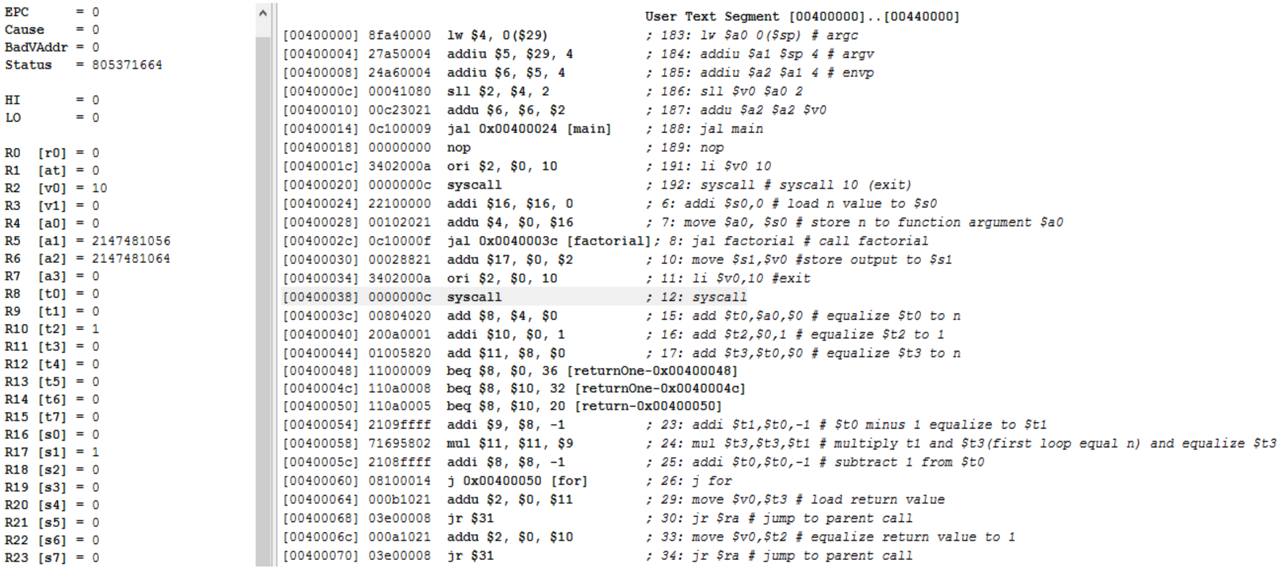
BURCU İSKENDER
21328103

COMPUTER ORGANIZATION – Homework 2

Factorial :

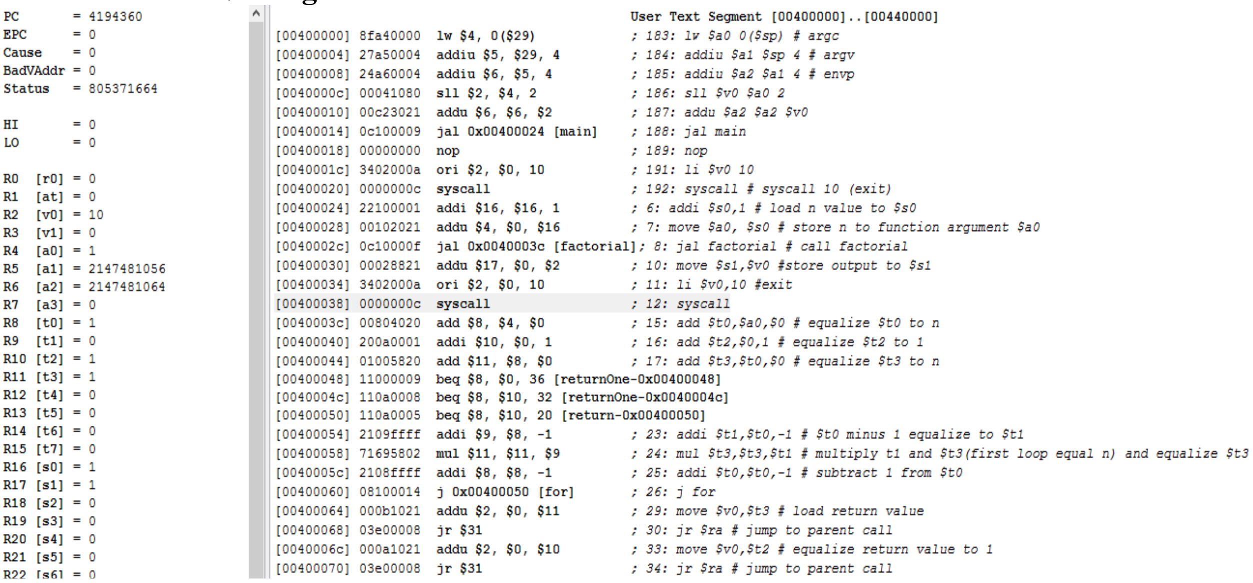
N = 0 /Result in $s1 register.



