# Computer Architecture

# Laboratory Lab 1

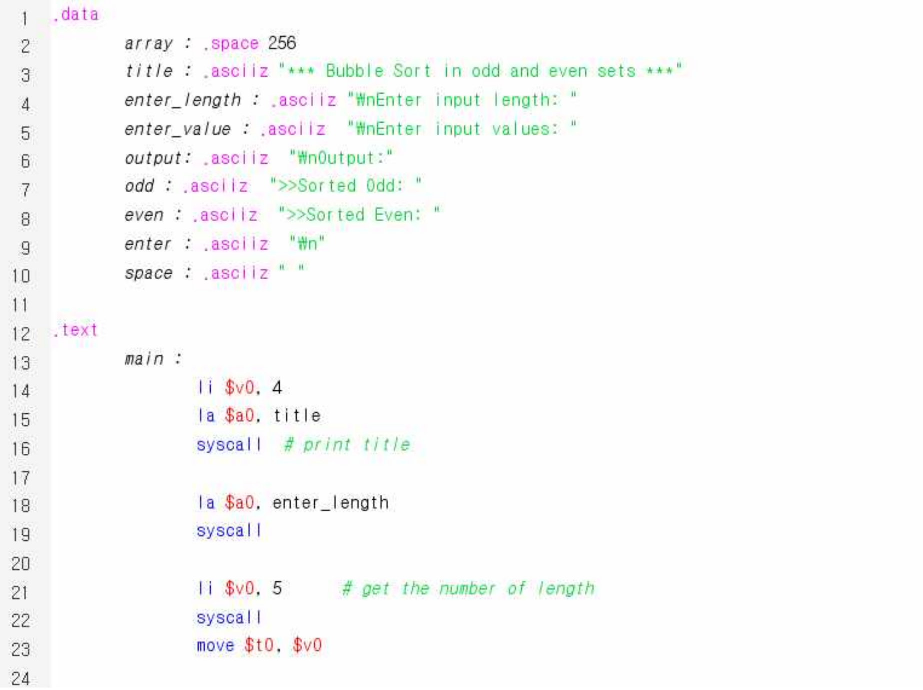
HOMEWORK: BUBBLE SORT DESCENDING ORDER

