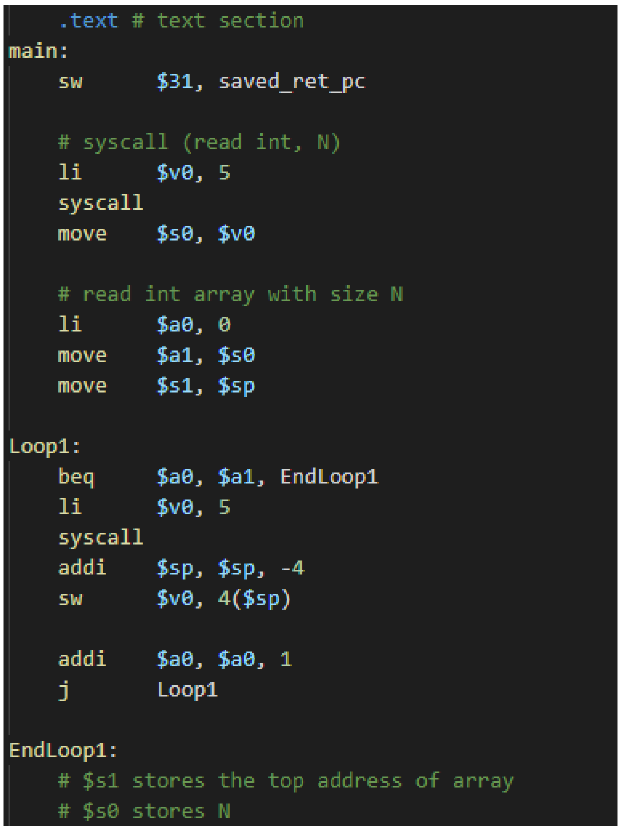
HW2 Report

20200639 채우진

1. Stack Allocation

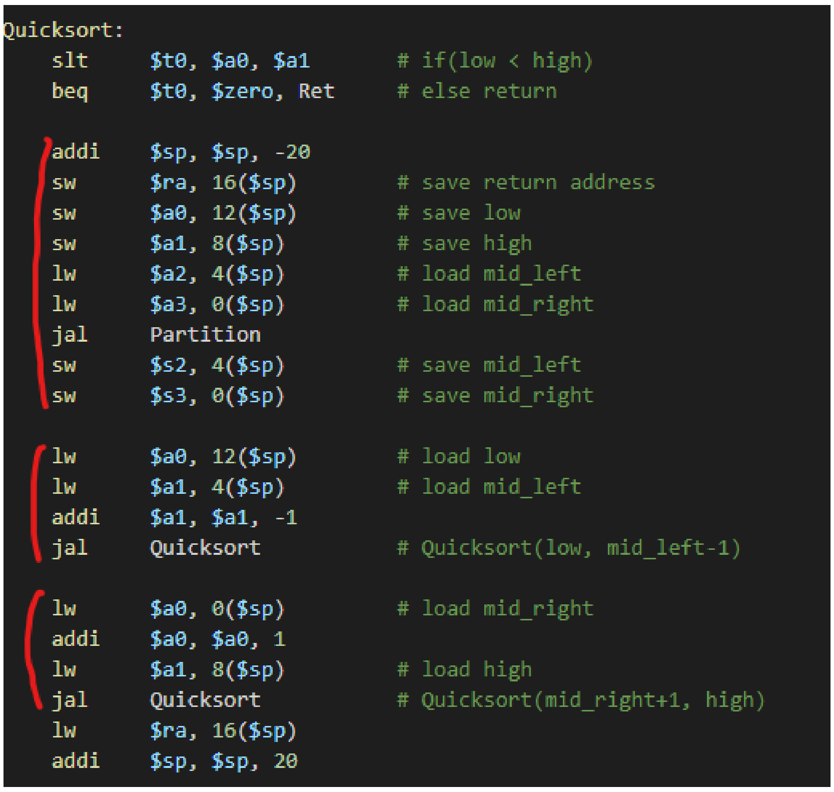
A. Getting series of integer size N:



```
    .text # text section
main:
    sw      $31, saved_ret_pc

    # syscall (read int, N)
    li      $v0, 5
    syscall
    move    $s0, $v0

    # read int array with size N
    li      $a0, 0
    move    $a1, $s0
    move    $s1, $sp

Loop1:
    beq     $a0, $a1, EndLoop1
    li      $v0, 5
    syscall
    addi    $sp, $sp, -4
    sw      $v0, 4($sp)

    addi    $a0, $a0, 1
    j       Loop1

EndLoop1:
    # $s1 stores the top address of array
    # $s0 stores N
```

Starting with the top address of the stack pointer, the received input was stored in each location while reducing the stack pointer. After I finished storing inputs, I stored the number of inputs, and the top address of array in $s0, $s1 respectively.