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

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 2147481056
R6  [a2] = 2147481064
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 1
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 1
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 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] 22100000  addi $16, $16, 0        ; 6: addi $s0,0 # load n value to $s0
[00400028] 00102021  addu $4, $0, $16        ; 7: move $a0, $s0 # store n to function argument $a0
[0040002c] 0c10000f  jal 0x0040003c [factorial]; 8: jal factorial # call factorial
[00400030] 00028821  addu $17, $0, $2        ; 10: move $s1,$v0 #store output to $s1
[00400034] 3402000a  ori $2, $0, 10          ; 11: li $v0,10 #exit
[00400038] 0000000c  syscall                 ; 12: syscall
[0040003c] 00804020  add $8, $4, $0          ; 15: add $t0,$a0,$0 # equalize $t0 to n
[00400040] 200a0001  addi $10, $0, 1         ; 16: add $t2,$0,1 # equalize $t2 to 1
[00400044] 01005820  add $11, $8, $0         ; 17: add $t3,$t0,$0 # equalize $t3 to n
[00400048] 11000009  beq $8, $0, 36 [returnOne-0x00400048]
[0040004c] 110a0008  beq $8, $10, 32 [returnOne-0x0040004c]
[00400050] 110a0005  beq $8, $10, 20 [return-0x00400050]
[00400054] 2109ffff  addi $9, $8, -1         ; 23: addi $t1,$t0,-1 # $t0 minus 1 equalize to $t1
[00400058] 71695802  mul $11, $11, $9        ; 24: mul $t3,$t3,$t1 # multiply t1 and $t3(first loop equal n) and equalize $t3
[0040005c] 2108ffff  addi $8, $8, -1         ; 25: addi $t0,$t0,-1 # subtract 1 from $t0
[00400060] 08100014  j 0x00400050 [for]      ; 26: j for
[00400064] 000b1021  addu $2, $0, $11        ; 29: move $v0,$t3 # load return value
[00400068] 03e00008  jr $31                  ; 30: jr $ra # jump to parent call
[0040006c] 000a1021  addu $2, $0, $10        ; 33: move $v0,$t2 # equalize return value to 1
[00400070] 03e00008  jr $31                  ; 34: jr $ra # jump to parent call
```

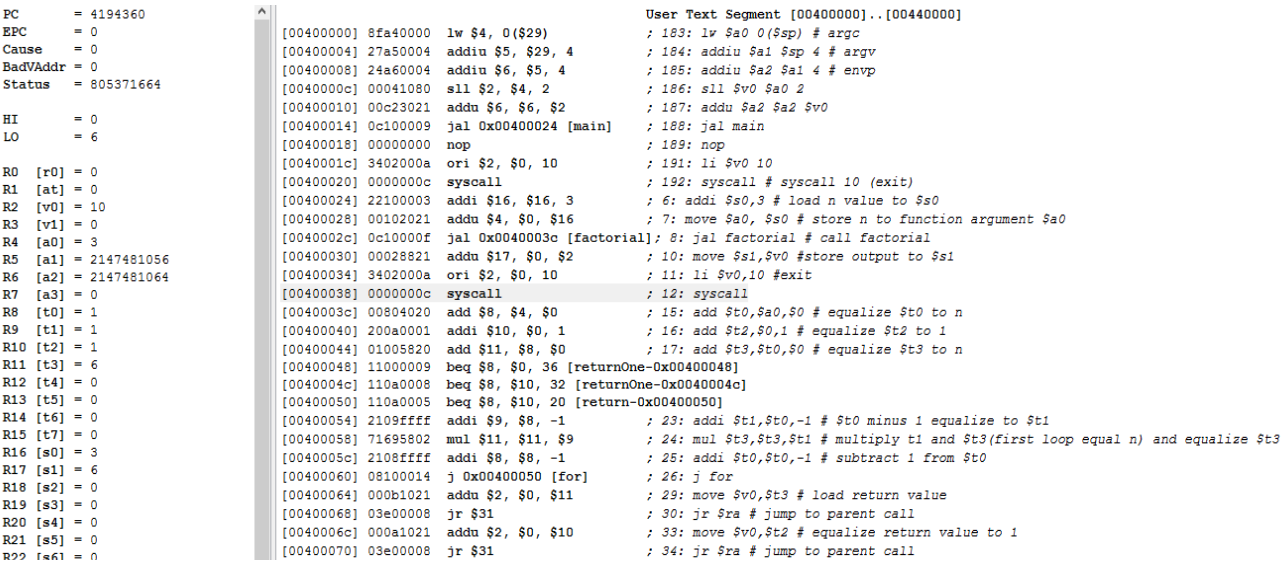
N= 1 / Result in $s1 register



