

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:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404fffd	addiu \$4,\$0,0xffffffff	2: main: li \$a0, -45 #load input parameter
	0x00400004	0xc1100000	jal 0x00400018	3: jal abs # jump and link to abs procedure
	0x00400008	0x00000000	nop	4: nop
	0x0040000c	0x00028020	add \$16,\$0,\$2	5: add \$s0, \$zero, \$v0
	0x00400010	0x24028000	addiu \$2,\$0,0x00000000	6: li \$v0, 10 #terminate
	0x00400014	0x00000000	syscall	7: syscall
	0x00400018	0x00041022	sub \$2,\$0,\$4	9: abs: sub \$v0, \$zero, \$a0 #if \$a0<0 then done
	0x0040001c	0x04000002	bltz \$4,0x00000002	10: bltz \$a0, done # if \$a0<0 then done
	0x00400020	0x00000000	nop	11: nop
	0x00400024	0x00801020	add \$2,\$4,\$0	12: add \$v0, \$a0, \$zero # else put \$a0 in v0
	0x00400028	0x03e00000	jr \$31	13: done: jr \$ra

Labels

Label	Address
main	0x00400000
endmain	0x00400018
abs	0x00400018
done	0x00400028

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

0x10010000 (Data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

Name

Number

Value

\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0xffffffff
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$s9	27	0x00000000
\$t0	28	0x00000000
\$t1	29	0x00000000
\$t2	30	0x00000000
\$t3	31	0x00000000
\$pc		0x00400018
\$hi		0x00000000
\$lo		0x00000000

Kết quả chạy:

Text Segment				Labels		Name		
Bkpt	Address	Code	Basic	Label	Address	Name	Number	Value
0	0x00400000	0x24000000	addiu \$t0,\$0,0x00000000	2: main: li \$t0, 2 #load test input		\$zero	0	0x00000000
1	0x00400004	0x24000000	addiu \$t1,\$0,0x00000006	3: li \$t1, 6		\$t0	1	0x00000000
2	0x00400008	0x24000000	addiu \$t2,\$0,0x00000009	4: li \$t2, 9		\$t1	2	0x00000000
3	0x0040000c	0x0c100007	jal 0x0040001c	5: jal max #call max procedure	0x0040001c	main	3	0x00000000
4	0x00400010	0x00000000	nop	6: nop	0x0040001c	endmain	4	0x00000000
5	0x00400014	0x24000000	addiu \$t2,\$0,0x0000000a	7: li \$t2, 10	0x0040001c	max	5	0x00000000
6	0x00400018	0x00000000	syscall	8: syscall	0x00400030	okay	6	0x00000000
7	0x0040001c	0x00000000	add \$t0,\$t0,\$zero # copy a0 in v0: largest so far	11: max: add \$v0,\$t0,\$zero # copy a0 in v0: largest so far	0x00400030	done	7	0x00000000
8	0x00400020	0x00000000	sub \$t0,\$t0,\$t1 # compute a1 - v0	12: sub \$t0,\$t1,\$t0 # compute a1 - v0			8	0x00000000
9	0x00400024	0x00000000	bltz \$t0,okay # if a1- v0 < 0 then no change	13: bltz \$t0,okay # if a1- v0 < 0 then no change			9	0x00000000
10	0x00400028	0x00000000	nop	14: nop			10	0x00000000
11	0x0040002c	0x00000000	add \$v0,\$t1,\$zero # else a1 is largest thus far	15: add \$v0,\$t1,\$zero # else a1 is largest thus far			11	0x00000000
12	0x00400030	0x00000000	sub \$t0,\$t2,\$t0 # compute a2 - v0	16: okay: sub \$t0,\$t2,\$t0 # compute a2 - v0			12	0x00000000
13	0x00400034	0x00000000	bltz \$t0,done # if a2 - v0 < 0 then no change	17: bltz \$t0,done # if a2 - v0 < 0 then no change			13	0x00000000
14	0x00400038	0x00000000	nop	18: nop			14	0x00000000
15	0x0040003c	0x00000000	add \$v0,\$t2,\$zero #else a2 is largest overall	19: done: jr \$rs # return to calling prog			15	0x00000000
16	0x00400040	0x00000000					16	0x00000000
17	0x00400044	0x00000000					17	0x00000000
18	0x00400048	0x00000000					18	0x00000000
19	0x0040004c	0x00000000					19	0x00000000
20	0x00400050	0x00000000					20	0x00000000
21	0x00400054	0x00000000					21	0x00000000
22	0x00400058	0x00000000					22	0x00000000
23	0x0040005c	0x00000000					23	0x00000000
24	0x00400060	0x00000000					24	0x00000000
25	0x00400064	0x00000000					25	0x00000000
26	0x00400068	0x00000000					26	0x00000000
27	0x0040006c	0x00000000					27	0x00000000
28	0x00400070	0x00000000					28	0x00000000
29	0x00400074	0x00000000					29	0x00000000
30	0x00400078	0x00000000					30	0x00000000
31	0x0040007c	0x00000000					31	0x00000000
32	0x00400080	0x00000000						
33	0x00400084	0x00000000						
34	0x00400088	0x00000000						
35	0x0040008c	0x00000000						
36	0x00400090	0x00000000						
37	0x00400094	0x00000000						
38	0x00400098	0x00000000						
39	0x0040009c	0x00000000						
40	0x004000a0	0x00000000						
41	0x004000a4	0x00000000						
42	0x004000a8	0x00000000						
43	0x004000ac	0x00000000						
44	0x004000b0	0x00000000						
45	0x004000b4	0x00000000						
46	0x004000b8	0x00000000						
47	0x004000bc	0x00000000						
48	0x004000c0	0x00000000						
49	0x004000c4	0x00000000						
50	0x004000c8	0x00000000						
51	0x004000cc	0x00000000						
52	0x004000d0	0x00000000						
53	0x004000d4	0x00000000						
54	0x004000d8	0x00000000						
55	0x004000dc	0x00000000						
56	0x004000e0	0x00000000						
57	0x004000e4	0x00000000						
58	0x004000e8	0x00000000						
59	0x004000ec	0x00000000						
60	0x004000f0	0x00000000						
61	0x004000f4	0x00000000						
62	0x004000f8	0x00000000						
63	0x004000fc	0x00000000						
64	0x00400100	0x00000000						
65	0x00400104	0x00000000						
66	0x00400108	0x00000000						
67	0x0040010c	0x00000000						
68	0x00400110	0x00000000						
69	0x00400114	0x00000000						
70	0x00400118	0x00000000						
71	0x0040011c	0x00000000						
72	0x00400120	0x00000000						
73	0x00400124	0x00000000						
74	0x00400128	0x00000000						
75	0x0040012c	0x00000000						
76	0x00400130	0x00000000						
77	0x00400134	0x00000000						
78	0x00400138	0x00000000						
79	0x0040013c	0x00000000						
80	0x00400140	0x00000000						
81	0x00400144	0x00000000						
82	0x00400148	0x00000000						
83	0x0040014c	0x00000000						
84	0x00400150	0x00000000						
85	0x00400154	0x00000000						
86	0x00400158	0x00000000						
87	0x0040015c	0x00000000						
88	0x00400160	0x00000000						
89	0x00400164	0x00000000						
90	0x00400168	0x00000000						
91	0x0040016c	0x00000000						
92	0x00400170	0x00000000						
93	0x00400174	0x00000000						
94	0x00400178	0x00000000						
95	0x0040017c	0x00000000						
96	0x00400180	0x00000000						
97	0x00400184	0x00000000						
98	0x00400188	0x00000000						
99	0x0040018c	0x00000000						
100	0x00400190	0x00000000						
101	0x00400194	0x00000000						
102	0x00400198	0x00000000						
103	0x0040019c	0x00000000						
104	0x004001a0	0x00000000						
105	0x004001a4	0x00000000						
106	0x004001a8	0x00000000						
107	0x004001ac	0x00000000						
108	0x004001b0	0x00000000						
109	0x004001b4	0x00000000						
110	0x004001b8	0x00000000						
111	0x004001bc	0x00000000						
112	0x004001c0	0x00000000						
113	0x004001c4	0x00000000						
114	0x004001c8	0x00000000						
115	0x004001cc	0x00000000						
116	0x004001d0	0x00000000						
117	0x004001d4	0x00000000						
118	0x004001d8	0x00000000						
119	0x004001dc	0x00000000						
120	0x004001e0	0x00000000						
121	0x004001e4	0x00000000						
122	0x004001e8	0x00000000						
123	0x004001ec	0x00000000						
124	0x004001f0	0x00000000						
125	0x004001f4	0x00000000						
126	0x004001f8	0x00000000						
127	0x004001fc	0x00000000						
128	0x00400200	0x00000000						
129	0x00400204	0x00000000						
130	0x00400208	0x00000000						
131	0x0040020c	0x00000000						
132	0x00400210	0x00000000						
133	0x00400214	0x00000000						
134	0x00400218	0x00000000						
135	0x0040021c	0x00000000						
136	0x00400220	0x00000000						
137	0x00400224	0x00000000						
138	0x00400228	0x00000000						
139	0x0040022c	0x00000000						
140	0x00400230	0x00000000						
141	0x00400234	0x00000000						
142	0x00400238	0x00000000						
143	0x0040023c	0x00000000						
144	0x00400240	0x00000000						
145	0x00400244	0x00000000						
146	0x00400248	0x00000000						
147	0x0040024c	0x00000000						
148	0x00400250	0x00000000						
149	0x00400254	0x00000000						
150	0x00400258	0x00000000						
151	0x0040025c	0x00000000						
152	0x00400260	0x00000000						
153	0x00400264	0x00000000						
154	0x00400268	0x00000000						
155	0x0040026c	0x00000000						
156	0x00400270	0x00000000						
157	0x00400274	0x00000000						
158	0x00400278	0x00000000						
159	0x0040027c	0x00000000						
160	0x00400280	0x00000000						
161	0x00400284	0x00000000						
162	0x00400288	0x00000000						
163	0x0040028c	0x00000000						
164	0x00400290	0x00000000						
165	0x00400294	0x00000000						
166	0x00400298	0x00000000						
167	0x0040029c	0x00000000						
168	0x004002a0	0x00000000						
169	0x004002a4	0x00000000						
170	0x004002a8	0x00000000						
171	0x004002ac	0x00000000						
172	0x004002b0							