학번 : 2016707079  이름 : 하상천

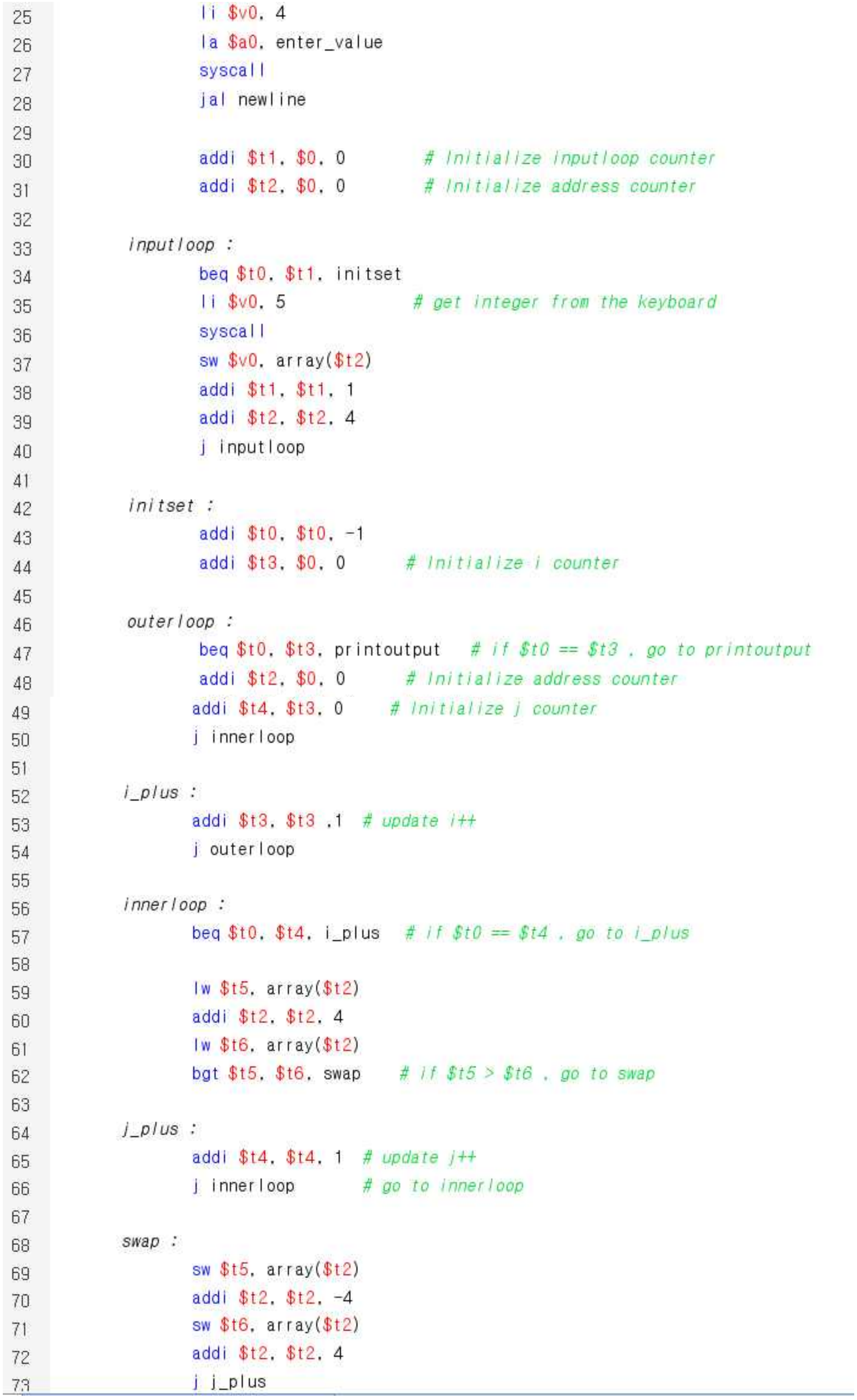
1. What changes did you make to do this?

