BURCU İSKENDER

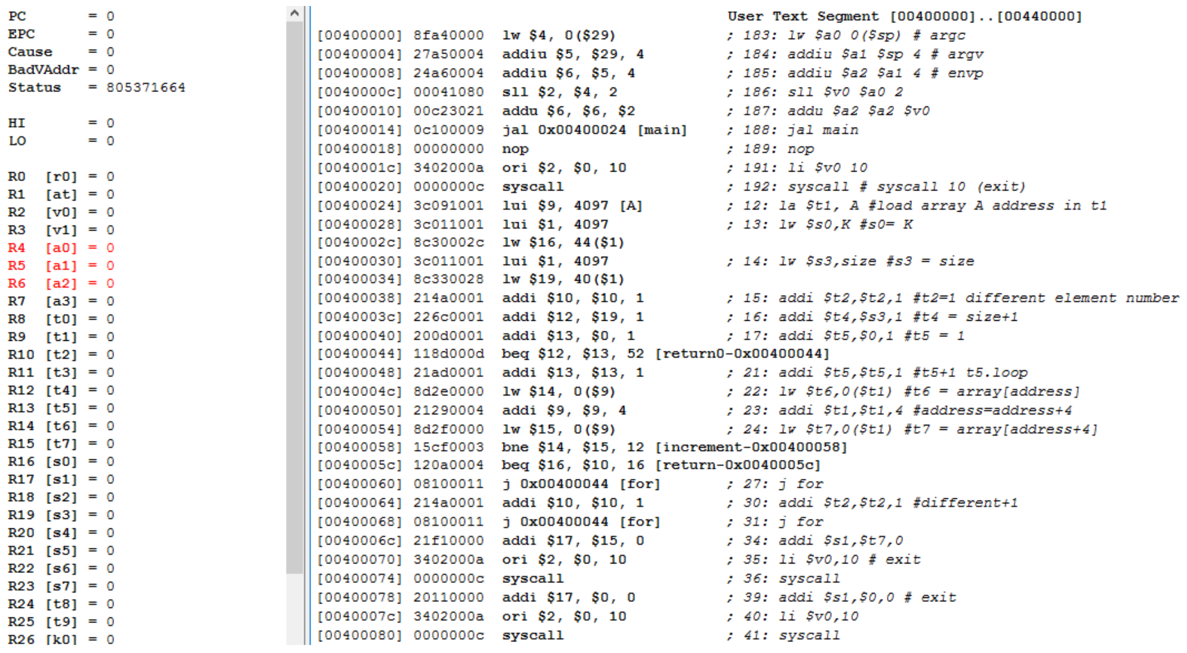
21328103

ADVANCED COMPUTER ARCHITECTURE – PROJECT 1

1)Order in sorted array

Store K in register s0, and the result in register s1.

Before Running :

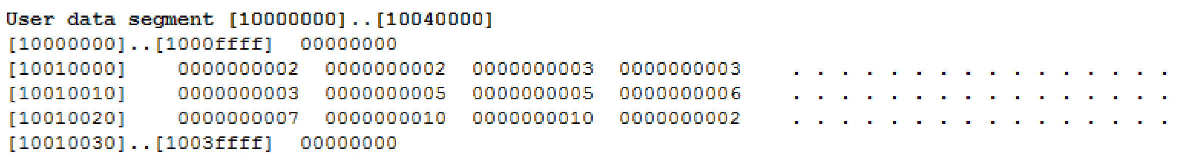


```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
```

```
                                            User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2        ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]  ; 188: jal main
[00400018] 00000000  nop                    ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10         ; 191: li $v0 10
[00400020] 0000000c  syscall                ; 192: syscall # syscall 10 (exit)
[00400024] 3c091001  lui $9, 4097 [A]       ; 12: la $t1, A #load array A address in t1
[00400028] 3c011001  lui $1, 4097           ; 13: lw $s0,K #s0= K
[0040002c] 8c30002c  lw $16, 44($1)
[00400030] 3c011001  lui $1, 4097           ; 14: lw $s3,size #s3 = size
[00400034] 8c330028  lw $19, 40($1)
[00400038] 214a0001  addi $10, $10, 1       ; 15: addi $t2,$t2,1 #t2=1 different element number
[0040003c] 226c0001  addi $12, $19, 1       ; 16: addi $t4,$s3,1 #t4 = size+1
[00400040] 200d0001  addi $13, $0, 1        ; 17: addi $t5,$0,1 #t5 = 1
[00400044] 118d000d  beq $12, $13, 52 [return0-0x00400044]
[00400048] 21ad0001  addi $13, $13, 1       ; 21: addi $t5,$t5,1 #t5+1 t5.loop
[0040004c] 8d2e0000  lw $14, 0($9)          ; 22: lw $t6,0($t1) #t6 = array[address]
[00400050] 21290004  addi $9, $9, 4         ; 23: addi $t1,$t1,4 #address=address+4
[00400054] 8d2f0000  lw $15, 0($9)          ; 24: lw $t7,0($t1) #t7 = array[address+4]
[00400058] 15cf0003  bne $14, $15, 12 [increment-0x00400058]
[0040005c] 120a0004  beq $16, $10, 16 [return-0x0040005c]
[00400060] 08100011  j 0x00400044 [for]     ; 27: j for
[00400064] 214a0001  addi $10, $10, 1       ; 30: addi $t2,$t2,1 #different+1
[00400068] 08100011  j 0x00400044 [for]     ; 31: j for
[0040006c] 21f10000  addi $17, $15, 0       ; 34: addi $s1,$t7,0
[00400070] 3402000a  ori $2, $0, 10         ; 35: li $v0,10 # exit
[00400074] 0000000c  syscall                ; 36: syscall
[00400078] 20110000  addi $17, $0, 0        ; 39: addi $s1,$0,0 # exit
[0040007c] 3402000a  ori $2, $0, 10         ; 40: li $v0,10
[00400080] 0000000c  syscall                ; 41: syscall
```

