

Báo cáo thực hành KTMT tuần 7

Họ và tên: Đỗ Gia Huy

MSSV: 20215060

Assignment 1

Code:

#Laboratory Exercise 7 Home Assignment 1

.text

main:

```
li    $a0, -45    #load input parameter
jal   abs         #jump and link to abs procedure
nop
add   $s0, $zero, $v0
li    $v0, 10     #terminate
syscall
```

endmain:

#-----

function abs

param[in] \$a0 the interger need to be gained the absolute value

return \$v0 absolute value

#-----

```
abs:  sub   $v0,$zero,$a0    #put -(a0) in v0; in case (a0)<0
      bltz  $a0,done         #if (a0)<0 then done
      nop
      add   $v0,$a0,$zero    #else put (a0) in v0
```

done:

```
jr    $ra
```

Thực thi:

- Trước khi chạy lệnh: jal abs

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404ff43	addiu \$4,\$0,-45	3: main: li \$a0, -45 #load input parameter
	0x00400004	0x0c100006	jal 0x00400018	4: jal abs #jump and l...
	0x00400008	0x00000000	nop	5: nop
	0x0040000c	0x00028020	add \$16,\$0,\$2	6: add \$a0, \$zero, \$v0
	0x00400010	0x2402000a	addiu \$2,\$0,10	7: li \$v0, 10 #terminate
	0x00400014	0x0000000c	syscall	8: syscall
	0x00400018	0x00041022	sub \$2,\$0,\$4	15: abs: sub \$v0,\$zero,\$a0 #put -(a0) ...
	0x0040001c	0x04800002	bltz \$4,2	16: bltz \$a0,done #if (a0)<0 ...
	0x00400020	0x00000000	nop	17: nop
	0x00400024	0x00801020	add \$2,\$4,\$0	18: add \$v0,\$a0,\$zero #else put (...

Labels

Label	Address
HomeAssignment1.asm	
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	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0

Mars Messages

Run IO

Clear

Reset: reset completed.

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	-45
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194308
hi		0
lo		0

- Sau khi chạy lệnh: jal abs

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404ff43	addiu \$4,\$0,-45	3: main: li \$a0, -45 #load input parameter
	0x00400004	0x0c100006	jal 0x00400018	4: jal abs #jump and l...
	0x00400008	0x00000000	nop	5: nop
	0x0040000c	0x00028020	add \$16,\$0,\$2	6: add \$a0, \$zero, \$v0
	0x00400010	0x2402000a	addiu \$2,\$0,10	7: li \$v0, 10 #terminate
	0x00400014	0x0000000c	syscall	8: syscall
	0x00400018	0x00041022	sub \$2,\$0,\$4	15: abs: sub \$v0,\$zero,\$a0 #put -(a0) ...
	0x0040001c	0x04800002	bltz \$4,2	16: bltz \$a0,done #if (a0)<0 ...
	0x00400020	0x00000000	nop	17: nop
	0x00400024	0x00801020	add \$2,\$4,\$0	18: add \$v0,\$a0,\$zero #else put (...

Labels

Label	Address
HomeAssignment1.asm	
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	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0

Mars Messages

Run IO

Clear

Reset: reset completed.

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	-45
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	4194312
pc		4194328
hi		0
lo		0

Result:

- Kết quả sau khi thực thi xong toàn bộ chương trình:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays assembly code with addresses and source code. The 'Labels' window shows the 'HomeAssignment1.asm' file with labels like 'main', 'endmain', 'abs', and 'done'. The 'Data Segment' window shows memory addresses and values. The 'Mars Messages' window shows the message 'program is finished running --'.

