

BÁO CÁO THỰC HÀNH TUẦN 7

Lương Thị Tâm – 20194663

Bài 1:

Code:

```

1 .text
2 main: li $a0, -45      #load input parameter
3 jal abs                # jump and link to abs procedure
4 nop
5 add $s0, $zero, $v0
6 li $v0, 10              #terminate
7 syscall
8 endmain:
9 abs: sub $v0, $zero, $a0 #put -a(0) in v0; in case (a0)<0
10 bltz $a0, done # if a0<0 then done
11 nop
12 add $v0, $a0, $zero # else put a0 in v0
13 done: jr $ra

```

Kết quả chạy:

The screenshot shows a debugger interface with several windows:

- Text Segment:** Shows assembly code with line numbers and comments.
- Registers:** Shows寄存器 (Registers) with values for \$zero, \$t0, \$v0, \$a0, \$s0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, pc, hi, and lo.
- Labels:** Shows labels and their addresses: main (0x00400000), endmain (0x00400010), done (0x00400028).
- Data Segment:** Shows memory dump from address 0x10010000 to 0x10010040.
- Registers:** Shows寄存器 (Registers) with values for \$zero, \$t0, \$v0, \$a0, \$s0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, pc, hi, and lo.

Bài 2:

Code:

```

mips1.asm mips2.asm
1 .text
2 main: li $a0, 2 #load test input
3 li $a1, 6
4 li $a2, 9
5 jal max #call max procedure
6 nop
7 li $v0, 10
8 syscall
9 endmain:
10
11 max: add $v0, $a0, $zero # copy a0 in v0; largest so far
12 sub $t0, $a1, $v0 # compute a1 - v0
13 bltz $t0, okay # if a1 - v0 < 0 then no change
14 nop
15 add $v0, $a1, $zero # else a1 is largest thus far
16 okay: sub $t0, $a2, $v0 # compute a2 - v0
17 bltz $t0, done # if a2 - v0 < 0 then no change
18 nop
19 add $v0, $a2, $zero # else a2 is largest overall
20 done: jr $ra # return to calling program
21

```

Kết quả chạy:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000000	0x24100005addiu \$16,\$0,0x000000... 2: main: li \$s0, 5		
	0x04000004	0x2411000aaddiu \$17,\$0,0x000000... 3: li \$s1, 10		
	0x04000008	0x24000005jal \$0,0x00000004 4: jal stack		
	0x0400000c	0x01000001jal 0x0400001c 5: jal max		
	0x04000010	0x05000005nop 6: nopl		
	0x04000014	0x2420000aaddiu \$2,\$0,0x00000000 7: li \$v0, 10		
	0x04000018	0x00000000syscall 8: syscall		
	0x04000020	0x00000020add \$2,\$4,\$0 11: max: add \$v0, \$s0, \$zero # copy a0 in v0: largest so far		
	0x04000024	0x00000000addiu \$8,\$5,\$2 12: sub \$t0, \$s1, \$v0 # compute a1 - v0		
	0x04000028	0x00000000addiu \$1,\$0,0x00000002 13: mult \$t0, okay # if a1 < v0 then no change		
	0x0400002c	0x00000000nop 14: nopl		
	0x04000030	0x00000020add \$2,\$5,\$0 15: add \$v0, \$s1, \$zero # else a1 is largest thus far		
	0x04000034	0x00000000addiu \$8,\$0,\$2 16: okay: sub \$t0, \$s2, \$v0 # compute a2 - v0		
	0x04000038	0x00000000addiu \$1,\$0,0x00000002 17: mult \$t0, \$t0, done # if a2 > v0 < 0 then no change		
	0x0400003c	0x00000000nop 18: nopl		
	0x04000040	0x00000020add \$2,\$6,\$0 19: add \$v0, \$s2, \$zero unless a2 is largest overall		
	0x04000044	0x03e00008jr \$31 20: done: jr \$ra # return to calling program		