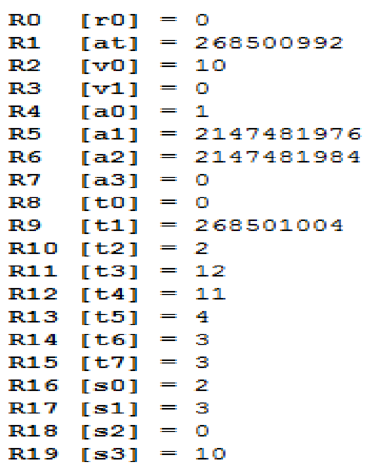
Test 1: A={2,2,3,3,3,5,5,6,7,10}, K=2

Before Running Memory :



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000002  0000000002  0000000003  0000000003    . . . . . . . . . . . . . . . .
[10010010]    0000000003  0000000005  0000000005  0000000006    . . . . . . . . . . . . . . . .
[10010020]    0000000007  0000000010  0000000010  0000000002    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

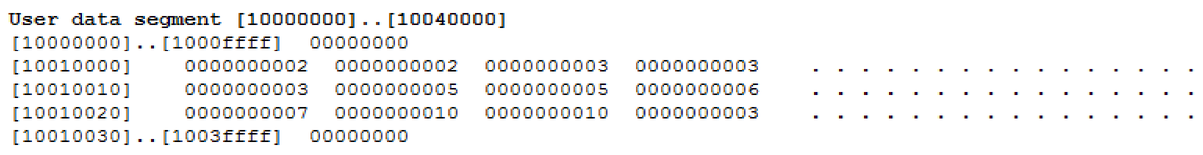
After Running Registers:



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 268501004
R10 [t2] = 2
R11 [t3] = 12
R12 [t4] = 11
R13 [t5] = 4
R14 [t6] = 3
R15 [t7] = 3
R16 [s0] = 2
R17 [s1] = 3
R18 [s2] = 0
R19 [s3] = 10
```

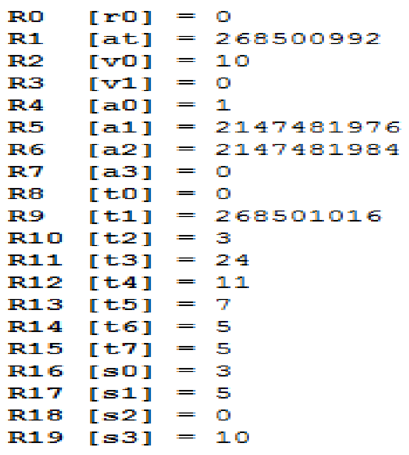
Test 2: A={2,2,3,3,3,5,5,6,7,10}, K=3

Before running: MEMORY



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000002  0000000002  0000000003  0000000003    . . . . . . . . . . . . . . . .
[10010010]    0000000003  0000000005  0000000005  0000000006    . . . . . . . . . . . . . . . .
[10010020]    0000000007  0000000010  0000000010  0000000003    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

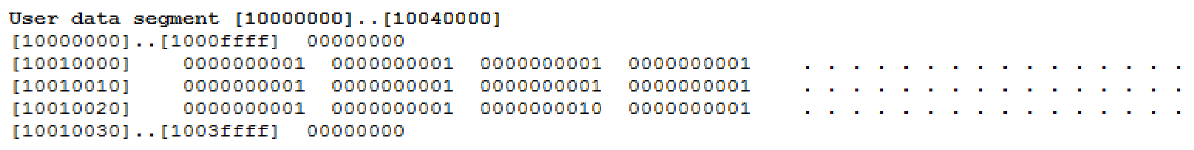
After running : REGISTERS