B. Quicksort Implementation:

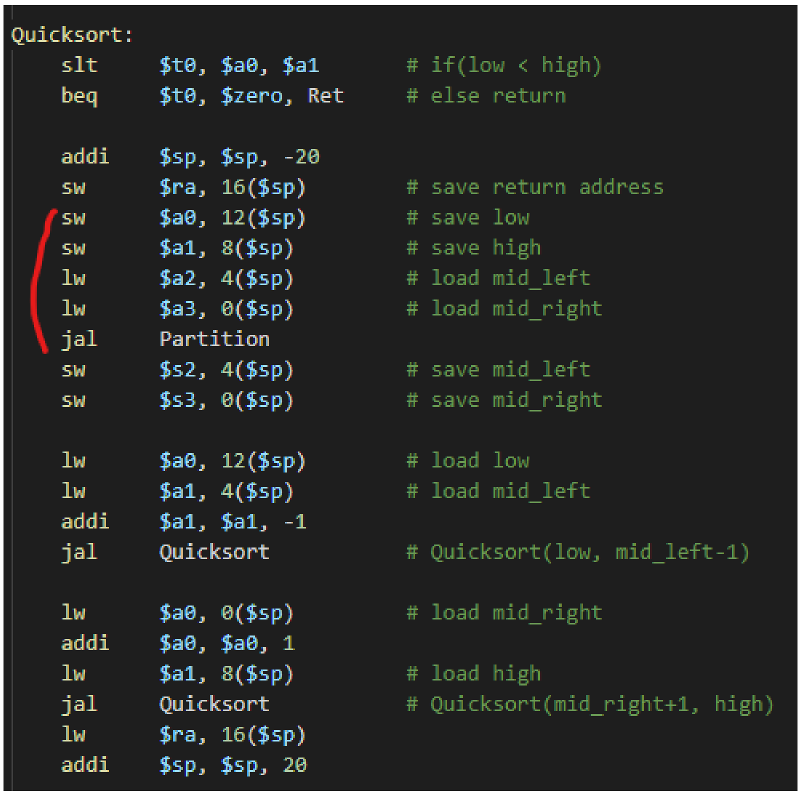


```
Quicksort:
    slt     $t0, $a0, $a1       # if(low < high)
    beq     $t0, $zero, Ret     # else return

    addi    $sp, $sp, -20
    sw      $ra, 16($sp)        # save return address
    sw      $a0, 12($sp)        # save low
    sw      $a1, 8($sp)         # save high
    lw      $a2, 4($sp)         # load mid_left
    lw      $a3, 0($sp)         # load mid_right
    jal     Partition
    sw      $s2, 4($sp)         # save mid_left
    sw      $s3, 0($sp)         # save mid_right

    lw      $a0, 12($sp)        # load low
    lw      $a1, 4($sp)         # load mid_left
    addi    $a1, $a1, -1
    jal     Quicksort           # Quicksort(low, mid_left-1)

    lw      $a0, 0($sp)         # load mid_right
    addi    $a0, $a0, 1
    lw      $a1, 8($sp)         # load high
    jal     Quicksort           # Quicksort(mid_right+1, high)
    lw      $ra, 16($sp)
    addi    $sp, $sp, 20
```

Everytime when I enter Quicksort, I decreased stack pointer by 20, to store 5 parameters; return address, low, high, mid_left, mid_right respectively. After each innated function(Partition, Quicksort, Quicksort) returned, I could reload modified value and use it as parameter for following functions.

2. Implementation:

The trick that I used for this programming assignment is “using interger array as a global variable” by storing it in $s1. It is quite efficient because I can save the number of arguments needed for calling functions. Specifically, the function ‘Partition' requires 5 parameters, although there are only 4 registers that I can use for arguments. If argument needed for function exceeds 4, then we should store extra arguments in the stack, but implementing those things are quite tough for me. So, I used the array as an global variable by storing it in the $s1.



```
Quicksort:
    slt     $t0, $a0, $a1       # if(low < high)
    beq     $t0, $zero, Ret     # else return

    addi    $sp, $sp, -20
    sw      $ra, 16($sp)        # save return address
    sw      $a0, 12($sp)        # save low
    sw      $a1, 8($sp)         # save high
    lw      $a2, 4($sp)         # load mid_left
    lw      $a3, 0($sp)         # load mid_right
    jal     Partition
    sw      $s2, 4($sp)         # save mid_left
    sw      $s3, 0($sp)         # save mid_right

    lw      $a0, 12($sp)        # load low
    lw      $a1, 4($sp)         # load mid_left
    addi    $a1, $a1, -1
    jal     Quicksort           # Quicksort(low, mid_left-1)

    lw      $a0, 0($sp)         # load mid_right
    addi    $a0, $a0, 1
    lw      $a1, 8($sp)         # load high
    jal     Quicksort           # Quicksort(mid_right+1, high)
    lw      $ra, 16($sp)
    addi    $sp, $sp, 20
```