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Labels

Label	Address
main	0x04000000
endmain	0x04000014
okay	0x04000030
done	0x04000040

Name Number Value

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000001
\$v0	2	0x00000002
\$v1	3	0x00000003
\$a0	4	0x00000004
\$a1	5	0x00000005
\$a2	6	0x00000006
\$a3	7	0x00000007
\$t0	8	0x00000008
\$t1	9	0x00000009
\$t2	10	0x0000000a
\$t3	11	0x0000000b
\$t4	12	0x0000000c
\$t5	13	0x0000000d
\$t6	14	0x0000000e
\$t7	15	0x0000000f
\$s0	16	0x00000010
\$s1	17	0x00000011
\$s2	18	0x00000012
\$s3	19	0x00000013
\$s4	20	0x00000014
\$s5	21	0x00000015
\$s6	22	0x00000016
\$s7	23	0x00000017
\$t8	24	0x00000018
\$t9	25	0x00000019
\$k0	26	0x0000001a
\$k1	27	0x0000001b
\$gp	28	0x10000000
\$sp	29	0x7fffffe0
\$fp	30	0x00000000
\$ra	31	0x0400010
pc		0x040001c
hi		0x00000000
lo		0x00000000

Bài 3: Code:

mips1.asm mips2.asm mips3.asm

```

1 .text
2 main: li $s0, 5
3 li $s1, 10
4 jal stack
5 addi $v0, $zero, 10
6 syscall
7 endmain:
8 stack:
9 push: addi $sp, $sp, -8 #adjust the stack pointer
10 sw $s0, 4($sp) #push $s0 to stack
11 sw $s1, 0($sp) # push $s1 to stack
12 work: nop
13 nop
14 pop: lw $s0, 0($sp) # pop from stack to $s0
15 lw $s1, 4($sp) # pop from stack to $s1
16 addi $sp, $sp, 8 # adjust the stack pointer
17 jr $ra

```

Kết quả chạy:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000000	0x24100005addiu \$16,\$0,0x000000... 2: main: li \$s0, 5		
	0x04000004	0x2411000aaddiu \$17,\$0,0x000000... 3: li \$s1, 10		
	0x04000008	0x24000005jal \$0,0x00000004 4: jal stack		
	0x0400000c	0x05000005nop 5: nopl		
	0x04000010	0x00000000syscall 6: syscall		
	0x04000014	0x2b3dff18addiu \$29,129,0xffff... 9: push: addi \$sp, \$sp, -8 #adjust the stack pointer		
	0x04000018	0x00000000sw \$s0, 4(\$sp) #push \$s0 to stack 10: sw \$s0, 4(\$sp) #push \$s0 to stack		
	0x0400001c	0x00000000lw \$s1, 0(\$sp) # push \$s1 to stack 11: lw \$s1, 0(\$sp) # push \$s1 to stack		
	0x04000020	0x00000000nop 12: work: nopl 13: nopl		
	0x04000024	0x00000000nop 14: nopl		
	0x04000028	0x00000000lw \$s0, 0(\$sp) # pop from stack to \$s0 15: pop: lw \$s0, 0(\$sp) # pop from stack to \$s0		
	0x04000032	0x00000000lw \$s1, 4(\$sp) # pop from stack to \$s1 16: lw \$s1, 4(\$sp) # pop from stack to \$s1		
	0x04000036	0x00000000addi \$sp, \$sp, 8 # adjust the stack pointer 17: addi \$sp, \$sp, 8 # adjust the stack pointer		
	0x04000040	0x03e00008jr \$31 18: jr \$ra		

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Labels

Label	Address
main	0x04000000
stack	0x04000014
push	0x04000020
work	0x04000020
pop	0x0400002c

Name Number Value

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000001
\$v0	2	0x00000002
\$v1	3	0x00000003
\$a0	4	0x00000004
\$a1	5	0x00000005
\$a2	6	0x00000006
\$t0	7	0x00000007
\$t1	8	0x00000008
\$t2	9	0x00000009
\$t3	10	0x0000000a
\$t4	11	0x0000000b
\$t5	12	0x0000000c
\$t6	13	0x0000000d
\$t7	14	0x0000000e
\$t8	15	0x0000000f
\$t9	16	0x00000010
\$t10	17	0x00000011
\$t11	18	0x00000012
\$t12	19	0x00000013
\$s0	20	0x00000014
\$s1	21	0x00000015
\$s2	22	0x00000016
\$s3	23	0x00000017
\$s4	24	0x00000018
\$s5	25	0x00000019
\$k0	26	0x0000001a
\$k1	27	0x0000001b
\$sp	28	0x10000000
\$fp	29	0x7fffffe0
\$t13	30	0x00000000
\$t14	31	0x0400010
hi		0x00000000
lo		0x00000000