```
R0   [r0] = 0
R1   [at] = 268500992
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 1
R5   [a1] = 2147481976
R6   [a2] = 2147481984
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 268501016
R10  [t2] = 3
R11  [t3] = 24
R12  [t4] = 11
R13  [t5] = 7
R14  [t6] = 5
R15  [t7] = 5
R16  [s0] = 3
R17  [s1] = 5
R18  [s2] = 0
R19  [s3] = 10
```

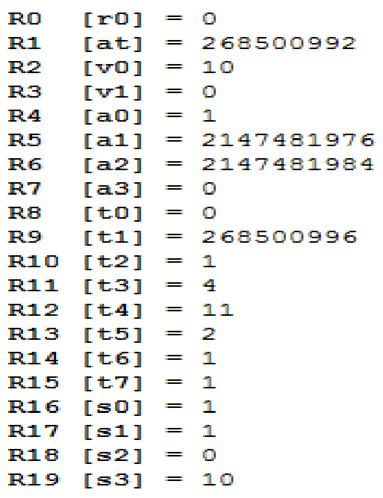
Test 3:A={1,1,1,1,1,1,1,1,1,1}, K=1

Before running: MEMORY



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010010]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010020]    0000000001  0000000001  0000000010  0000000001    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

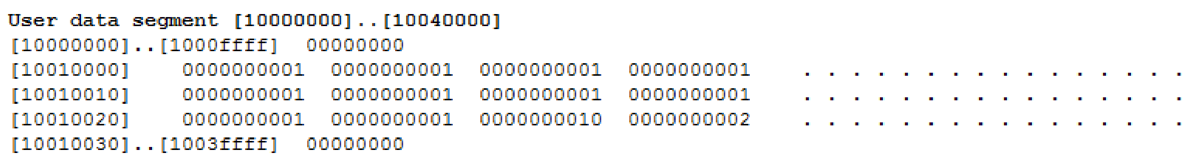
After running:REGISTERS



```
R0   [r0] = 0
R1   [at] = 268500992
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 1
R5   [a1] = 2147481976
R6   [a2] = 2147481984
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 268500996
R10  [t2] = 1
R11  [t3] = 4
R12  [t4] = 11
R13  [t5] = 2
R14  [t6] = 1
R15  [t7] = 1
R16  [s0] = 1
R17  [s1] = 1
R18  [s2] = 0
R19  [s3] = 10
```

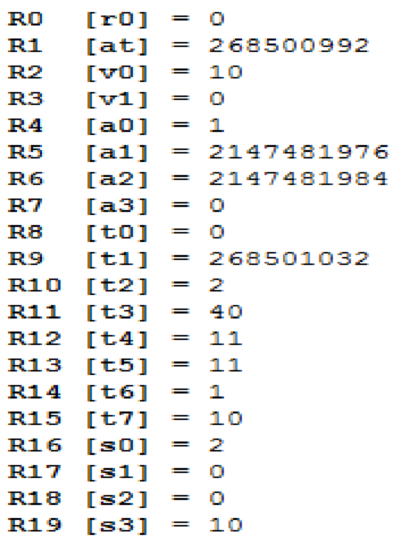
Test 4:A={1,1,1,1,1,1,1,1,1,1}, K=2

Before running:MEMORY



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010010]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010020]    0000000001  0000000001  0000000010  0000000002    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

After running: REGISTERS