- **Nhận xét:**
 - Khi chạy lệnh `jal abs` (địa chỉ lệnh `0x00400004`) thì thanh ghi `$ra` được gán bằng địa chỉ của câu lệnh tiếp theo (`0x00400008`) và thanh ghi `pc` được gán bằng địa chỉ `0x00400018` (địa chỉ tại nhãn `abs`)
 - Kết quả cuối cùng ta lấy được giá trị tuyệt đối của số được nạp vào trong thanh ghi `$a0` và ghi kết quả đó vào thanh ghi `$s0`

<code>\$a0</code>	4	-45
<code>\$s0</code>	16	45

Assignment 2

Code:

#Laboratory Exercise 7, Home Assignment 2

.text

main:

```

li    $a0, 2      #load test input
li    $a1, 6
li    $a2, 9
jal   max         #call max procedure
nop
add   $s0, $v0, $zero
li    $v0, 10     #terminate
syscall

```

endmain:

#-----

#Procedure max: find the largest of three integers

#param[in] \$a0 integers

#param[in] \$a1 integers

#param[in] \$a2 integers

#return \$v0 the largest value

#-----

```
max: add  $v0,$a0,$zero    #copy (a0) in v0; largest so far
      sub  $t0,$a1,$v0     #compute (a1)-(v0)
      bltz $t0,okay        #if (a1)-(v0)<0 then no change
      nop
      add  $v0,$a1,$zero    #else (a1) is largest thus far
okay: sub  $t0,$a2,$v0     #compute (a2)-(v0)
      bltz $t0,done        #if (a2)-(v0)<0 then no change
      nop
      add  $v0,$a2,$zero    #else (a2) is largest overall
done: jr   $ra             #return to calling program
```

Result:

- Kết quả chạy chương trình mẫu:

The screenshot displays the MARS MIPS simulator interface. The main window shows assembly code with columns for Address, Code, Basic, and Source. The code implements a 'max' procedure to find the largest of three integers. Below the code, the 'Data Segment' is shown with memory addresses and values. To the right, the 'Labels' window lists labels like 'main', 'endmain', 'max', 'okay', and 'done' with their corresponding addresses. On the far right, a table shows the state of registers (Name, Number, Value). The 'Mars Messages' window at the bottom indicates that the program is finished running.

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	10
\$v1	3	0
\$a0	4	2
\$a1	5	6
\$a2	6	9
\$a3	7	0
\$t0	8	3
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268469224
\$sp	29	2147479544
\$fp	30	0
\$ra	31	4194320
pc		4194336
hi		0
lo		0

Thanh ghi \$s0 = 9 là thanh ghi chứa giá trị lớn nhất

- Sau khi sửa giá trị các thanh ghi:

\$a0 = 1, \$a1 = 7, \$a2 = -5

Text Segment					Labels				
Bkpt	Address	Code	Basic	Source	Label	Address	name	number	value
	0x00400000	0x24040001	addiu \$t4,\$s0,1	3: main: li \$a0, 1 #load test input			\$zero	0	0
	0x00400004	0x24050007	addiu \$t5,\$s0,7	4: li \$a1, 7			\$at	1	0
	0x00400008	0x2406ffff	addiu \$t6,\$s0,-5	5: li \$a2,-5			\$v0	2	10
	0x0040000c	0x0c100008	jal 0x00400020	6: jal max #call max p...	main	0x00400000	\$v1	3	0
	0x00400010	0x00000000	nop	7: nop	endmain	0x00400020	\$a0	4	1
	0x00400014	0x00400020	add \$t6,\$t2,\$s0	8: add \$s0, \$v0, \$zero	max	0x00400020	\$a1	5	7
	0x00400018	0x2402000a	addiu \$t2,\$s0,10	9: li \$v0, 10 #terminate	okay	0x00400034	\$a2	6	-5
	0x0040001c	0x0000000c	syscall	10: syscall	done	0x00400044	\$t0	8	-12
	0x00400020	0x00801020	add \$t2,\$t4,\$s0	19: max: add \$v0,\$a0,\$zero #copy (\$a0) ...			\$t1	9	0
	0x00400024	0x00a24022	sub \$t5,\$t5,\$t2	20: sub \$t0,\$t1,\$v0 #compute (\$a...			\$t2	10	0
							\$t3	11	0
							\$t4	12	0
							\$t5	13	0
							\$t6	14	0
							\$t7	15	0
							\$t8	16	7
							\$t9	17	0
							\$a0	18	0
							\$a1	19	0
							\$a2	20	0
							\$a3	21	0
							\$a4	22	0
							\$a5	23	0
							\$a6	24	0
							\$a7	25	0
							\$a8	26	0
							\$a9	27	0
							\$sp	28	268468224
							\$fp	29	2147479548
							\$tp	30	0
							\$ra	31	4194320
							pc		4194336
							hi		0
							lo		0

