, 1           # This prints the length that we found in 'strlen'
        syscall

reverse:
        li      $t0, 0
        li      $t3, 0
        For:
                add     $t3, $t2, $t0    # $t2 is the base address for our 'input' array,
add loop index
                lb      $t4, 0($t3)      # load a byte at a time according to counter
                beqz    $t4, exit        # Null byte
                sb      $t4, output($t1) # Overwrite this byte address in memory

                subi    $t1, $t1, 1
                addi    $t0, $t0, 1
                j       For

exit:
      # Pint the output
        li      $v0, 4
```

```
	la	$a0, output
	syscall

	# End the program
	li	$v0, 10
	syscall


strlen:
	li	$t0, 0
	li	$t2, 0

	For2:
		add	$t2, $a0, $t0
		lb	$t1, 0($t2)
		beqz	$t1, strlen_exit
		addiu	$t0, $t0, 1
		j	For2

	strlen_exit:
		subi	$t0, $t0, 1
		add	$v0, $zero, $t0
		add	$t0, $zero, $zero
		jr	$ra
```

Problem 3:

```
#for(i = 0; i < a; i++){
#   for(j = i; j >= 0; j--){
#        C[i] *= j;
#       }
#}

# i = $t1 # a = 5 # j = $t2 # c[$t1] = $s0

.data
        c: .word 1, 2, 3, 4, 5
.text

main:
        li $t0, 5 # a = 5
        li $t1, 0 # i = 0
        la $s0, c # $s0 = &c[0]
        move $t3, $s0 # loads &c[0]

For1:

        bge  $t0, $t1, exit # i < a
        move $t2, $0 # j = 0
        addi $t1,$t1,1 # i = i + 1
        lw   $s1, ($t3) # load c[i] value

For2:
        bge $0, $t2, body # j >= 0
        addi $t2,$t1,0
        mul $s1, $s1, $t2 # c[i] = c[i] * j
        subi $t2, $t2, 1 # j--
        j    For2

body:
        sw   $s1, ($t3) # store to c[i]
        addi $t3, $t3, 4 # move pointer to next word
        j    For1

exit:
        li $v0, 10
        syscall
```

Problem 4:

```
#int main() {
#    ...
#    t1 = fact(8);
#    t2 = fact(3);
#    t3 = t1 + t2;
#    ...
# }

#int fact(int n) {
#        int i, result = 1;
#        for (i = n; i > 1; i--)
#                result = result * i;
#        return result;
#  }


.data

.text

Main:
        #Fact(8)
        li $s1,8
        move $a0, $s1
        jal Fact
        move $t1,$v0 # t1 = Fact(8)

        # Fact(3)
        li $s2,3
        move $a1, $s2
        jal Fact
        move $t2,$v1 # t2 = Fact(3)
        add $t3,$t1,$t2  # t3 = t1 + t2




Fact:

       addiu $sp,$sp,-4
       sw $ra,0($sp)
       li $v0,1 # result = 1
       #li $v1,1 # var = 1
       add $t0,$a0,$zero # i = n
```

```
For:

    ble $t0,1,Else # i>1  (Using DeMorgan's Law)
    mul $v0,$v0,$t0 # result = result * i
    sub $t0,$t0,1 # OR (addi $t0,$t0,-1)  .. i--
    b For
    Else:
    lw $ra,0($sp) # pop ra
    addiu $sp,$sp,4
    jr $ra



EndtheProgram:
            li $v0,10
            syscall
```

Problem 5:

```
.data
.text
main:

   # Get the length of the word.
        addi $t0, $0, 0 # j=0
        FOR: add $t2, $a0, $t0
        lb $t2, 0($t2)
        beq $t2, $0, exit
        addi $t0, $t0, 1
        j FOR
        #------------------------------------#

        exit: addi $t0, $t0, -1 # j-- (Length of word - 1)
        addi $t1, $0, 0 # i=0

        # Check the word is palindrome or not
        FOR2: slt $t2, $t1, $t0
        beq $t2, $0, True
        add $t2, $a0, $t1
        lb $t2, 0($t2)

        add $t3, $a0, $t0
        lb $t3, 0($t3)
        bne $t2, $t3, False

        addi $t0, $t0, -1 # j--
        addi $t1, $t1, 1 # i++
        j FOR2
        #---------------------------------------#

        # The output
        True:
        addi $v0, $0, 1
        j True
        jr $ra

        False:
        addi $v0, $0, 0
        j False
        jr $ra
        #----------------------------------------#
```