```
R0   [r0] = 0
R1   [at] = 268500992
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 1
R5   [a1] = 2147481976
R6   [a2] = 2147481984
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 268501032
R10  [t2] = 2
R11  [t3] = 40
R12  [t4] = 11
R13  [t5] = 11
R14  [t6] = 1
R15  [t7] = 10
R16  [s0] = 2
R17  [s1] = 0
R18  [s2] = 0
R19  [s3] = 10
```

ASSEMBLY CODE :

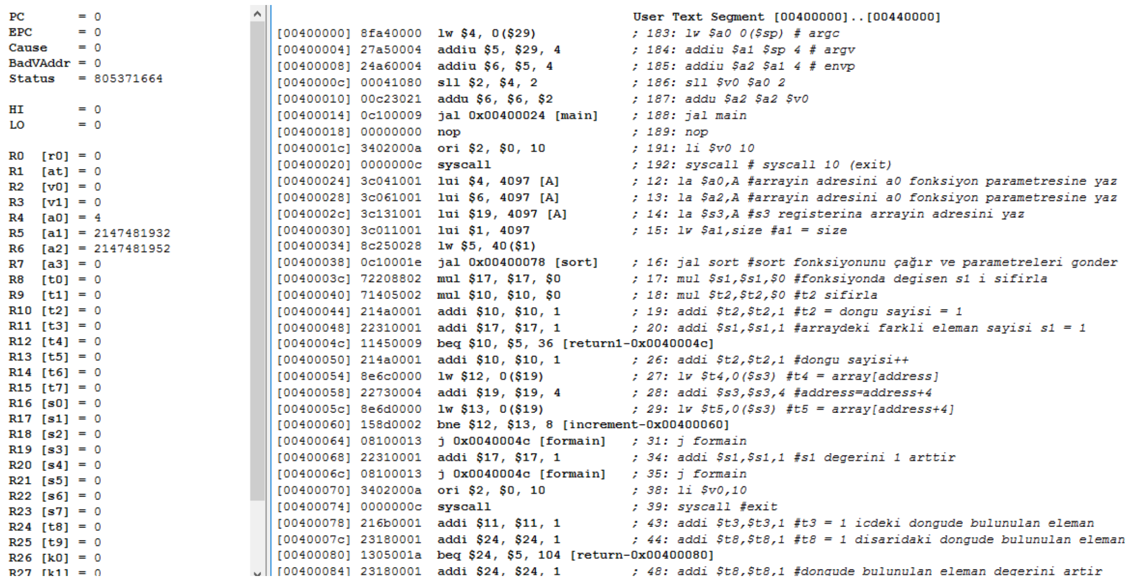
```
main:

    la          $t1, A                  #load array A address in t
    lw          $s0,K                   #load K to s0 register
    lw          $s3,size                #load s3 = size
    addi        $t2,$t2,1               #t2=1 different element number
    addi        $t4,$s3,1               #t4 = size+1
    addi        $t5,$0,1                #t5 = 1

for:

    beq         $t4,$t5,return0         #if size+1. loop return 0
    addi        $t5,$t5,1               #t5+1 t5.loop
    lw          $t6,0($t1)              #t6 = array[address]
    addi        $t1,$t1,4               #address=address+4
    lw          $t7,0($t1)              #t7 = array[address+4]
    bne         $t6,$t7,increment       #t6 != t7 increment t2
    beq         $s0,$t2,return          #K = s0 return
    j for
```

```
increment:

    addi        $t2,$t2,1                  #differentelementnumber+1
    j for

return:

    addi        $s1,$t7,0                  # load finding element to s1 register
    li          $v0,10                     # exit
    syscall

return0:                                   #if no K.th different element

    addi        $s1,$0,0                   #load 0 to s1
    li          $v0,10                     # exit
    syscall
```

2) Number of different values:

sort function has a1,a2,a3 arguments.in main finding the number of different values.result in **$s1** register.

Before Running :

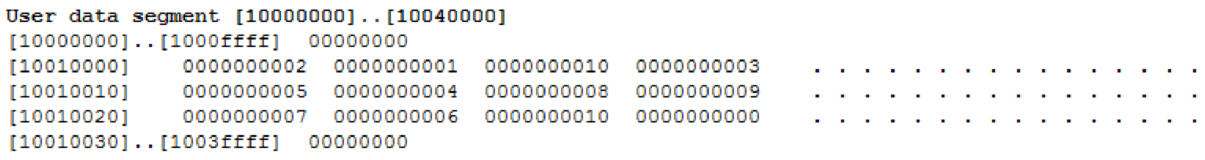


```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 4
R5  [a1] = 2147481932
R6  [a2] = 2147481952
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

```
                                                 User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [A]           ; 12: la $a0,A #arrayin adresini a0 fonksiyon parametresine yaz
[00400028] 3c061001  lui $6, 4097 [A]           ; 13: la $a2,A #arrayin adresini a0 fonksiyon parametresine yaz
[0040002c] 3c131001  lui $19, 4097 [A]          ; 14: la $s3,A #s3 registerina arrayin adresini yaz
[00400030] 3c011001  lui $1, 4097               ; 15: lw $a1,size #a1 = size
[00400034] 8c250028  lw $5, 40($1)
[00400038] 0c10001e  jal 0x00400078 [sort]      ; 16: jal sort #sort fonksiyonunu çağır ve parametreleri gonder
[0040003c] 72208802  mul $17, $17, $0           ; 17: mul $s1,$s1,$0 #fonksiyonda degisen s1 i sifirla
[00400040] 71405002  mul $10, $10, $0           ; 18: mul $t2,$t2,$0 #t2 sifirla
[00400044] 214a0001  addi $10, $10, 1           ; 19: addi $t2,$t2,1 #t2 = dongu sayisi = 1
[00400048] 22310001  addi $17, $17, 1           ; 20: addi $s1,$s1,1 #arraydeki farkli eleman sayisi s1 = 1
[0040004c] 11450009  beq $10, $5, 36 [return1-0x0040004c]
[00400050] 214a0001  addi $10, $10, 1           ; 26: addi $t2,$t2,1 #dongu sayisi++
[00400054] 8e6c0000  lw $12, 0($19)             ; 27: lw $t4,0($s3) #t4 = array[address]
[00400058] 22730004  addi $19, $19, 4           ; 28: addi $s3,$s3,4 #address=address+4
[0040005c] 8e6d0000  lw $13, 0($19)             ; 29: lw $t5,0($s3) #t5 = array[address+4]
[00400060] 158d0002  bne $12, $13, 8 [increment-0x00400060]
[00400064] 08100013  j 0x0040004c [formain]     ; 31: j formain
[00400068] 22310001  addi $17, $17, 1           ; 34: addi $s1,$s1,1 #s1 degerini 1 arttir
[0040006c] 08100013  j 0x0040004c [formain]     ; 35: j formain
[00400070] 3402000a  ori $2, $0, 10             ; 38: li $v0,10
[00400074] 0000000c  syscall                    ; 39: syscall #exit
[00400078] 216b0001  addi $11, $11, 1           ; 43: addi $t3,$t3,1 #t3 = 1 icdeki dongude bulunulan eleman
[0040007c] 23180001  addi $24, $24, 1           ; 44: addi $t8,$t8,1 #t8 = 1 disaridaki dongude bulunulan eleman
[00400080] 1305001a  beq $24, $5, 104 [return-0x00400080]
[00400084] 23180001  addi $24, $24, 1           ; 48: addi $t8,$t8,1 #dongude bulunulan eleman degerini artir
```

Test 1: A={2,1,10,3,5,4,8,9,7,6} :

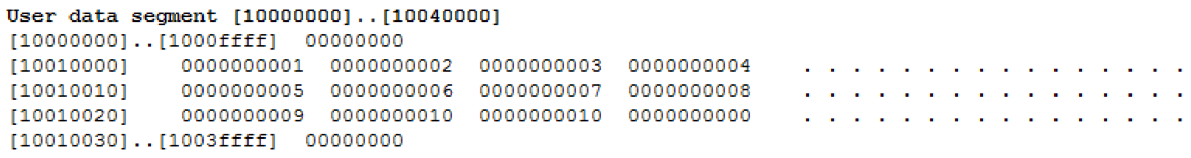
Before running memory :



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]   00000000
[10010000]     0000000002  0000000001  0000000010  0000000003    . . . . . . . . . . . . . . . .
[10010010]     0000000005  0000000004  0000000008  0000000009    . . . . . . . . . . . . . . . .
[10010020]     0000000007  0000000006  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]   00000000
```

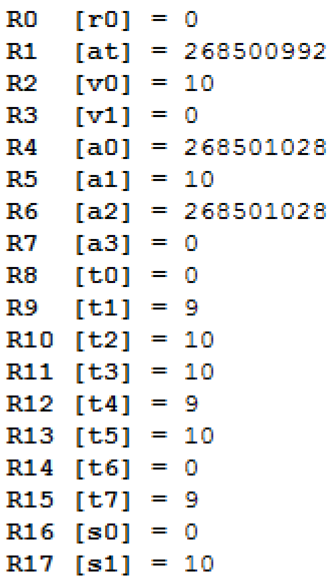
After Running :

Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]   00000000
[10010000]     0000000001  0000000002  0000000003  0000000004    . . . . . . . . . . . . . . . .
[10010010]     0000000005  0000000006  0000000007  0000000008    . . . . . . . . . . . . . . . .
[10010020]     0000000009  0000000010  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]   00000000
```

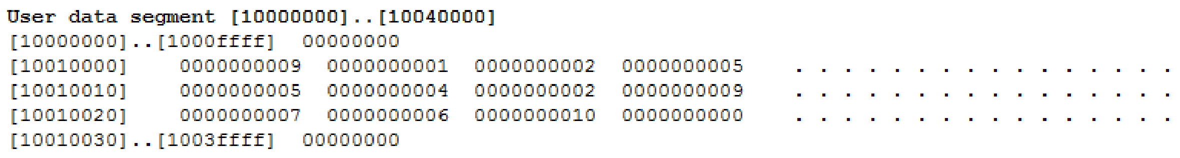
Registers :