```
PC       = 4194360
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481056
R6  [a2] = 2147481064
R7  [a3] = 0
R8  [t0] = 1
R9  [t1] = 0
R10 [t2] = 1
R11 [t3] = 1
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 1
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 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] 22100001  addi $16, $16, 1        ; 6: addi $s0,1 # load n value to $s0
[00400028] 00102021  addu $4, $0, $16        ; 7: move $a0, $s0 # store n to function argument $a0
[0040002c] 0c10000f  jal 0x0040003c [factorial]; 8: jal factorial # call factorial
[00400030] 00028821  addu $17, $0, $2        ; 10: move $s1,$v0 #store output to $s1
[00400034] 3402000a  ori $2, $0, 10          ; 11: li $v0,10 #exit
[00400038] 0000000c  syscall                 ; 12: syscall
[0040003c] 00804020  add $8, $4, $0          ; 15: add $t0,$a0,$0 # equalize $t0 to n
[00400040] 200a0001  addi $10, $0, 1         ; 16: add $t2,$0,1 # equalize $t2 to 1
[00400044] 01005820  add $11, $8, $0         ; 17: add $t3,$t0,$0 # equalize $t3 to n
[00400048] 11000009  beq $8, $0, 36 [returnOne-0x00400048]
[0040004c] 110a0008  beq $8, $10, 32 [returnOne-0x0040004c]
[00400050] 110a0005  beq $8, $10, 20 [return-0x00400050]
[00400054] 2109ffff  addi $9, $8, -1         ; 23: addi $t1,$t0,-1 # $t0 minus 1 equalize to $t1
[00400058] 71695802  mul $11, $11, $9        ; 24: mul $t3,$t3,$t1 # multiply t1 and $t3(first loop equal n) and equalize $t3
[0040005c] 2108ffff  addi $8, $8, -1         ; 25: addi $t0,$t0,-1 # subtract 1 from $t0
[00400060] 08100014  j 0x00400050 [for]      ; 26: j for
[00400064] 000b1021  addu $2, $0, $11        ; 29: move $v0,$t3 # load return value
[00400068] 03e00008  jr $31                  ; 30: jr $ra # jump to parent call
[0040006c] 000a1021  addu $2, $0, $10        ; 33: move $v0,$t2 # equalize return value to 1
[00400070] 03e00008  jr $31                  ; 34: jr $ra # jump to parent call
```

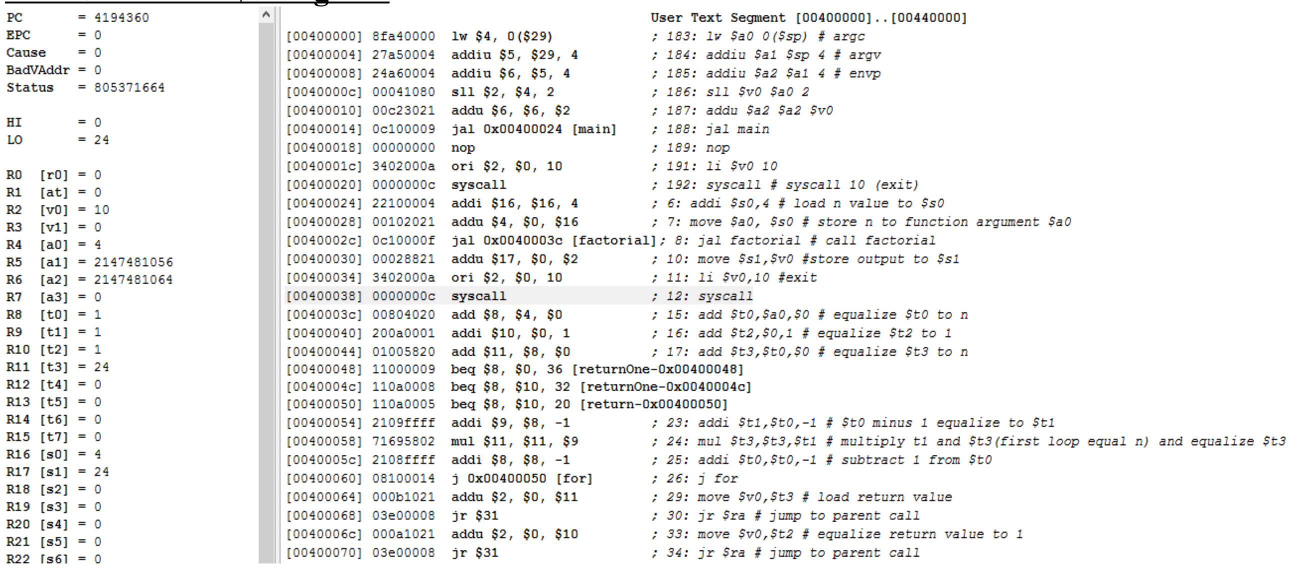
N = 3 / Result in $s1 register