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0

Mars Messages		Run IO
Clear -- program is finished running --		

Kết quả thu được: \$s0 = 7

- Nhận xét:
 - Khi chạy lệnh jal thì thanh ghi \$ra được gán bằng giá trị của địa chỉ của câu lệnh tiếp theo sau jal trong nhãn main. Thanh ghi pc được gán bằng địa chỉ của nhãn max để câu lệnh tiếp tục được thực hiện bắt đầu từ nhãn max. Sau khi chạy đến jr \$ra thì pc được gán bằng địa chỉ trong \$ra (địa chỉ của nop)

Assignment 3

Code:

#Laboratory Exercise 7, Home Assignment 3

.text

li \$s0, 7

li \$s1, -3

push:

addi \$sp,\$sp,-8 #adjust the stack pointer

sw \$s0,4(\$sp) #push \$s0 to stack

sw \$s1,0(\$sp) #push \$s1 to stack

work:

nop

nop

nop

pop: lw \$s0,0(\$sp) #pop from stack to \$s0

lw \$s1,4(\$sp) #pop from stack to \$s1

addi \$sp,\$sp,8 #adjust the stack pointer

Result:

- Trước khi chạy lệnh addi ở nhãn push:

The screenshot shows the Mars MIPS simulator interface. The Text Segment window displays the following instructions:

Bkpt	Address	Code	Basic	Source
	0x24100007	addiu \$t6,\$0,7	3:	li \$t6, 7
	0x24100008	addiu \$t1,\$0,-3	4:	li \$t1, -3
	0x24100008	addi \$sp,\$sp,-8	5:	push: addi \$sp,\$sp,-8 #adjust the stack p...
	0x24100009	sw \$t6,4(\$sp)	6:	sw \$t6,4(\$sp) #push \$t6 to stack
	0x2410000a	sw \$t1,0(\$sp)	7:	sw \$t1,0(\$sp) #push \$t1 to stack
	0x2410000b	nop	8:	work: nop
	0x2410000c	nop	9:	nop
	0x2410000d	nop	10:	nop
	0x2410000e	lw \$t6,0(\$sp)	11:	pop: lw \$t6,0(\$sp) #pop from stack to \$t6
	0x2410000f	lw \$t1,4(\$sp)	12:	lw \$t1,4(\$sp) #pop from stack to \$t1

The Data Segment window shows memory addresses from 0x10010000 to 0x1001000f. The Labels window shows the 'push' instruction at 0x24100008. The Memory window shows the current state of memory, with \$sp at 0x7ffffc00.

