FA23-BCE-113 Mohammad Ahmad

Question 1:

```
# initialization:
addi $t0, $zero, s     # s(32-bit) init. in $t0

addi $t1, $zero, 20    # $t1= 20 (i)

addi $t2, $zero, -1    # $t2 = -1

Loop:
   beq $t1, $t2, Exit        # if (i == -1), exit loop

 # Load X[i-1] from RAM
   addi $t3, $t1, -1         # t3 = i-1
   SLL  $t4, $t3, 2          # t4 = t3 X4
   add  $t4, $S1, $t4  #add base address X[]
   lw   $t5, 0($t4)  # load X[i-1] → $t5
```

R.W

$\boxed{SLL}$

let

$$\left.\begin{array}{l} x = (0010)_2 = 2 \\ \text{SLL } x \text{ 2 times} \\ x = (1000)_2 = 8 \end{array}\right) \times 4$$

So SLL 2 time is equal to X 4

```
 # Load Y[i]
    SLL $t6, $t1, 2          # t6 = i X 4
    add $t6, $t6, $S2   # add base address of Y[]
    lw  $t7, 0($t6)   # Load Y[i] → $t7

 # s x Y[i]
    mul $t8, $t0, $t7          # $t8 = s * [Yi]
 # X[i-1] + s * Y[i]
    add $t9, $t5, $t8      # $t9= X[i-1] + S*Y[i]

# Store $t9 at W[i+1]
   addi $t3, $t1, 1        # t3 = i+1
   SLL  $t3, $t3, 2          # t3 = t3 X4
```

we calculate address (base + offset) eaarlier and then while loading we give 0 address as we already done that.

in next question I am doing the other method in which offset address give at load command.

```
add $t3 ,$s0, $t3    # add base address of w[]
 sw  $t9,0($t3)      #

 # i=i-1
 addi $t1, $t1,-1
 j Loop                  # jump back to loop procedure

# Exit
 Exit:   # Exit from code.
```

Q3:

```
find_max:
        lw $t0, 0($s0)  # $s0 is base_address of x[ ] also
                        # the 1st index, let it be x[0]
                        # t0 = x[0], assuming $t0 as max
        li $t1, 1   #   $t1 = i = 1.
      loop:
          bge  $t1, $s1, Exit   # $s1 = size , if (i >= size) goto→ exit
          SLL  $t2, $t1, 2      # offset = i*4
          LW   $t4, $t2($s0)    # $t4 = x[i] ← loaded from mem.

          bLE  $t4 ,$t0, else   # if x[i] <= max goto→ else
          move  $t0 , $t4        # max = x[i]
      Else:
          addi $t1, $t1, 1      # $t1 = (i = i+1) OR i++
          j  loop
                                # jump back to loop
```

```
Exit:
    move $v0, $t0      # moving $t0(max) to return value
                       register
    jr  $ra
                       # jump register to return address
                       # which redirect function to main
                                                   context
main:
  # initialization of variables
    jal  find-max      # calling function from main
```

Question 2:

~~Q~~^{2} initial value given in Question followed in below code:

```
everse array:
    addi  $sp,$sp,-4          # stack allocate
    sw    $a0, 0($sp)         # store array X[] address in stack

    li    $s0 , 0             # $s0 = left =0
    addi  $s1, $a1, -1        # $s1= right= size-1 , $a1 = size
loop:
    bge   $s0, $s1, exit      # if(left >= right jump→ exit
    sll   $t2, $s0, 2         # $t2 ··· = $0 x4  (offset). X-left
    add   $t2, $t2, $a0       # add base address X[]
    lw    $t0, 0($t2)         # $t0 = temp = X[left]
    sll   $t3, $s1, 2         # $t3 = $s1x4   offset for X-right
    add   $t3, $t3, $a0       # add base address of X[]
    lw    $t4, 0($t3)         # $t4 = X[right]
```

~~dar~~

```
# Now: $t0 = X[left)                              swap logic
       $t4 = X[right)  & swap to &t3    |  $t7 = $t0
                                        |  $t0 = $t4
                                        |  $t4 = $t7
```

```
    move  $t7, $t0          # swap logic
    move  $t0, $t4
    move  $t4 = $t7
 # now store  $t0 & $t3 in memory

    sw $t0, 0($t2)     # store updated value to
    sw $t4, 0($t3)     # Their registers.


    addi $s0, $s0, 1          #   left ++
    addi $s1, $s1, -1         #   right ++
    j    loop                 # loop repeat

Exit:
    lw  $a0, 0($sp)    # restore base address of
                         arr X[ ] from stack
    addi $sp, $sp, 4   # dellocate stack

    jr  $ra
```

Since The function is void we dont have to return but we can also return base address $a0 of X[ ] so that we can access the full array after reverse.