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501028
R5  [a1] = 10
R6  [a2] = 268501028
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 9
R10 [t2] = 10
R11 [t3] = 10
R12 [t4] = 9
R13 [t5] = 10
R14 [t6] = 0
R15 [t7] = 9
R16 [s0] = 0
R17 [s1] = 10
```

Test 2: A={9,1,2,5,5,4,2,9,7,6} :

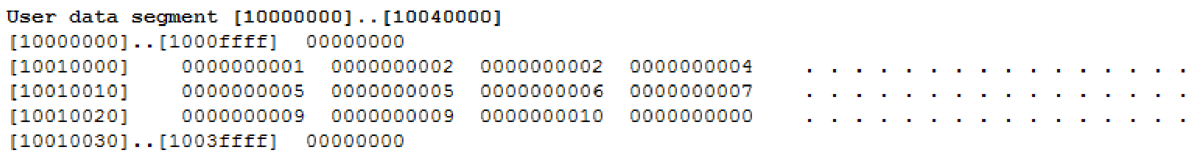
Before running:MEMORY



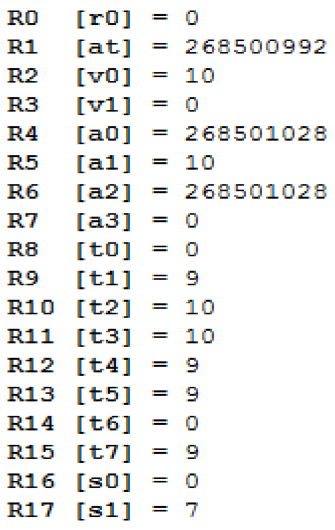
```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]   00000000
[10010000]     0000000009  0000000001  0000000002  0000000005    . . . . . . . . . . . . . . . .
[10010010]     0000000005  0000000004  0000000002  0000000009    . . . . . . . . . . . . . . . .
[10010020]     0000000007  0000000006  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]   00000000
```

After Running :

MEMORY :

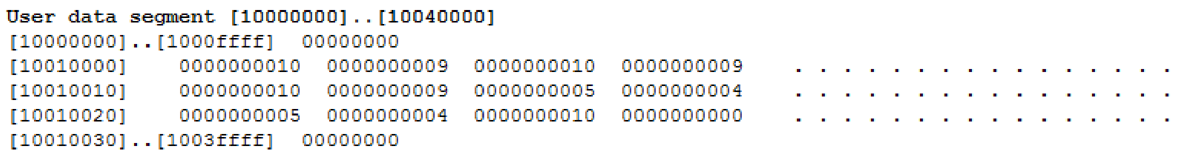

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000002  0000000002  0000000004    . . . . . . . . . . . . . . . .
[10010010]    0000000005  0000000005  0000000006  0000000007    . . . . . . . . . . . . . . . .
[10010020]    0000000009  0000000009  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

REGISTERS:


```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501028
R5  [a1] = 10
R6  [a2] = 268501028
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 9
R10 [t2] = 10
R11 [t3] = 10
R12 [t4] = 9
R13 [t5] = 9
R14 [t6] = 0
R15 [t7] = 9
R16 [s0] = 0
R17 [s1] = 7
```

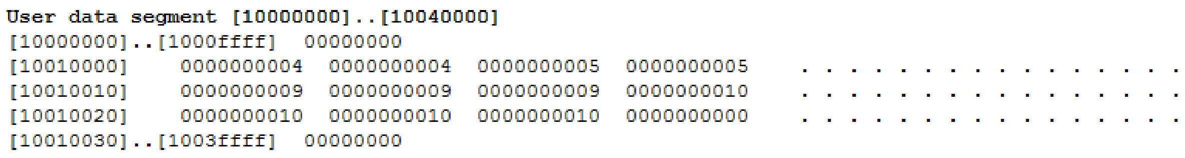
Test 3:A={10,9,10,9,10,9,5,4,5,4}:

<u>Before Running:MEMORY:</u>

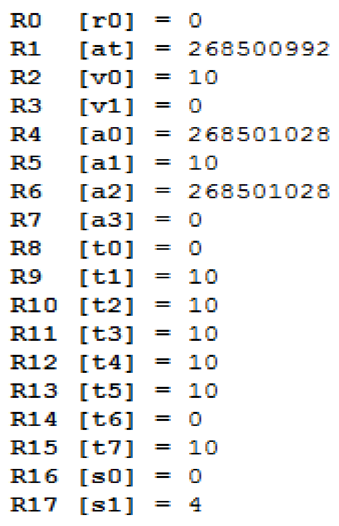

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000010  0000000009  0000000010  0000000009    . . . . . . . . . . . . . . . .
[10010010]    0000000010  0000000009  0000000005  0000000004    . . . . . . . . . . . . . . . .
[10010020]    0000000005  0000000004  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

<u>After Running :</u>

MEMORY:


```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000004  0000000004  0000000005  0000000005    . . . . . . . . . . . . . . . .
[10010010]    0000000009  0000000009  0000000009  0000000010    . . . . . . . . . . . . . . . .
[10010020]    0000000010  0000000010  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

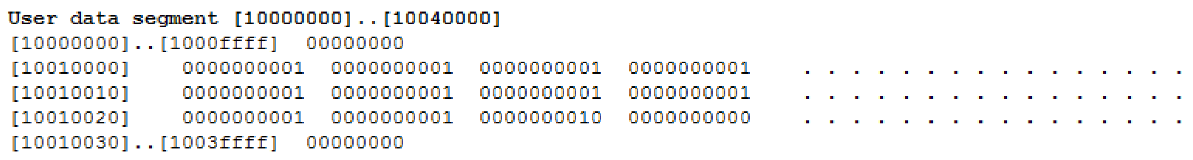
REGISTERS:



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501028
R5  [a1] = 10
R6  [a2] = 268501028
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 10
R10 [t2] = 10
R11 [t3] = 10
R12 [t4] = 10
R13 [t5] = 10
R14 [t6] = 0
R15 [t7] = 10
R16 [s0] = 0
R17 [s1] = 4
```

Test 4:A={1,1,1,1,1,1,1,1,1,1} :

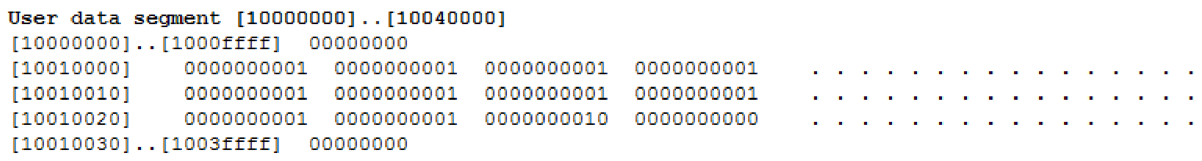
<u>Before Running:MEMORY</u>



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010010]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010020]    0000000001  0000000001  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

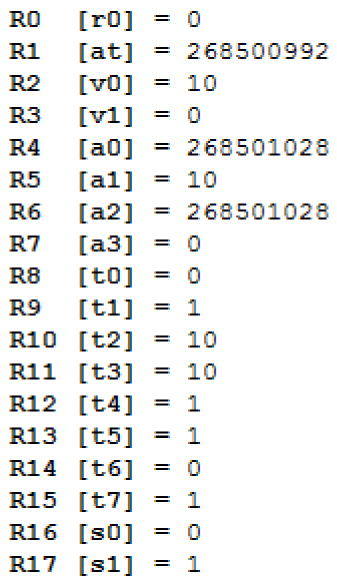
<u>After Running :</u>

MEMORY:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010010]    0000000001  0000000001  0000000001  0000000001    . . . . . . . . . . . . . . . .
[10010020]    0000000001  0000000001  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

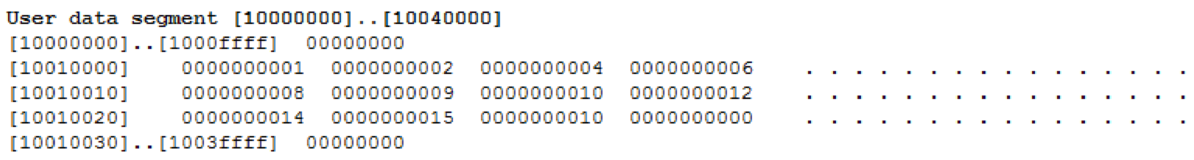
REGISTERS :



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501028
R5  [a1] = 10
R6  [a2] = 268501028
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 1
R10 [t2] = 10
R11 [t3] = 10
R12 [t4] = 1
R13 [t5] = 1
R14 [t6] = 0
R15 [t7] = 1
R16 [s0] = 0
R17 [s1] = 1
```

Test 5: A={1,2,4,6,8,9,10,12,14,15}:

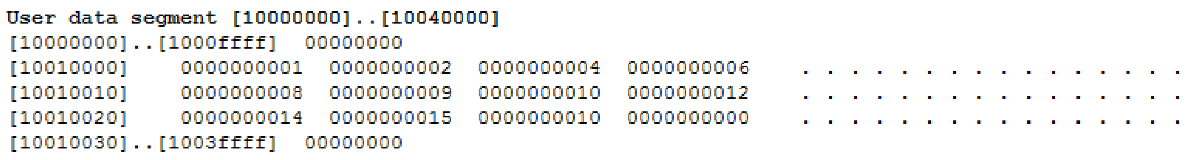
Before Running:MEMORY



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000002  0000000004  0000000006    . . . . . . . . . . . . . . . .
[10010010]    0000000008  0000000009  0000000010  0000000012    . . . . . . . . . . . . . . . .
[10010020]    0000000014  0000000015  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

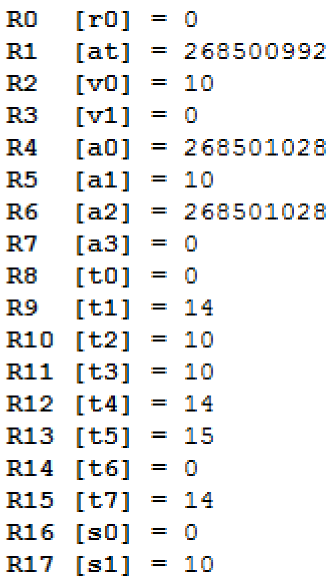
After Running :

MEMORY:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000001  0000000002  0000000004  0000000006    . . . . . . . . . . . . . . . .
[10010010]    0000000008  0000000009  0000000010  0000000012    . . . . . . . . . . . . . . . .
[10010020]    0000000014  0000000015  0000000010  0000000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

REGISTERS:



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501028
R5  [a1] = 10
R6  [a2] = 268501028
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 14
R10 [t2] = 10
R11 [t3] = 10
R12 [t4] = 14
R13 [t5] = 15
R14 [t6] = 0
R15 [t7] = 14
R16 [s0] = 0
R17 [s1] = 10
```

ASSEMBLY CODE :

main:

```
        la          $a0,A           #arrayin adresini a0 fonksiyon parametresine yaz
        la          $a2,A           #arrayin adresini a0 fonksiyon parametresine yaz
        la          $s3,A           #s3 registerina arrayin adresini yaz
        lw          $a1,size        #a1 = size
        jal sort                    #sort fonksiyonunu çağır ve parametreleri gonder
        mul         $s1,$s1,$0      #fonksiyonda degisen s1 i sifirla
        mul         $t2,$t2,$0      #t2 sifirla
        addi        $t2,$t2,1       #t2 = dongu sayisi = 1
        addi        $s1,$s1,1       #arraydeki farkli eleman sayisi s1 = 1
```

```
formain:

        beq     $t2,$a1,return1         # size+1. döngüde return1 labeline git
        addi    $t2,$t2,1               #dongu sayisi++
        lw      $t4,0($s3)              #t4 = array[address]
        addi    $s3,$s3,4               #address=address+4
        lw      $t5,0($s3)              #t5 = array[address+4]
        bne     $t4,$t5,increment       #array[address] != array[address+4] increment labeline git
        j formain

increment:

        addi     $s1,$s1,1              #s1 degerini 1 arttir
        j formain

return1:

        li      $v0,10
        syscall                         #exit

sort:

        addi    $t3,$t3,1               #t3 = 1 icdeki dongude bulunulan eleman
        addi    $t8,$t8,1               #t8 = 1  disaridaki dongude bulunulan eleman

for1:

        beq     $t8,$a1,return          #arrayin son elemanina gelindiyse return labeline git
        addi    $t8,$t8,1               #dongude bulunulan eleman degerini artir
        addi    $t4,$t8,-1
        mul     $t5,$t5,$0              #t5=0
        add     $t5,$t5,$a0             #t5 = a0 (baslangic adresi)
        lw      $t1,0($a0)              #min deger($t1) = array[t4]

for:

        beq     $t3,$a1,minelemanata #arrayin son elemanina gelindiyse mineleman labeline git
        addi    $t3,$t3,1               #dongu degerini artir
        addi    $a2,$a2,4               #a2 adresini artir
        lw      $t2,0($a2)              #t2 = array[t3]
        slt     $s1,$t1,$t2
        beq     $s1,$0,minvalue         #t1>=t2 ise minvalue labeline git
        j for

minvalue:

        mul     $t5,$t5,$0
        add     $t5,$t5,$a2             #t5 = min degerin adresi
        lw      $t1,0($t5)              #minvalue(t1) = kucuk olan deger
        j for
```

```
minelemanata:

        lw      $t7,0($a0)              #array[t4] degerini temp registera($t7) ata
        sw      $t7,0($t5)              #t7 registerındaki degeri bulunan min degerin bulundugu
        adrese ata
        sw      $t1,0($a0)              #array[t4] adresine bulunan min eleman degerini ata
        addi    $a0,$a0,4               #disaridaki array üzerinde bir ilerle (i yi arttır)
        mul     $a2,$a2,$0
        add     $a2,$a2,$a0             #ic dongude bulunulan elemanın adresidıs dongudekine esitle
        mul     $t3,$t3,$0
        add     $t3,$t3,$t8             #ic dongude bulunulan elemanı dıs dongudekine esitle
        j for1

return:

        jr $ra                          #maine don
```