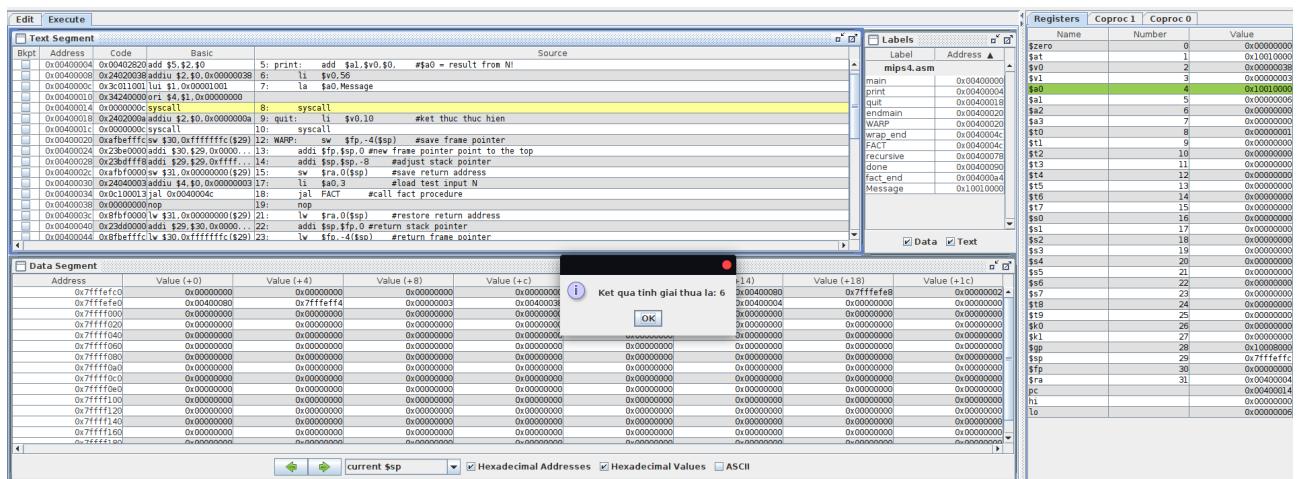
Bài 4:

a, Khi dùng \$sp:

Code:

```
mips1.asm mips2.asm mips3.asm mips4.asm
1 .data
2 Message: .asciiz "Kết quả tính giải thua là: "
3 .text
4 main: jal WARP
5 print: add $a1,$v0,$0,      #$a0 = result from N!
6 li $v0,56
7 la $a0,Message
8 syscall
9 quit: li $v0,10          #ket thuc thuc hien
10 syscall
11 endmain:
12 WARP: sw $fp,-4($sp)    #save frame pointer
13 addi $fp,$sp,0 #new frame pointer point to the top
14 addi $sp,$sp,-8   #adjust stack pointer
15 sw $ra,0($sp)     #save return address
16
17 li $a0,3           #load test input N
18 jal FACT          #call fact procedure
19 nop
20
21 lw $ra,0($sp)      #restore return address
22 addi $sp,$fp,0 #return stack pointer
23 lw $fp,-4($sp)     #return frame pointer
24 jr $ra
25 wrap_end:
26 FACT: sw $fp,-4($sp)    #save frame pointer
27 addi $fp,$sp,0 #new frame pointer point to stack's top
28 addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack
29 sw $ra,4($sp)      #save return address
30 sw $a0,0($sp)      #save $a0 register
31 slti $t0,$a0,2      #if input argument N < 2
32 beq $t0,$zero,recursive #if it is false ((a0 = N) >=2)
33 nop
34 li $v0,1           #return the result N!=1
35 j done
36 nop
37 recursive: addi $a0,$a0,-1      #adjust input argument
38 jal FACT          #recursive call
39 nop
40 lw $v1,0($sp)      #load a0
41 mult $v1,$v0        #compute the result
42 mflo $v0
43 done: lw $ra,4($sp)    #restore return address
44 lw $a0,0($sp)      #restore a0
45 addi $sp,$fp,0 #restore stack pointer
46 lw $fp,-4($sp)     #restore frame pointer
47 jr $ra             #jump to calling
48 fact_end:
49
```

Kết quả chạy:



b, Khi không dùng \$fp:

Code:

```

Edit Execute
mips1.asm mips2.asm mips3.asm mips4.asm mips5.asm
1 .data
2     Message: .asciiz "Ket qua tinh giai thua la: "
3 .text
4 main:
5     jal WARP
6 print:
7     add $a1, $v0, $zero # $a0 = result from N!
8     li $v0, 56
9     la $a0, Message
10    syscall
11 quit:
12    li $v0, 10 #terminate
13    syscall
14 endmain:
15 WARP:
16     #sw $fp,-4($sp) #save frame pointer (1)
17     #addi $fp,$sp,0 #new frame pointer point to the top (2)
18     addi $sp,$sp,-8 #adjust stack pointer (3)
19     sw $ra,0($sp) #save return address (4)
20     li $a0,3 #load test input N
21     jal FACT #call fact procedure
22     nop
23     lw $ra,0($sp) #restore return address (5)
24     addi $sp,$fp,0 #return stack pointer (6)
25     lw $fp,-4($sp) #return frame pointer (7)
26     jr $ra
27 wrap_end:
28 FACT:
29     addi $sp,$sp,-8      #allocate space for $fp,$ra,$a0 in stack
30     sw $ra,4($sp)        #save return address
31     sw $a0,0($sp)        #save $a0 register
32
33     slti $t0,$a0,2        #if input argument N < 2
34     beq $t0,$zero,recursive #if it is false ((a0 = N) >=2)
35     nop
36     li $v0,1             #return the result N!=1
37     j done
38     nop
39 recursive:
40     addi $a0,$a0,-1       #adjust input argument
41     jal FACT            #recursive call
42     nop
43     lw $v1,0($sp)         #load a0
44     mult $v1,$v0           #compute the result
45     mflo $v0
46 done:
47     lw $ra,4($sp)         #restore return address
48     lw $a0,0($sp)         #restore a0
49     addi $sp,$sp,8          #restore frame pointer
50     jr $ra               #jump to calling
51 fact_end:

```

Kết quả chạy:

Name	Number	Value
\$zero	0	0x00000000
\$t0	1	0x10010000
\$v0	2	0x00000038
\$s0	3	0x00000035
\$a0	4	0x10100000
\$s1	5	0x00000006
\$s2	6	0x00000002
\$s3	7	0x00000000
\$s4	8	0x00000001
\$s5	9	0x00000000
\$s6	10	0x00000000
\$s7	11	0x00000000
\$s8	12	0x00000000
\$s9	13	0x00000000
\$s10	14	0x00000000
\$s11	15	0x00000000
\$s12	16	0x00000000
\$s13	17	0x00000000
\$s14	18	0x00000000
\$s15	19	0x00000000
\$s16	20	0x00000000
\$s17	21	0x00000000
\$s18	22	0x00000000
\$s19	23	0x00000000
\$s20	24	0x00000000
\$s21	25	0x00000000
\$s22	26	0x00000000
\$s23	27	0x00000000
\$s24	28	0x00000000
\$s25	29	0x00000000
\$s26	30	0x00000000
\$s27	31	0x00400004
\$pc		0x00000000
\$hi		0x00000000
\$lo		0x00000000

Bài 5:

Code:

```
1 .data
2 Message1:    .ascii "Largest: "
3 Message2:    .ascii "\nSmallest:"
4 Comma:        .ascii ","
5 .text
6 main: li $s0, 2          # Load input
7     li $s1, 3
8     li $s2, -1
9     li $s3, 4
10    li $s4, 9
11    li $s5, -2
12    li $s6, 8
13    li $s7, 5
14    jal init
15    nop
16    li $v0, 4
17    la $a0, Message1      #in ra message1
18    syscall
19    li $v0 , 1
20    add $a0,$t0,$zero
21    syscall      #in ra max value
22    li $v0, 4
23    la $a0, Comma       #in ra ","
24    syscall
25    li $v0 , 1
26    add $a0,$t5,$zero      # in ra max value's position
27    syscall
28    li $v0, 4
29    la $a0, Message2      #in ra message2
30    syscall
31    li $v0 ,1
32    add $a0,$t1,$zero
33    syscall      #in ra min value
34    li $v0 ,4
35    la $a0, Comma       #in ra ","
36    syscall
37    li $v0 , 1
38    add $a0,$t6,$zero
39    syscall      # in ra min value's position
40    li $v0, 10
41    syscall      # exit
42 endmain:

42 endmain:
43 Max.add $t0,$t3,$zero    # set Max = $t3
44 add $t5,$t2,$zero      # set i of max = $t2
45 jr $ra
46 Min.add $t1,$t3,$zero    # set Min = $t3
47 add $t6,$t2,$zero      # set i of min = $t2
48 jr $ra
49 init:   add $fp,$sp,$zero  # save address of origin sp
50 addi $sp,$sp, -32      # create space for stack
51 sw $s1, 0($sp)
52 sw $s2, 4($sp)
53 sw $s3, 8($sp)
54 sw $s4, 12($sp)
55 sw $s5, 16($sp)
56 sw $s6, 20($sp)
57 sw $s7, 24($sp)
58 sw $ra, 28($sp)        # save $ra for main
59 add $t0,$s0,$zero      # set Max = $s0
60 add $t1,$s0,$zero      # set Min = $s0
61 li $t5, 0              # set $t5 to 0
62 li $t6, 0              # set $t6 to 0
63 li $t2, 0              # set $t2 to 0 , i = 0
64 max_min: addi $sp,$sp,4
65 lw $t3,-4($sp)
66 sub $t4, $sp, $fp      # check if meet $ra
67 beq $t4,$zero, done # if true then done
68 addi $t2,$t2,1          # i++
69 sub $t4,$t0,$t3
70 bltzal $t4, Max      # if Max - $t3 < 0, swap Max
71 sub $t4,$t3,$t1
72 bltzal $t4, Min      # if $t3 - Min < 0 , swap Min
73 j max_min            # repeat
74 done:    lw $ra, -4($sp)      # load #$ra
75 jr $ra                # return
```