```
PC       = 4194360
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 6

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 2147481056
R6  [a2] = 2147481064
R7  [a3] = 0
R8  [t0] = 1
R9  [t1] = 1
R10 [t2] = 1
R11 [t3] = 6
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 3
R17 [s1] = 6
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 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] 22100003  addi $16, $16, 3        ; 6: addi $s0,3 # load n value to $s0
[00400028] 00102021  addu $4, $0, $16        ; 7: move $a0, $s0 # store n to function argument $a0
[0040002c] 0c10000f  jal 0x0040003c [factorial]; 8: jal factorial # call factorial
[00400030] 00028821  addu $17, $0, $2        ; 10: move $s1,$v0 #store output to $s1
[00400034] 3402000a  ori $2, $0, 10          ; 11: li $v0,10 #exit
[00400038] 0000000c  syscall                 ; 12: syscall
[0040003c] 00804020  add $8, $4, $0          ; 15: add $t0,$a0,$0 # equalize $t0 to n
[00400040] 200a0001  addi $10, $0, 1         ; 16: add $t2,$0,1 # equalize $t2 to 1
[00400044] 01005820  add $11, $8, $0         ; 17: add $t3,$t0,$0 # equalize $t3 to n
[00400048] 11000009  beq $8, $0, 36 [returnOne-0x00400048]
[0040004c] 110a0008  beq $8, $10, 32 [returnOne-0x0040004c]
[00400050] 110a0005  beq $8, $10, 20 [return-0x00400050]
[00400054] 2109ffff  addi $9, $8, -1         ; 23: addi $t1,$t0,-1 # $t0 minus 1 equalize to $t1
[00400058] 71695802  mul $11, $11, $9        ; 24: mul $t3,$t3,$t1 # multiply t1 and $t3(first loop equal n) and equalize $t3
[0040005c] 2108ffff  addi $8, $8, -1         ; 25: addi $t0,$t0,-1 # subtract 1 from $t0
[00400060] 08100014  j 0x00400050 [for]      ; 26: j for
[00400064] 000b1021  addu $2, $0, $11        ; 29: move $v0,$t3 # load return value
[00400068] 03e00008  jr $31                  ; 30: jr $ra # jump to parent call
[0040006c] 000a1021  addu $2, $0, $10        ; 33: move $v0,$t2 # equalize return value to 1
[00400070] 03e00008  jr $31                  ; 34: jr $ra # jump to parent call
```

N = 4 / Result in $s1 register



```
PC       = 4194360
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 24

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 4
R5  [a1] = 2147481056
R6  [a2] = 2147481064
R7  [a3] = 0
R8  [t0] = 1
R9  [t1] = 1
R10 [t2] = 1
R11 [t3] = 24
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 4
R17 [s1] = 24
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 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] 22100004  addi $16, $16, 4          ; 6: addi $s0,4 # load n value to $s0
[00400028] 00102021  addu $4, $0, $16          ; 7: move $a0, $s0 # store n to function argument $a0
[0040002c] 0c10000f  jal 0x0040003c [factorial]; 8: jal factorial # call factorial
[00400030] 00028821  addu $17, $0, $2          ; 10: move $s1,$v0 #store output to $s1
[00400034] 3402000a  ori $2, $0, 10            ; 11: li $v0,10 #exit
[00400038] 0000000c  syscall                   ; 12: syscall
[0040003c] 00804020  add $8, $4, $0            ; 15: add $t0,$a0,$0 # equalize $t0 to n
[00400040] 200a0001  addi $10, $0, 1           ; 16: add $t2,$0,1 # equalize $t2 to 1
[00400044] 01005820  add $11, $8, $0           ; 17: add $t3,$t0,$0 # equalize $t3 to n
[00400048] 11000009  beq $8, $0, 36 [returnOne-0x00400048]
[0040004c] 110a0008  beq $8, $10, 32 [returnOne-0x0040004c]
[00400050] 110a0005  beq $8, $10, 20 [return-0x00400050]
[00400054] 2109ffff  addi $9, $8, -1           ; 23: addi $t1,$t0,-1 # $t0 minus 1 equalize to $t1
[00400058] 71695802  mul $11, $11, $9          ; 24: mul $t3,$t3,$t1 # multiply t1 and $t3(first loop equal n) and equalize $t3
[0040005c] 2108ffff  addi $8, $8, -1           ; 25: addi $t0,$t0,-1 # subtract 1 from $t0
[00400060] 08100014  j 0x00400050 [for]        ; 26: j for
[00400064] 000b1021  addu $2, $0, $11          ; 29: move $v0,$t3 # load return value
[00400068] 03e00008  jr $31                    ; 30: jr $ra # jump to parent call
[0040006c] 000a1021  addu $2, $0, $10          ; 33: move $v0,$t2 # equalize return value to 1
[00400070] 03e00008  jr $31                    ; 34: jr $ra # jump to parent call
```

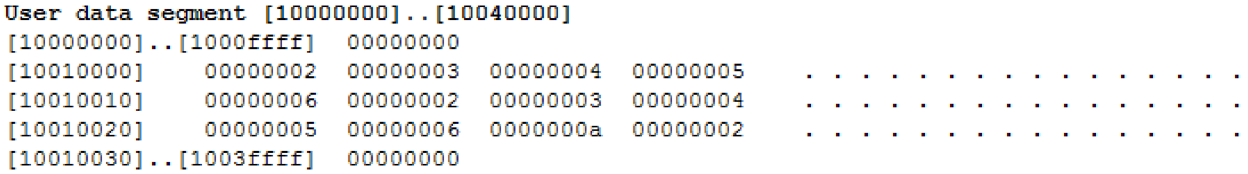
```
main function
1) load n value to $s0
2) store n to function argument $a0
3)call factorial
4)store output to $s1
5)exit program