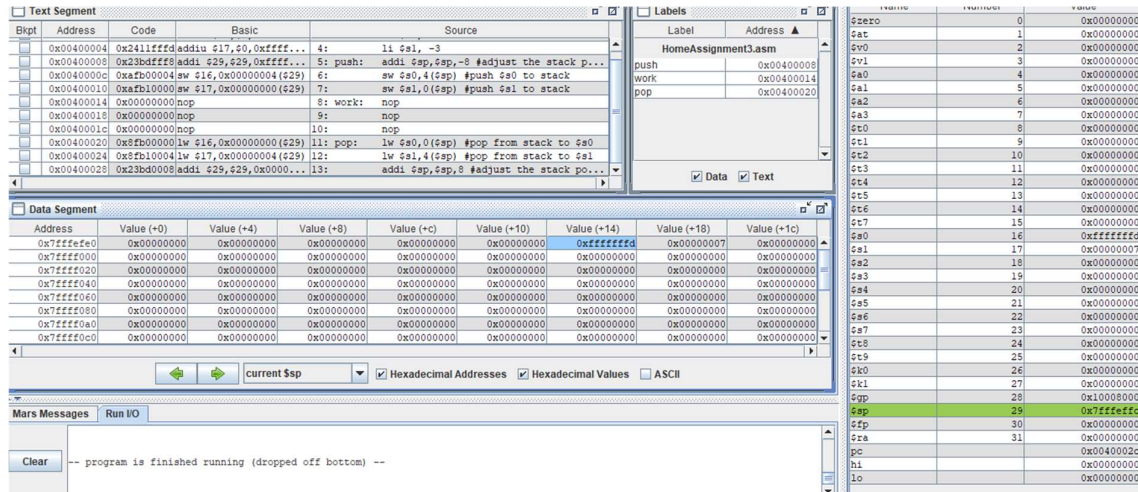
\$sp = 0x7ffffc00

- Sau khi chạy lệnh addi ở nhãn push:

The screenshot shows the Mars MIPS simulator interface after the 'push' instruction has been executed. The Text Segment window displays the same instructions as before. The Data Segment window shows memory addresses from 0x10010000 to 0x1001000f. The Labels window shows the 'push' instruction at 0x24100008. The Memory window shows the current state of memory, with \$sp at 0x7ffffc08.

\$sp = 0x7ffffc08

- Thanh ghi \$sp được giảm đi 8 byte (tức là có sự cấp phát cho bộ nhớ stack 8 byte)
- Sau đó lần lượt ghi giá trị trong \$s0 vào \$sp + 4, giá trị trong \$s1 vào \$sp + 0
- Sau khi chạy lệnh `addi` ở nhấn `pop`:



- Thực hiện đổi chỗ hai số bằng cách load giá trị tại địa chỉ \$sp + 0 vào \$s1, load giá trị tại địa chỉ \$sp + 4 vào \$s0
- Lệnh `add $sp, $sp, 8` (giúp giải phóng stack, trả lại đỉnh stack)

Assignment 4

Code tính giai thừa có sử dụng thanh ghi \$fp:

#Laboratory Exercise 7, Home Assignment 4

.data

Message: .asciiz "Ket qua tinh giai thua la: "

.text

main: jal WARP

print: add \$a1, \$v0, \$zero # \$a0 = result from N!

li \$v0, 56

la \$a0, Message

syscall

quit: li \$v0, 10 #terminate

syscall

endmain:

#-----

#Procedure WARP: assign value and call FACT

#-----

WARP:

```
sw    $fp,-4($sp) #save frame pointer (1)
addi  $fp,$sp,0 #new frame pointer point to the top (2)
addi  $sp,$sp,-8 #adjust stack pointer (3)
sw    $ra,0($sp) #save return address (4)
li    $a0,7 #load test input N
jal   FACT #call fact procedure
nop

lw    $ra,0($sp) #restore return address (5)
addi  $sp,$fp,0 #return stack pointer (6)
lw    $fp,-4($sp) #return frame pointer (7)
jr    $ra
```

wrap_end:

#-----

#Procedure FACT: compute N!

#param[in] \$a0 integer N

#return \$v0 the largest value

#-----

FACT:

```
sw    $fp,-4($sp) #save frame pointer
addi  $fp,$sp,0 #new frame pointer point to stack's top
addi  $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack
sw    $ra,4($sp) #save return address
```



```

sw    $a0,0($sp) #save $a0 register

slti   $t0,$a0,2 #if input argument N < 2
beq    $t0,$zero,recursive#if it is false ((a0 = N) >=2)
nop
li     $v0,1 #return the result N!=1
j      done
nop
recursive:
addi   $a0,$a0,-1 #adjust input argument
jal    FACT #recursive call
nop
lw     $v1,0($sp) #load a0
mult   $v1,$v0 #compute the result
mflo   $v0
done:
lw     $ra,4($sp) #restore return address
lw     $a0,0($sp) #restore a0
addi   $sp,$fp,0 #restore stack pointer
lw     $fp,-4($sp) #restore frame pointer
jr     $ra #jump to calling
fact_end:

```

Thực thi, kết quả và nhận xét:

Với $n = 7$, tức là $\$a0 = 7$:

.text

```
main: li    $v0, 51
      la    $a0, message1      # Hien message1 & Nhap so N
      syscall
      bltz  $a0, main          # Kiem tra N khong am
      nop

      jal   fact
      nop

      add   $a1, $v0, $zero     # $a1 = ket qua
      li    $v0, 56
      la    $a0, message2      # Hien message2 va ket qua
      syscall

      li    $v0, 10            # Exit
      syscall

fact:  addi  $sp, $sp, -8 # Cap phat Stack
      sw    $ra, ($sp)
      sw    $s0, 4($sp)

      li    $v0, 1             # v0 = 1
      beq   $a0, $zero, endfact # N = 0 branch to the endfact
      nop
      add   $s0, $a0, $zero     # s0 = N
      addi  $a0, $a0, -1 # N = N - 1
      jal   fact
```

```

        nop
        mul   $v0, $v0, $s0      #  $N! = N * (N-1) * (N-2) * \dots * 1$ 
endfact: lw    $ra, ($sp)
        lw    $s0, 4($sp)
        addi  $sp, $sp, 8        # Giai phong Stack
        jr    $ra

```

Assignment 5

Code:

Laboratory Exercise 7, Assignment 5

.data

```

largest: .asciiz "Largest: "
smallest: .asciiz "\nSmallest: "
comma: .asciiz ", "

```

.text

```

main: li     $s0, 5
      li     $s1, -12
      li     $s2, 56
      li     $s3, 12
      li     $s4, 87
      li     $s5, -2
      li     $s6, -343
      li     $s7, 23

```

```

jal    saveNumbers

```

```

nop

```

```

li     $v0, 4      # Print message Largest
la     $a0, largest

```

syscall

add \$a0, \$t0, \$zero # Print Max

li \$v0, 1

syscall

li \$v0, 4 # Print message Comma

la \$a0, comma

syscall

add \$a0, \$t5, \$zero

li \$v0, 1 # Print the register number of Max

syscall

li \$v0, 4 # Print message Smallest

la \$a0, smallest

syscall

add \$a0, \$t1, \$zero # Print Min

li \$v0, 1

syscall

li \$v0, 4 # Print message Comma

la \$a0, comma

syscall

add \$a0, \$t6, \$zero

li \$v0, 1 # Print the register number of Min

```

        syscall
endmain:  li    $v0, 10          # Exit
        syscall

#-----
# Return $t0 = Max
# Return $t1 = Min
# Index of Max = $t5
# Index of Min = $t6
#return $v0 the largest value
#-----

swapMax:
    add    $t0,$t3,$zero
    add    $t5,$t2,$zero
    jr     $ra

swapMin:
    add    $t1,$t3,$zero
    add    $t6,$t2,$zero
    jr     $ra

saveNumbers:
    add    $t9,$sp,$zero      # Save address of origin $sp
    addi   $sp,$sp, -32
    sw     $s1, 0($sp)
    sw     $s2, 4($sp)
    sw     $s3, 8($sp)
    sw     $s4, 12($sp)
    sw     $s5, 16($sp)
    sw     $s6, 20($sp)
    sw     $s7, 24($sp)

```



```

sw    $ra, 28($sp)          # Save $ra for main
add   $t0,$s0,$zero         # Max = $s0
add   $t1,$s0,$zero         # Min = $s0
li    $t5, 0                # Index of Max to 0
li    $t6, 0                # Index of Min to 0
li    $t2, 0                # i = 0

```

findMaxMin:

```

addi  $sp,$sp,4
lw    $t3,-4($sp)
sub    $t4, $sp, $t9
beq    $t4,$zero, done      # If $sp = $fp branch to the 'done'
nop
addi  $t2,$t2,1             # i++
sub    $t4,$t0,$t3
bltzal $t4, swapMax         # If $t3 > Max branch to the swapMax
nop
sub    $t4,$t3,$t1
bltzal $t4, swapMin         # If $t3 < Min branch to the swapMin
nop
j      findMaxMin           # Repeat

```

done:

```

lw    $ra, -4($sp)
jr    $ra                  # Return to calling program

```

Result:

- Kết quả các thanh ghi sau khi thực hiện xong chương trình trên:

