- REPORT -

Name: Đào Minh Châu

ID:24560025

1. Classwork: Answer all question in the instruction.

Question 2.1.

Offset = 4*i because the difference between each cell's address is 4 bytes, so we have to multiply the index by 4 to represent a word(4 bytes) in order to work with the next memory cell.

Question 2.2.

Register $t2 is the offset which is used to calculate the address of the cell that we are working with.

Question 2.3.

```
.data
A: .word 2, 3, 5, 6, 1, 2, 3, 4
B: .space 32
.text
la $a0, A
la $a1, B
addi $a2, $zero, 8
add $t0, $t0, $zero #$t0 = i = 0
LOOP:  bge $t0, $a2, ENDLP #if i >= numVals, jump to ENDLP
  sll $t2, $t0, 2 #$t2 = 4 * i
  add $t3, $a0, $t2 #t3 = A + 4i, ie. &A[i]
  add $t4, $a1, $t2 #t4 = B + 4i, ie. &B[i]
```

```
  lw $t5, 0($t3) #load A[i] into $t5
  sw $t5, 0($t4) #store $t5 to B[i]
  addi $t0, $t0, 1 #increment i
j LOOP
ENDLP:
```

Question 3.1.

-Pointers: +User works with memory cells' addresses.

+The loop only stops when the address of the current cell is equal or greater than the last cell's address.

+Need to declare pointers.

-Index: +User works with array index.

+The loop only stops when the index of the current cell is equal or greater than numvals.

+Need to declare a counter.

Question 3.2.

```
.data
A: .word 2, 3, 5, 6, 1, 2, 3, 4
B: .space 32
.text
#a0 = A, $a1 = B, $a2 = numVals
la $a0, A
la $a1, B
addi $a2, $a2, 8
add $t0, $a0, $zero #$t0 = ptrA = A  (or &A[0])
add $t1, $a1, $zero #$t1 = ptrB = B  (or &B[0])
sll $t2, $a2, 2 #$t2 = 4 * numVals
```

```
add $t2, $a0, $t2 #$t2 = A + 4 * numVals
LOOP:  bge $t0, $t2, ENDLP #if ptrA >= A + 4*numVals, jump to ENDLP
  lw  $t4, 0($t0) #read *ptrA into $t4
  sw  $t4, 0($t1) #store $t4 out to *ptrB
  addi $t0, $t0, 4 #increment ptrA  (by 4 because ptr to int)
  addi $t1, $t1, 4 #increment ptrB  (by 4 because ptr to int)
  j LOOP #jump to top of loop
ENDLP:
```

2. Exercise

2.1. Write a program which allow user to input an array (n elements), print to I/O window of MARS as each below requirements:

- Print the maximum and minimum value of the array
- Calculate and print the sum of all elements
- Let user to input an index and then print the value of that element

```
.data
input1: .asciiz "Number of elements in your array: "
input2: .asciiz "Input an element: "
max: .asciiz "Max:"
min: .asciiz "Min: "
sum: .asciiz "The total sum is "
null: .space 3
arr: .space 400
newline: .asciiz "\n"
choose: .asciiz "Pick your index: "
output: .asciiz "Your number:  "
```

```
.text
la $s1, arr #arr base is at $s1
li $v0, 4
la $a0, input1
syscall
li $v0, 5
syscall
add $t0, $zero, $v0 #$t0 =numVals
addi $t1, $zero, 0 #$t1 =counter=i=0
Input_arr_LOOP:
bge $t1, $t0, Max
li $v0, 4
la $a0, input2
syscall
li $v0,5
syscall
add $s2, $zero, $v0 #input value is stored to A[i]=$s2
sll $t2, $t1, 2 #$t2 =4*i
add $t3, $s1, $t2 #$t3 is the place that holds the address A[i]
sw $s2, 0($t3)
addi $t1, $t1,1
j Input_arr_LOOP

Max:
addi $t1, $zero,0 #reset counter
addi $s3, $zero,0 #$s3 is the MAX VALUE
```

```
CHECK_MAX_LOOP:
bge $t1, $t0, COUT_MAX
sll $t2, $t1, 2 #$t2 -4*i
add $t3, $s1, $t2 #$t3 is the place that holds the address A[i]
lw $t4, 0($t3) # the value of A[i] at $t4
bgt $s3, $t4, mid
add $s3, $zero, $t4
mid:
addi $t1,$t1,1
j CHECK_MAX_LOOP

COUT_MAX:
li $v0,4
la $a0, max
syscall
li $v0,1
add $a0, $zero, $s3
syscall
li $v0,4
la $a0, newline
syscall

MIN:
addi $t1, $zero, 0 #reset counter
addi $s3, $zero, 9 #$s3 is the MIN VALUE
CHECK_MIN_LOOP:
```

```
bge $t1, $t0, COUT_MIN
sll $t2, $t1,2 #$t2 =4*i
add $t3, $s1, $t2 #$t3 is the place that holds the address A[i]
lw $t4, 0($t3) # the value of A[i] at $t4
blt $s3, $t4, mid2
add $s3,$zero,$t4
mid2:
addi $t1,$t1,1
j CHECK_MIN_LOOP

COUT_MIN:
li $v0,4
la $a0, min
syscall
li $v0, 1
add $a0, $zero, $s3
syscall
li $v0, 4
la $a0, newline
syscall

SUM:
addi $t5, $zero, 0 #$t5=SUM
addi $t1, $zero, 0 #reset counter
SUM_LOOP:
bge $t1, $t0, COUT_SUM
```

```
sll $t2, $t1, 2 #$t2 =4*i
add $t3, $s1, $t2 #$t3 is the place that holds the address A[i]
lw $t4, 0($t3) # the value of A[i] at $t4
add $t5, $t5, $t4
addi $t1, $t1,1
j SUM_LOOP

COUT_SUM:
li $v0,4
la $a0, sum
syscall
li $v0,1
add $a0, $zero, $t5
syscall
li $v0,4
la $a0, newline
syscall

PICK_NUM:
addi $t5, $zero, 0 #$t5 is now the ouput num
addi $t1, $zero, 0 #reset counter
li $v0, 4
la $a0, choose
syscall
li $v0, 5
syscall
```

```
add $s4, $v0, $zero
li $v0, 4
la $a0, newline
syscall
FINAL_LOOP:
bgt $t1, $s4, COUT_INDEX
sll $t2, $t1, 2 #$t2 =4*i
add $t3, $s1, $t2 #$t3 is the place that holds the address A[i]
lw $t4, 0($t3) # the value of A[i] at $t4
addi $t5, $t4, 0 #$t5 is now the ouput num
addi $t1, $t1, 1
j FINAL_LOOP
COUT_INDEX:
li $v0, 4
la $a0, output
syscall
li $v0, 1
add $a0, $zero, $t5
syscall
```

2.2. Given three arrays which are declared in ".data" as below

```
.data
array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
size1: .word 10
```

```
array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16
size2: .word 16

array3: .space 8
size3: .word 8
```

- Print to I/O windows of MARS all elements of `array1` and `array2`.
- Assign values for elements of `array3` as:

```
array3[i] = array1[i] + array2[size2 - 1 - i]
```

```
.data
array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
size1: .word 10
array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
size2: .word 16
array3: .space 8
size3: .word 8
a1: .asciiz "ARRAY 1: "
a2: .asciiz "ARRAY 2: "
a3: .asciiz "ARRAY 3: "
newline: .asciiz "\n"
space: .asciiz " "
.text
la $s1,array1 #base address of array1 =$s1
la $s2,array2 #base address of array2 =$s2
la $s5,array3 #base address of array3 =$s5
```

```
la $t4, size1 #$t4 temporary holds the value
lw $s3, 0($t4) # $s3=size1
la $t4, size2  #$t4 temporary holds the value
lw $s4, 0($t4) #$s4=size2
la $t4,size3 #$t4 temporary holds the value
lw $t9, 0($t4 )#$t9=size3

Print_arr1:
addi $t1, $zero, 0 #$t1=counter
li $v0, 4
la $a0, a1
syscall
LOOP1:
bge $t1, $s3, Print_arr2
sll $t2, $t1, 2 # $t2=4*i
add $t3, $s1, $t2
lw $t4, 0($t3) #$t4 temporary holds the value of the arr1
li $v0, 1
add $a0, $zero, $t4
syscall
li $v0, 4
la $a0, space
syscall
addi $t1, $t1, 1
j LOOP1
```

```
Print_arr2:
li $v0,4
la $a0, newline
syscall
li $v0,4
la $a0, a2
syscall
addi $t1, $zero, 0 #reset counter
LOOP2:
bge $t1, $s4, ASSIGN_arr3
add $t3, $s2, $t1
lb $t4, 0($t3) #$t4 temporary holds the value of the arr2
li $v0, 1
add $a0, $zero, $t4
syscall
li $v0, 4
la $a0, space
syscall
addi $t1, $t1, 1
j LOOP2

ASSIGN_arr3:
li $v0, 4
la $a0, newline
syscall
addi $t1, $zero, 0 #reset counter
```

```
LOOP3:
bge $t1, $t9, Print_arr3
sll $t2, $t1, 2 # $t2=4*i
add $t3, $s1, $t2 # the address of array1[i] =$t3 temporary
lw $t5, 0($t3) #array1[i]=$t5
add $s6, $s5, $t2 # the address of array3[i] =$s6 temporary
addi $s7, $s4, -1#$s7=size2-1
sub $s7, $s7, $t1 #$s7=$s7-i
add $t6, $s2, $s7
lb $t7, 0($t6) #array2[size2-1-i]=$t7
add $t8, $t5, $t7
sw $t8, 0($s6)
addi $t1, $t1, 1
j LOOP3

Print_arr3:
li $v0, 4
la $a0, newline
syscall
li $v0, 4
la $a0, a3
syscall
addi $t1, $zero, 0#reset counter
LOOP4:
bge $t1, $t9, END
sll $t2, $t1, 2
```

```
add $t3, $s5, $t2
lw $t4, 0($t3) #$t4 temporary holds the value of the arr3
li $v0, 1
add $a0, $zero, $t4
syscall
li $v0, 4
la $a0, space
syscall
addi $t1, $t1, 1
j LOOP4
END:
```