factorial function
1)equalize $t0 to n
2)equalize $t2 to 1
3) equalize $t3 to n
4)if n($t0) equal 0 branch returnOne label
5)if n($t0) equal 1 branch returnOne label
        loop:
        1) if $t0 equal 1 branch to return label
        2) $t0 minus 1 equalize to $t1
        3) multiply t1 and $t3(first loop equal n) and equalize $t3
        4)subtract 1 from $t0
return:
1) load return value
2)jump to main
returnOne:
1)equalize return value to 1
2)jump to main
```

2)Key in an array

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

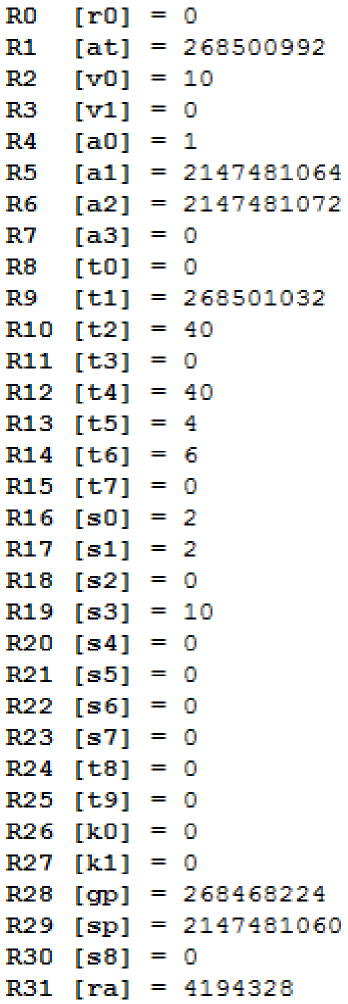
Test 1: A={2,3,4,5,6,2,3,4,5,6}, K=2

Before running - Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000002  00000003  00000004  00000005    . . . . . . . . . . . . . . . .
[10010010]    00000006  00000002  00000003  00000004    . . . . . . . . . . . . . . . .
[10010020]    00000005  00000006  0000000a  00000002    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

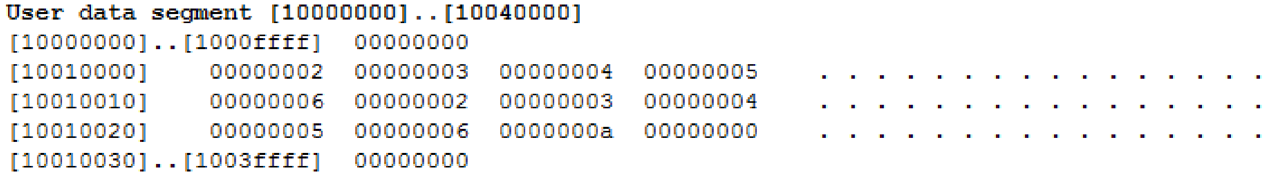
After running registers :



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481064
R6  [a2] = 2147481072
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 268501032
R10 [t2] = 40
R11 [t3] = 0
R12 [t4] = 40
R13 [t5] = 4
R14 [t6] = 6
R15 [t7] = 0
R16 [s0] = 2
R17 [s1] = 2
R18 [s2] = 0
R19 [s3] = 10
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147481060
R30 [s8] = 0
R31 [ra] = 4194328
```

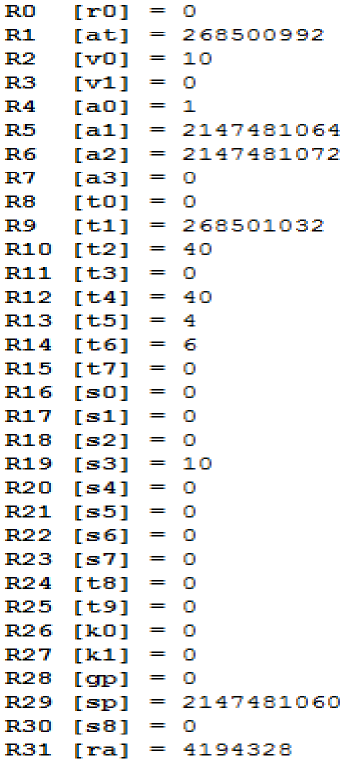
Test 2: A={2,3,4,5,6,2,3,4,5,6}, K=0

Before running - Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000002  00000003  00000004  00000005    . . . . . . . . . . . . . . . .
[10010010]    00000006  00000002  00000003  00000004    . . . . . . . . . . . . . . . .
[10010020]    00000005  00000006  0000000a  00000000    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

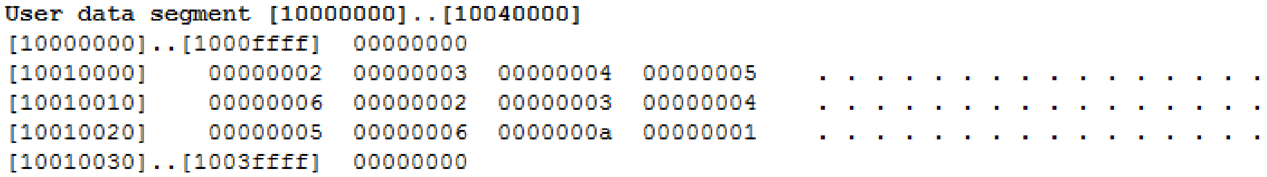
After running – Registers:



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481064
R6  [a2] = 2147481072
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 268501032
R10 [t2] = 40
R11 [t3] = 0
R12 [t4] = 40
R13 [t5] = 4
R14 [t6] = 6
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 10
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 0
R29 [sp] = 2147481060
R30 [s8] = 0
R31 [ra] = 4194328
```

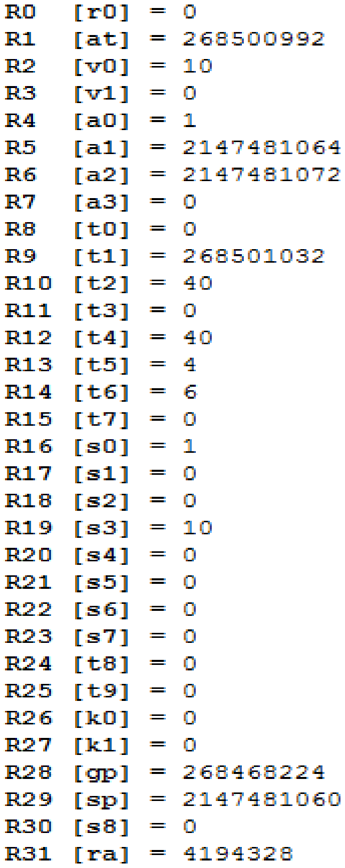
<u>Test 3:A={1,1,1,1,1,1,1,1,1,1}, K=1</u>

Before Running – Memory



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000002  00000003  00000004  00000005    . . . . . . . . . . . . . . . .
[10010010]    00000006  00000002  00000003  00000004    . . . . . . . . . . . . . . . .
[10010020]    00000005  00000006  0000000a  00000001    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

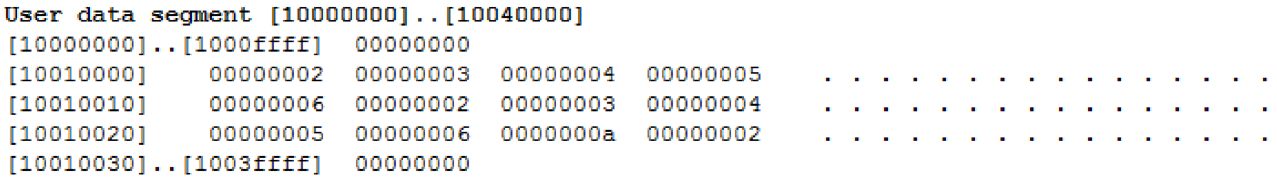
After Running – Registers



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481064
R6  [a2] = 2147481072
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 268501032
R10 [t2] = 40
R11 [t3] = 0
R12 [t4] = 40
R13 [t5] = 4
R14 [t6] = 6
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 10
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147481060
R30 [s8] = 0
R31 [ra] = 4194328
```

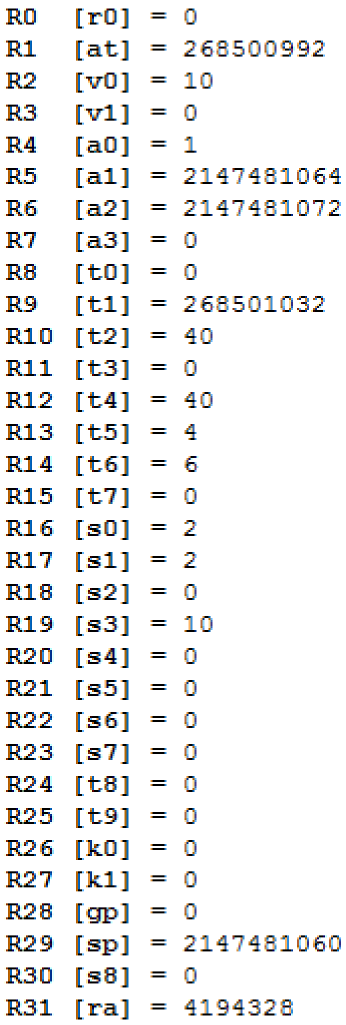
Test 4:A={1,1,1,1,1,1,1,1,1,1}, K=2

Before running – Memory :



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000002  00000003  00000004  00000005    . . . . . . . . . . . . . . . .
[10010010]    00000006  00000002  00000003  00000004    . . . . . . . . . . . . . . . .
[10010020]    00000005  00000006  0000000a  00000002    . . . . . . . . . . . . . . . .
[10010030]..[1003ffff]  00000000
```

After running – Registers:



```
R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481064
R6  [a2] = 2147481072
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 268501032
R10 [t2] = 40
R11 [t3] = 0
R12 [t4] = 40
R13 [t5] = 4
R14 [t6] = 6
R15 [t7] = 0
R16 [s0] = 2
R17 [s1] = 2
R18 [s2] = 0
R19 [s3] = 10
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 0
R29 [sp] = 2147481060
R30 [s8] = 0
R31 [ra] = 4194328
```

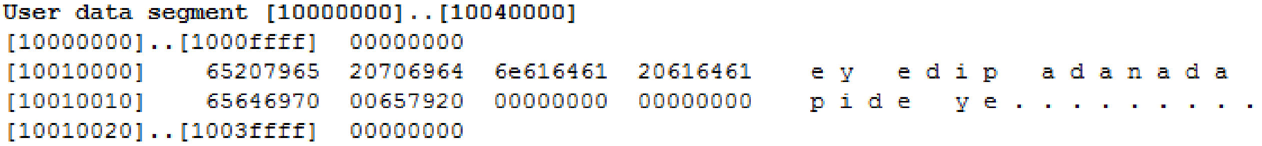
```
main function
1) store key K in $s0
2) store 0 in $t2
3) number of key in array $s1 = 0
4) s3 = size
5) store 4 in $t5
6) t4 = sizex4 (total array size )
        for:
        1)arraysize*4 = t2 return
        2)$t6 = array[x]
        3)address = address+4,$t2 = $t2+4
        4)if array[x]== key branch to increment
increment label:
        numberofkeyinarray=numberofkeyinarray+1
    7)  exit
```

3)Palindrome:

*If it is palindrome, store 1 to register s7. Otherwise, store 0 to s7 register.

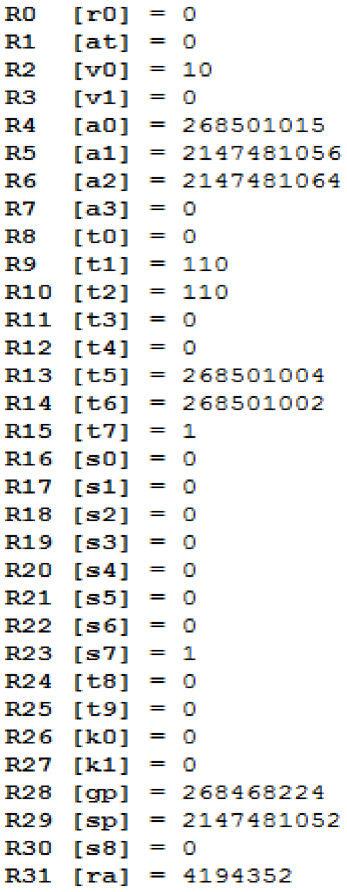
Test1: str= “ey edip adanada pide ye”

Before running – Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    65207965  20706964  6e616461  20616461    e y   e d i p   a d a n a d a
[10010010]    65646970  00657920  00000000  00000000    p i d e   y e . . . . . . . . . .
[10010020]..[1003ffff]  00000000
```

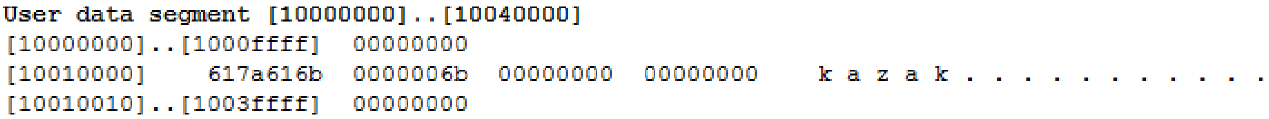
After running – Registers:



```
R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501015
R5  [a1] = 2147481056
R6  [a2] = 2147481064
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 110
R10 [t2] = 110
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 268501004
R14 [t6] = 268501002
R15 [t7] = 1
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 1
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147481052
R30 [s8] = 0
R31 [ra] = 4194352
```

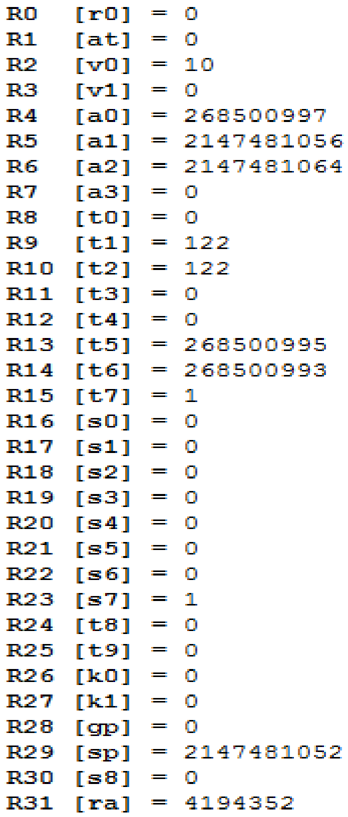
Test2: str= “kazak”

Before running – Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    617a616b  0000006b  00000000  00000000    k a z a k . . . . . . . . . . .
[10010010]..[1003ffff]  00000000
```

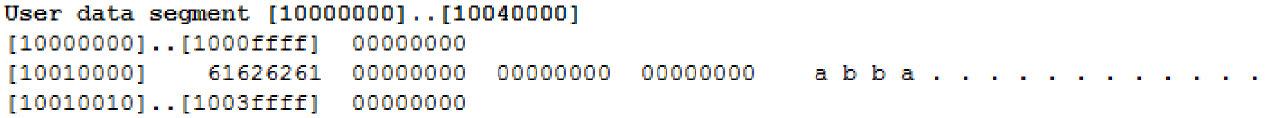
After running – Registers:



```
R0   [r0] = 0
R1   [at] = 0
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 268500997
R5   [a1] = 2147481056
R6   [a2] = 2147481064
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 122
R10  [t2] = 122
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 268500995
R14  [t6] = 268500993
R15  [t7] = 1
R16  [s0] = 0
R17  [s1] = 0
R18  [s2] = 0
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
R22  [s6] = 0
R23  [s7] = 1
R24  [t8] = 0
R25  [t9] = 0
R26  [k0] = 0
R27  [k1] = 0
R28  [gp] = 0
R29  [sp] = 2147481052
R30  [s8] = 0
R31  [ra] = 4194352
```

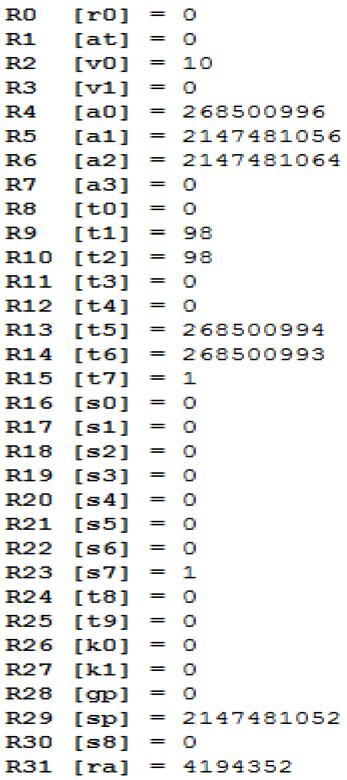
Test3: str= “abba”

Before running – Memory:



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    61626261  00000000  00000000  00000000    a b b a . . . . . . . . . . . .
[10010010]..[1003ffff]  00000000
```

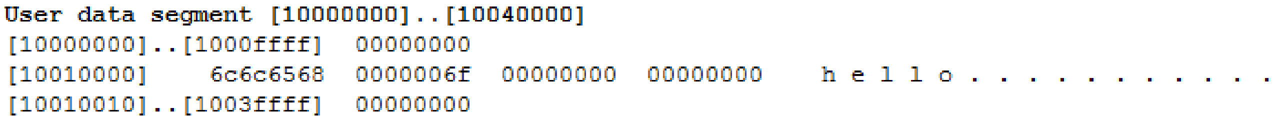
After running – Registers:



```
R0   [r0] = 0
R1   [at] = 0
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 268500996
R5   [a1] = 2147481056
R6   [a2] = 2147481064
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 98
R10  [t2] = 98
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 268500994
R14  [t6] = 268500993
R15  [t7] = 1
R16  [s0] = 0
R17  [s1] = 0
R18  [s2] = 0
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
R22  [s6] = 0
R23  [s7] = 1
R24  [t8] = 0
R25  [t9] = 0
R26  [k0] = 0
R27  [k1] = 0
R28  [gp] = 0
R29  [sp] = 2147481052
R30  [s8] = 0
R31  [ra] = 4194352
```

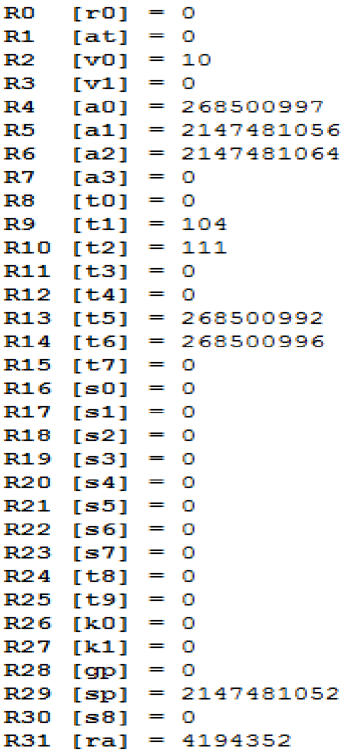
Test3: str= “hello”

Before running – Memory :



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]     6c6c6568  0000006f  00000000  00000000    h e l l o . . . . . . . . . . .
[10010010]..[1003ffff]  00000000
```

After running – Registers :



```
R0   [r0] = 0
R1   [at] = 0
R2   [v0] = 10
R3   [v1] = 0
R4   [a0] = 268500997
R5   [a1] = 2147481056
R6   [a2] = 2147481064
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 104
R10  [t2] = 111
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 268500992
R14  [t6] = 268500996
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
R28  [gp] = 0
R29  [sp] = 2147481052
R30  [s8] = 0
R31  [ra] = 4194352
```

```
main function:
1)function argument $a0 = str address
2)call length function load $v0 register length
3) $s7 = palindrome = 1
4) $v0 = length-1
5) $t5 = straddress
6) $t6 = last char address
    loop:
    1) last char address < first char address return
    2) $t1 =  str[first]
    3) $t2 = str[last]
    4)str[first] not equal str[last]  branch to returnZero
    5) first=first+1 (according to address)
    6)last= last-1 (according to address)
returnZero label:
    1) $s7 = 0 NOT polindorome
    2)exit
return: - exiy

length function:
    1) length = 0
    loop:
        1) if $t1 = null branch to exit label
        2)length = length+1
        3)increment to address
```