Bài 4:

a, Khi dùng \$sp:

Code:

```
mips1.asm mips2.asm mips3.asm mips4.asm
1 .data
2 Message: .asciiz "Ket qua tinh giai thua la: "
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:

The screenshot displays a MIPS simulator interface with three main panels:

- Text Segment:** Shows the assembly code with addresses, codes, and comments. The code implements a recursive factorial function. The current instruction being executed is at address 0x00400028: `addi $a0,$a0,-1`.
- Labels:** A list of labels and their corresponding addresses, including `main`, `print`, `quit`, `endmain`, `WARP`, `wrap_end`, `FACT`, `recursive`, `done`, `fact_end`, and `Message`.
- Registers:** A table showing the values of registers. The `$zero` register is 0, `$t0` is 1, `$v0` is 56, and `$a0` is 4. Other registers like `$a1`, `$a2`, `$a3`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$t8`, `$t9`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$sp`, `$fp`, `$pc`, `$hi`, and `$lo` are also listed.

A dialog box is open in the center of the screen, displaying the text "Ket qua tinh giai thua la: 6" (The result of the calculation is: 6) and an "OK" button.

b, Khi không dùng \$fp:

Code:

```
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:

The screenshot shows the MARS MIPS simulator interface. The 'Text Segment' window displays the assembly code with line numbers 1 through 51. The 'Registers' window on the right shows the state of registers \$zero through \$t0, with \$a0 containing the value 6. The 'Data Segment' window at the bottom shows memory addresses and their corresponding values. A dialog box in the center displays the output message: 'Ket qua tinh giai thua la: 6'.

Bài 5:

Code:

```
1 .data
2 Message1: .asciiz "Largest: "
3 Message2: .asciiz "\nSmallest:"
4 Comma: .asciiz ","
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   # Max - $t3
70          bltzal $t4, Max   # if Max - $t3 < 0, swap Max
71          sub $t4,$t3,$t1   # $t3 - Min
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:

[illegible]