입력할 숫자의 개수를 입력 받고, 그 수만큼 숫자를 입력받았다. Bubble sort 방법을 이용해서 작은 숫자부터 큰 숫자 순서로 정렬했다. 그리고 출력할 때 2로 나누어서 나머지가 1이면 출력을 다하고, 다시 address counter를 0으로 한 후 2로 나누어서 나머지가 0인 숫자들을 출력했다. 처음에는 space를 두 개 만들어서 짝수와 홀수를 먼저 나누어주고, bubble sort를 하려고 했는데 그러면 space 두 개를 만들기 때문에 메모리 낭비라고 생각해서 그렇게 하지 않았다.

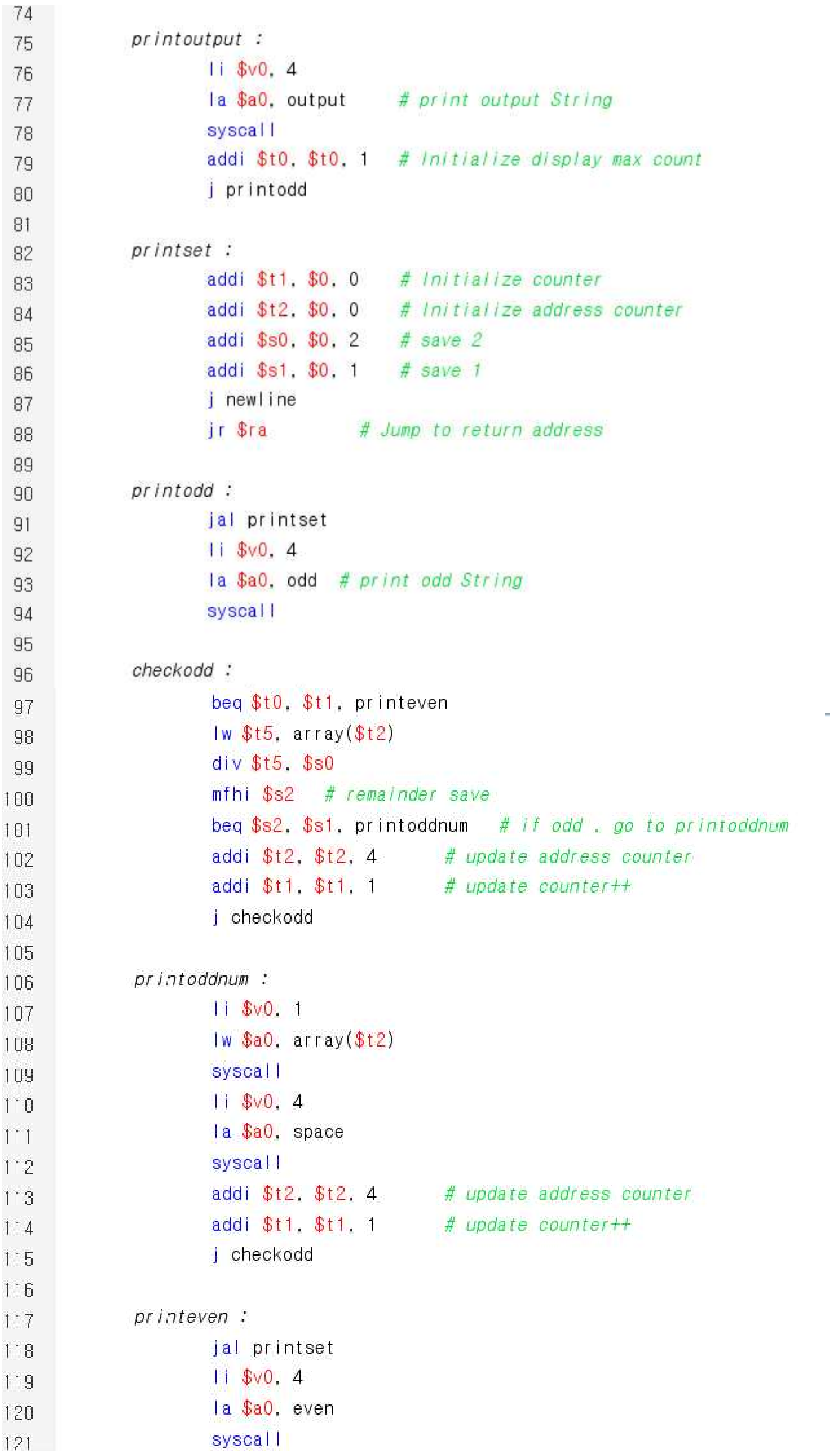
2. Show the screen capture of the modified part of the code.