Line: 38 Column: 19 Show Line Numbers

Kết quả chạy:

The screenshot shows the assembly debugger interface with several panes:

- Registers:** Shows registers \$zero through \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, \$t32, \$t33, \$t34, \$t35, \$t36, \$t37, \$t38, \$t39, \$t40, \$t41, \$t42, \$t43, \$t44, \$t45, \$t46, \$t47, \$t48, \$t49, \$t50, \$t51, \$t52, \$t53, \$t54, \$t55, \$t56, \$t57, \$t58, \$t59, \$t60, \$t61, \$t62, \$t63, \$t64, \$t65, \$t66, \$t67, \$t68, \$t69, \$t70, \$t71, \$t72, \$t73, \$t74, \$t75, \$t76, \$t77, \$t78, \$t79, \$t80, \$t81, \$t82, \$t83, \$t84, \$t85, \$t86, \$t87, \$t88, \$t89, \$t90, \$t91, \$t92, \$t93, \$t94, \$t95, \$t96, \$t97, \$t98, \$t99, \$t9a, \$t9b, \$t9c, \$t9d, \$t9e, \$t9f, \$t9g, \$t9h, \$t9i, \$t9j, \$t9k, \$t9l.
- Text Segment:** Shows assembly code for the main routine, including syscalls for reading and writing strings, and arithmetic operations like addiu, add, and addi.
- Data Segment:** Shows memory dump from address 0x7fffffc0 to 0x7fffff00, mostly containing zeros.
- Labels:** Shows labels defined in the assembly code, such as main, endmain, max, min, done, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$t8, \$t9, \$t10, \$t11, \$t12, \$t13, \$t14, \$t15, \$t16, \$t17, \$t18, \$t19, \$t20, \$t21, \$t22, \$t23, \$t24, \$t25, \$t26, \$t27, \$t28, \$t29, \$t30, \$t31, \$t32, \$t33, \$t34, \$t35, \$t36, \$t37, \$t38, \$t39, \$t40, \$t41, \$t42, \$t43, \$t44, \$t45, \$t46, \$t47, \$t48, \$t49, \$t50, \$t51, \$t52, \$t53, \$t54, \$t55, \$t56, \$t57, \$t58, \$t59, \$t60, \$t61, \$t62, \$t63, \$t64, \$t65, \$t66, \$t67, \$t68, \$t69, \$t70, \$t71, \$t72, \$t73, \$t74, \$t75, \$t76, \$t77, \$t78, \$t79, \$t80, \$t81, \$t82, \$t83, \$t84, \$t85, \$t86, \$t87, \$t88, \$t89, \$t90, \$t91, \$t92, \$t93, \$t94, \$t95, \$t96, \$t97, \$t98, \$t99, \$t9a, \$t9b, \$t9c, \$t9d, \$t9e, \$t9f, \$t9g, \$t9h, \$t9i, \$t9j, \$t9k, \$t9l.
- Coproc 0:** Shows floating-point coprocessor values.