**Q1-a)**

input 27 and 4,

|  |  |  |  |
| --- | --- | --- | --- |
| $t0 | $t1 | $t3 | Comments (#) |
| (?) | (?) | (?) | Na |
| (?) | 27 | (?) | the value of input 27 move in to $t1 |
| 0 | 27 | (?) | $t0 = 0 |
| 0 | 27 | 0 | $t3 = 0, if t1 >4 |
| 0 | 23 | 0 | $t1 = 27 - 4 |
| 1 | 23 | 0 | $t0++ |
| 1 | 23 | 0 | $t3 = 0, if t1 >4 |
| 1 | 19 | 0 | $t1 = 23 - 4 |
| 2 | 19 | 0 | $t0++ |
| 2 | 19 | 0 | $t3 = 0, if t1 >4 |
| 2 | 15 | 0 | $t1 = 19 - 4 |
| 3 | 15 | 0 | $t0++ |
| 3 | 15 | 0 | $t3 = 0, if t1 > 4 |
| 3 | 11 | 0 | $t1 = 15 - 4 |
| 4 | 11 | 0 | $t0++ |
| 4 | 11 | 0 | $t3 = 0, if t1 > 4 |
| 4 | 7 | 0 | $t1 = 11 - 4 |

**Q1-b)**

when put a negative int (-4), it works in infinite loop and nothing printed in console.

**Q1-c)**

DONE:

li $v0, 4 # syscall to print a string

la $a0, msg3

syscall

li $v0, 1 # syscall to print an integer

add $a0, $t1, $zero

syscall

li $v0, 10 # syscall code to exit

syscall

**Q2-a)**

.text

la $t0, A\_LENGTH

lw $t0, 0($t0) #t0 <- A\_LENGTH

la $t1, A #t1: to hold the "address" of the next

lw $s0, $t1. # s0 = 0 Max set first value(element) of the NEXT\_ARRAY\_ELEMENT:

#array element, initialised to the

#address of the first byte of the array

addi $s0, $zero, 0 #s0: will hold the total sum,

# initialised to zero

NEXT\_ARRAY\_ELEMENT:

slt $t3, $zero, $t0 # t3 <-(0<t0), t3 will be 0 if t0 <=0

beq $t3, $zero, DONE

lw $t2, 0($t1) #t2 <- the current array element

slt $t4, $t2, $w0 #t4=0 if s2 <=t2

beq $t4, 1, MOVE # go to MOVE

j NEXT\_ARRAY\_ELEMENT #jump to NEXT\_ARRAY\_ELEMENT (for loop)

MOVE:

add $s0, $zero, $t2. # set s0 large array value

DONE:

addi $v0, $zero, 1 #set v0 to "1" to select

#"print integer" syscall

add $a0, $zero, $s0 #a0 <-s0 (the total sum) to be printed

syscall #invoking the syscall to actually exit!

addi $v0, $zero, 10 #set v0 to "10" to select exit syscall

syscall #invoking the syscall to acutally exit!

.data

A: #our integer array

.word -1

.word 4

.word -16

.word 0

.word -2

.word 5

.word 13

.word 2

A\_LENGTH: .word 8 # the length of the array

**Q2-b)**

.text

la $t0, A\_LENGTH

lw $t0, 0($t0) #t0 <- A\_LENGTH

la $t1, A #t1: to hold the "address" of the next

lw $s0, $t1. # s0 = 0 Max set first value(element) of the NEXT\_ARRAY\_ELEMENT:

#array element, initialised to the

#address of the first byte of the array

NEXT\_ARRAY\_ELEMENT:

slt $t3, $zero, $t0 # t3 <-(0<t0), t3 will be 0 if t0 <=0

beq $t3, $zero, DONE

lw $t2, 0($t1) # t2 <- the current array element

andi $t4, $t2, 0

beq $t4, $zero, CHANGE # go to Change

j NEXT\_ARRAY\_ELEMENT #jump to NEXT\_ARRAY\_ELEMENT (for loop)

CHANGE:

addi $v0, $zero, 1 # v0=1

add $a0, $zero, $t4 # print total sum

syscall

DONE:

addi $v0, $zero, 1 #set v0 to "1" to select

#"print integer" syscall

add $a0, $zero, $s0 #a0 <-s0 (the total sum) to be printed

syscall #invoking the syscall to actually exit!

addi $v0, $zero, 10 #set v0 to "10" to select exit syscall

syscall #invoking the syscall to acutally exit!

.data

A: #our integer array

.word -1

.word 4

.word -16

.word 0

.word -2

.word 5

.word 13

.word 2

A\_LENGTH: .word 8 # the length of the array

**Q2-c-i)**

andi $t0, $t1, 0x0007 #t0 = t1 & 0x0007

It can be divided into 8 because bit units are calculated to assemble bit patterns. At runtime, the 16-bit immedate operand extends to a 32-bit length by attaching a zero to the left. Store the result in register (t0) after performs AND assembly with the source register (t1) and operand constant (0x0007).

**Q2-c-ii)**

. text

la $t0, A\_LENGTH

lw $t0, 0($t0) #t0 <- A\_LENGTH

la $t1, A #t1: to hold the "address" of the next

NEXT\_ARRAY\_ELEMENT:

slt $t3, $zero, $t0 # t3 <-(0<t0), t3 will be 0 if t0 <=0

beq $t3, $zero, DONE

lw $t2, 0($t1) # t2 <- the current array element

andi $t4, $t2, 0

beq $t4, $zero, CHANGE # go to Change

j NEXT\_ARRAY\_ELEMENT #jump to NEXT\_ARRAY\_ELEMENT (for loop)

CHANGE:

addi $v0, $zero, 1 # v0=1

add $a0, $zero, $t4 # print total sum

syscall

DONE:

addi $v0, $zero, 1 #set v0 to "1" to select

#"print integer" syscall

add $a0, $zero, $s0 #a0 <-s0 (the total sum) to be printed

syscall #invoking the syscall to actually exit!

addi $v0, $zero, 10 #set v0 to "10" to select exit syscall

syscall #invoking the syscall to acutally exit!

.data

A: #our integer array

.word -1

.word 4

.word -16

.word 0

.word -2

.word 5

.word 13

.word 2

A\_LENGTH: .word 8 # the length of the array

**Q3-a)**

Move the bit in the register 'rt' to the left as much as 'h', then save in the register 'rd'. The range of the h bit length is 0<=h<32. If the bit is an unsigned integer, the left-hand shift is equal to multiplying by two (e.g. ,3 -> 2^3). So, sll $rd, $rt, h is rd=rt\*2h.

**Q3-b)**

sll $t0, $t1, 2

Name; 6 bits; op; sll

Format; 5bits; rs; R

Layout; 5 bits; rt; 0

Example; 5 bits; rd; 0

5 bits; shamt; 1

6 bits; funct; 0

10

0

->(therefore, format is) 0000 00ss ssst tttt dddd dhhh hh00 0000

$t0=8, $t1=9, ss sss= 00 000, t tttt= 0 1001, dddd d= 0100 0, hhh hh= 000 10

**Q3-c-iii)**

sll $t1, $t1, 3

**Q3-c-iv)**

sll $t0, $t1, 4. # multiply $t1 by 16 (2^4) and save in $t0

add $t1 $t0, $t1

sub $t0, $t1, $t0

**Q3-c-v)**

sll $t0, $t1, 5 # multiply $t1 by 32 (2^5) and save in $t0

sub $t1, $t0, $t1

**Q3-c-vi)**

sll $t0, $t1, 6. # multiply $t1 by 64 (2^6) and save in $t0

add $t1, $t0, $t1