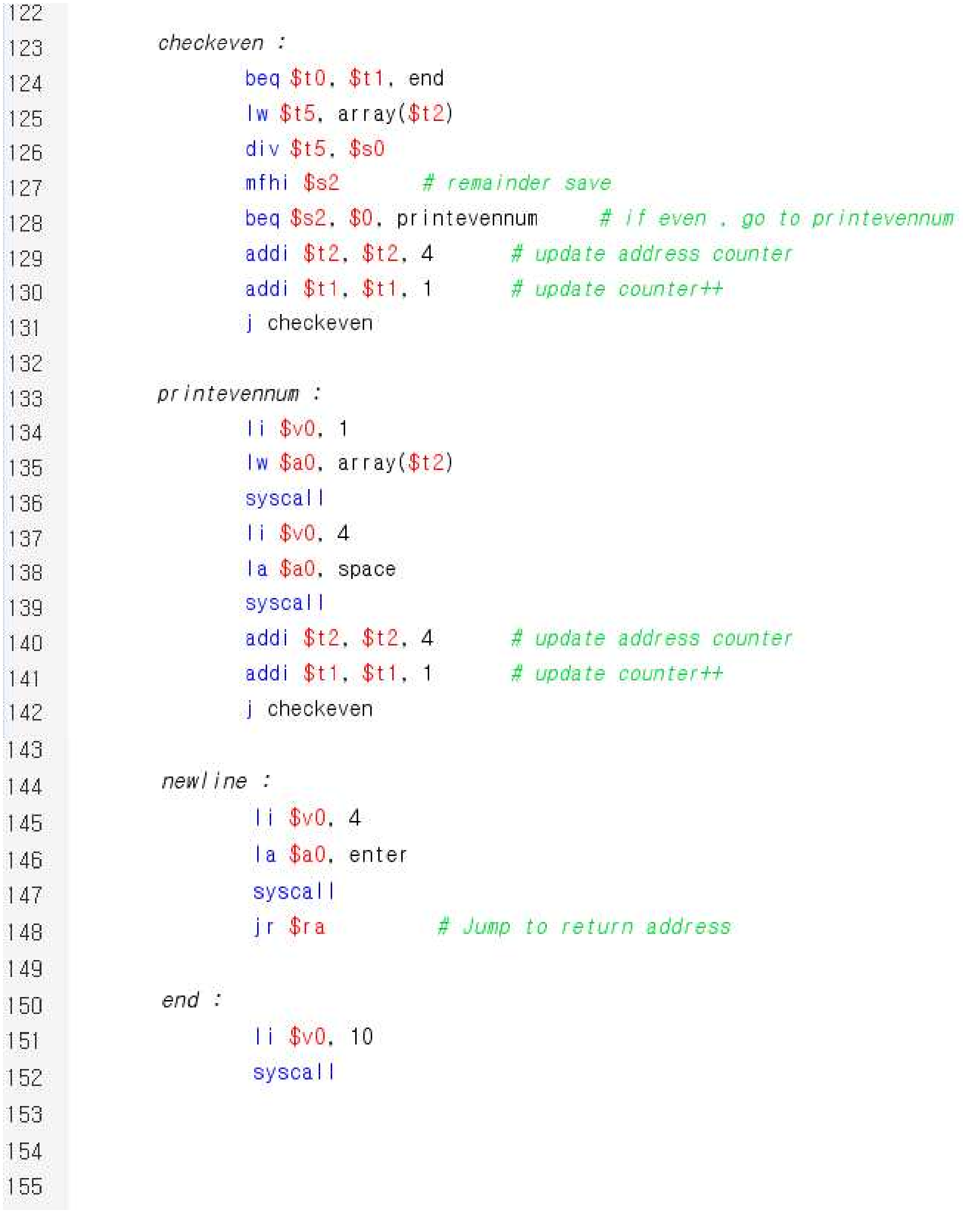
```
.data
        array : .space 256
        title : .asciiz "*** Bubble Sort in odd and even sets ***"
        enter_length : .asciiz "₩nEnter input length: "
        enter_value : .asciiz  "₩nEnter input values: "
        output: .asciiz  "₩nOutput:"
        odd : .asciiz  ">>Sorted Odd: "
        even : .asciiz  ">>Sorted Even: "
        enter : .asciiz  "₩n"
        space : .asciiz " "

.text
        main :
                li $v0, 4
                la $a0, title
                syscall  # print title

                la $a0, enter_length
                syscall

                li $v0, 5      # get the number of length
                syscall
                move $t0, $v0

```



```
            li $v0, 4
            la $a0, enter_value
            syscall
            jal newline

            addi $t1, $0, 0         # Initialize inputloop counter
            addi $t2, $0, 0         # Initialize address counter

    inputloop :
            beq $t0, $t1, initset
            li $v0, 5               # get integer from the keyboard
            syscall
            sw $v0, array($t2)
            addi $t1, $t1, 1
            addi $t2, $t2, 4
            j inputloop

    initset :
            addi $t0, $t0, -1
            addi $t3, $0, 0       # Initialize i counter

    outerloop :
            beq $t0, $t3, printoutput   # if $t0 == $t3 , go to printoutput
            addi $t2, $0, 0       # Initialize address counter
            addi $t4, $t3, 0     # Initialize j counter
            j innerloop

    i_plus :
            addi $t3, $t3 ,1   # update i++
            j outerloop

    innerloop :
            beq $t0, $t4, i_plus   # if $t0 == $t4 , go to i_plus

            lw $t5, array($t2)
            addi $t2, $t2, 4
            lw $t6, array($t2)
            bgt $t5, $t6, swap    # if $t5 > $t6 , go to swap

    j_plus :
            addi $t4, $t4, 1   # update j++
            j innerloop        # go to innerloop

    swap :
            sw $t5, array($t2)
            addi $t2, $t2, -4
            sw $t6, array($t2)
            addi $t2, $t2, 4
            j j_plus
```

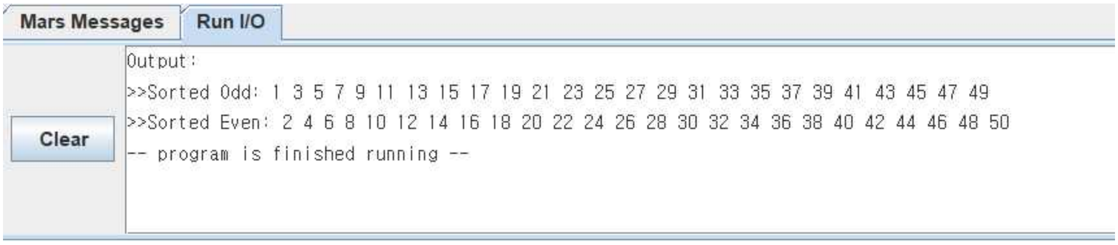


```
    printoutput :
        li $v0, 4
        la $a0, output      # print output String
        syscall
        addi $t0, $t0, 1    # Initialize display max count
        j printodd

    printset :
        addi $t1, $0, 0     # Initialize counter
        addi $t2, $0, 0     # Initialize address counter
        addi $s0, $0, 2     # save 2
        addi $s1, $0, 1     # save 1
        j newline
        jr $ra          # Jump to return address

    printodd :
        jal printset
        li $v0, 4
        la $a0, odd  # print odd String
        syscall

    checkodd :
        beq $t0, $t1, printeven
        lw $t5, array($t2)
        div $t5, $s0
        mfhi $s2   # remainder save
        beq $s2, $s1, printoddnum   # if odd , go to printoddnum
        addi $t2, $t2, 4       # update address counter
        addi $t1, $t1, 1       # update counter++
        j checkodd

    printoddnum :
        li $v0, 1
        lw $a0, array($t2)
        syscall
        li $v0, 4
        la $a0, space
        syscall
        addi $t2, $t2, 4       # update address counter
        addi $t1, $t1, 1       # update counter++
        j checkodd

    printeven :
        jal printset
        li $v0, 4
        la $a0, even
        syscall
```

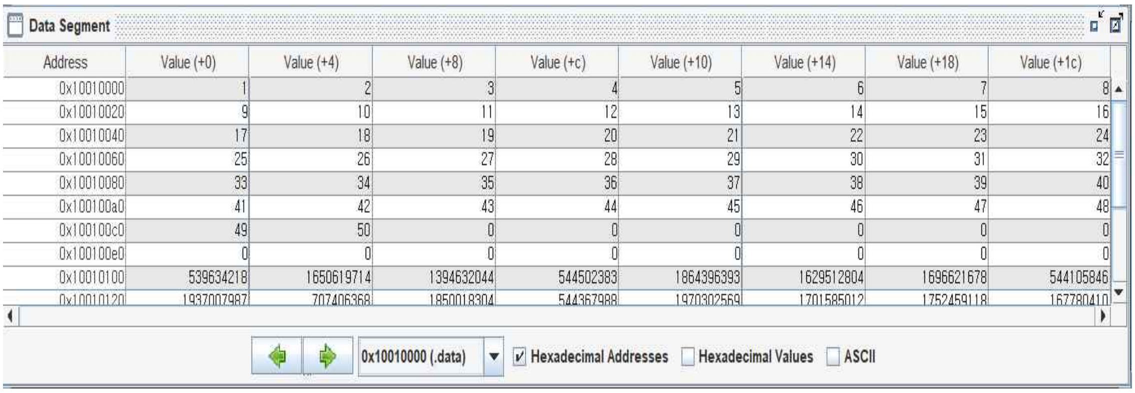


```
    checkeven :
        beq $t0, $t1, end
        lw $t5, array($t2)
        div $t5, $s0
        mfhi $s2        # remainder save
        beq $s2, $0, printevennum     # if even , go to printevennum
        addi $t2, $t2, 4      # update address counter
        addi $t1, $t1, 1      # update counter++
        j checkeven

    printevennum :
        li $v0, 1
        lw $a0, array($t2)
        syscall
        li $v0, 4
        la $a0, space
        syscall
        addi $t2, $t2, 4      # update address counter
        addi $t1, $t1, 1      # update counter++
        j checkeven

    newline :
         li $v0, 4
         la $a0, enter
         syscall
         jr $ra          # Jump to return address

    end :
         li $v0, 10
         syscall
```

3. Test the modified code with more than 30 elements then show the results. Also show the output of the data segment.

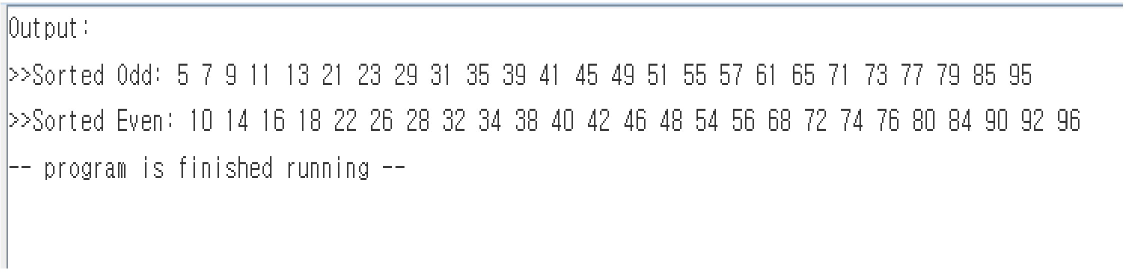


```
Output:
>>Sorted Odd: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
>>Sorted Even: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
-- program is finished running --
```

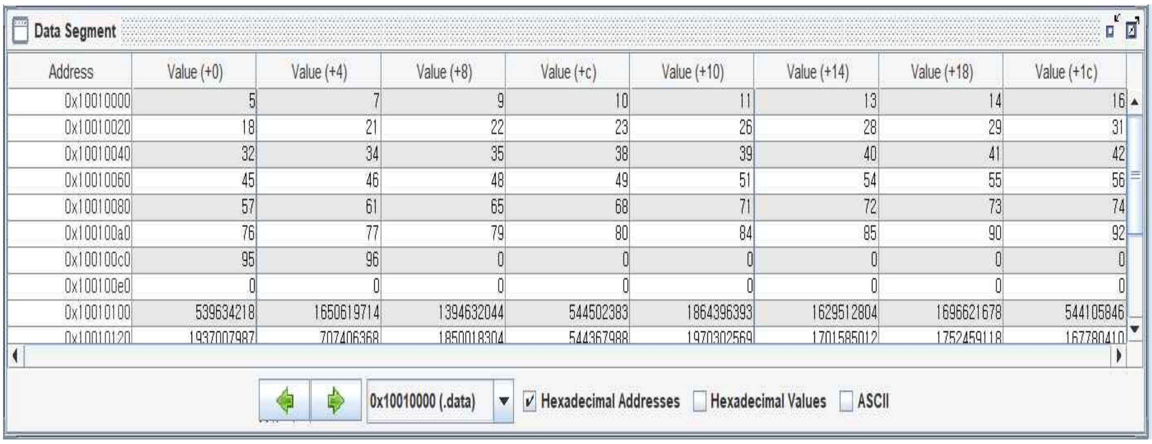


Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|---|---|---|---|---|---|---|---|---|
| 0x10010000 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0x10010020 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 0x10010040 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 0x10010060 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 0x10010080 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 0x100100a0 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 0x100100c0 | 49 | 50 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010100 | 539634218 | 1650619714 | 1394632044 | 544502383 | 1864396393 | 1629512804 | 1696621678 | 544105846 |
| 0x10010120 | 1937007987 | 707406368 | 1850018304 | 544367988 | 1970302569 | 1701585012 | 1752459118 | 167780410 |



```
Output:
>>Sorted Odd: 5 7 9 11 13 21 23 29 31 35 39 41 45 49 51 55 57 61 65 71 73 77 79 85 95
>>Sorted Even: 10 14 16 18 22 26 28 32 34 38 40 42 46 48 54 56 68 72 74 76 80 84 90 92 96
-- program is finished running --
```



Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|---|---|---|---|---|---|---|---|---|
| 0x10010000 | 5 | 7 | 9 | 10 | 11 | 13 | 14 | 16 |
| 0x10010020 | 18 | 21 | 22 | 23 | 26 | 28 | 29 | 31 |
| 0x10010040 | 32 | 34 | 35 | 38 | 39 | 40 | 41 | 42 |
| 0x10010060 | 45 | 46 | 48 | 49 | 51 | 54 | 55 | 56 |
| 0x10010080 | 57 | 61 | 65 | 68 | 71 | 72 | 73 | 74 |
| 0x100100a0 | 76 | 77 | 79 | 80 | 84 | 85 | 90 | 92 |
| 0x100100c0 | 95 | 96 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010100 | 539634218 | 1650619714 | 1394632044 | 544502383 | 1864396393 | 1629512804 | 1696621678 | 544105846 |
| 0x10010120 | 1937007987 | 707406368 | 1850018304 | 544367988 | 1970302569 | 1701585012 | 1752459118 | 167780410 |