Thực hành Kiến trúc máy tính tuần 3

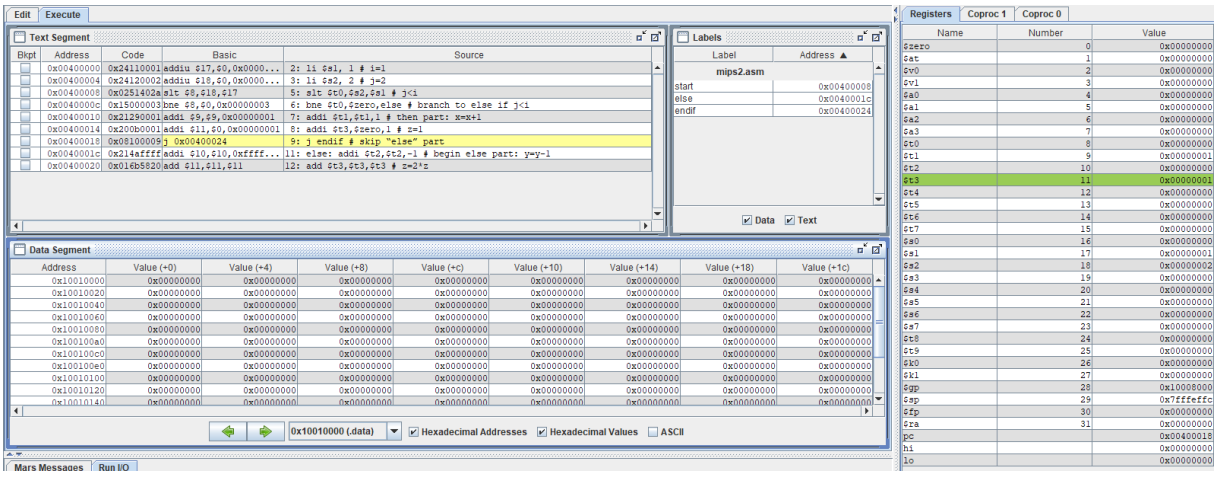
Họ tên: Đỗ Hoàng Minh Hiếu

MSSV: 20225837

Bài 1

Trường hợp 1: i < j (Gán i = 1, j = 2)

```
1  .text
2  li $s1, 1 # i=1
3  li $s2, 2 # j=2
4  start:
5  slt $t0,$s2,$s1 # j<i
6  bne $t0,$zero,else # branch to else if j<i
7  addi $t1,$t1,1 # then part: x=x+1
8  addi $t3,$zero,1 # z=1
9  j endif # skip “else” part
10
11 else: addi $t2,$t2,-1 # begin else part: y=y-1
12 add $t3,$t3,$t3 # z=2*z
13
14 endif:
```



Lệnh slt kiểm tra $s2<$s1. Trong trường hợp 1 này điều kiện sai nên sẽ gán thanh ghi $t0 giá trị 0.

Lệnh bne kiểm tra $t0 khác 0 hay không. Vì $t0 bằng giá trị thanh ghi $zero nên sẽ không nhảy đến câu lệnh ở nhãn else.

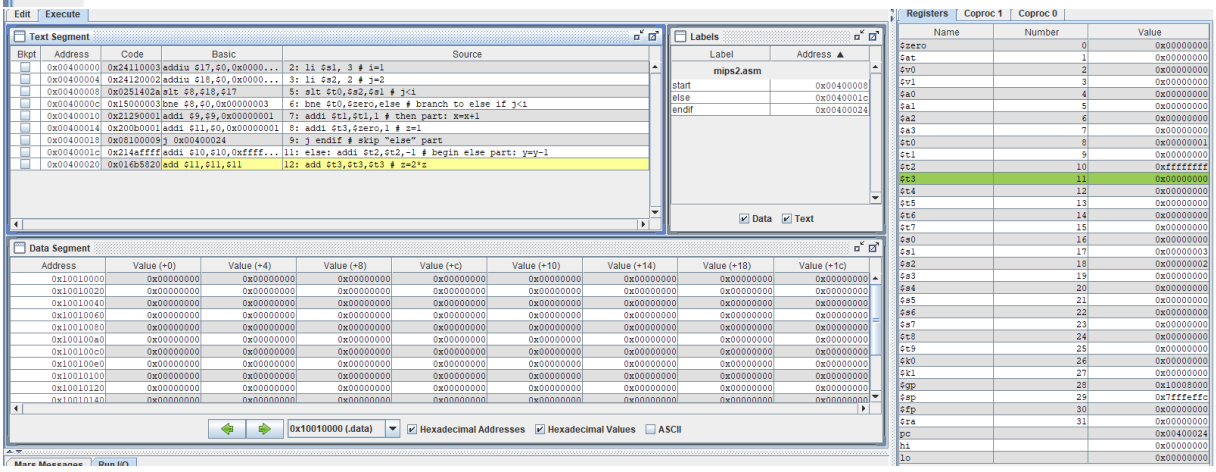
Lệnh addi $t1,$t1,1 cộng giá trị ở $t1 thêm 1 và lưu lại vào $t1. Vì $t1 bằng 0 nên sẽ gán giá trị thanh $t1 bằng 0x00000001.

Lệnh addi $t3,$zero,1 gán 1 vào thanh ghi $t3 nên giá trị thanh $t3 bằng 0x00000001.

j endif sẽ nhảy đến nhãn endif và kết thúc lệnh if.

Trường hợp 2: i >= j (Gán i = 3, j = 2)

```
1  .text
2  li $s1, 3 # i=1
3  li $s2, 2 # j=2
4  start:
5  slt $t0,$s2,$s1 # j<i
6  bne $t0,$zero,else # branch to else if j<i
7  addi $t1,$t1,1 # then part: x=x+1
8  addi $t3,$zero,1 # z=1
9  j endif # skip "else" part
10
11 else: addi $t2,$t2,-1 # begin else part: y=y-1
12 add $t3,$t3,$t3 # z=2*z
13
14 endif:
```



Lệnh slt kiểm tra $s2<$s1. Trong trường hợp 1 này điều kiện đúng nên sẽ gán thanh ghi $t0 giá trị 1.

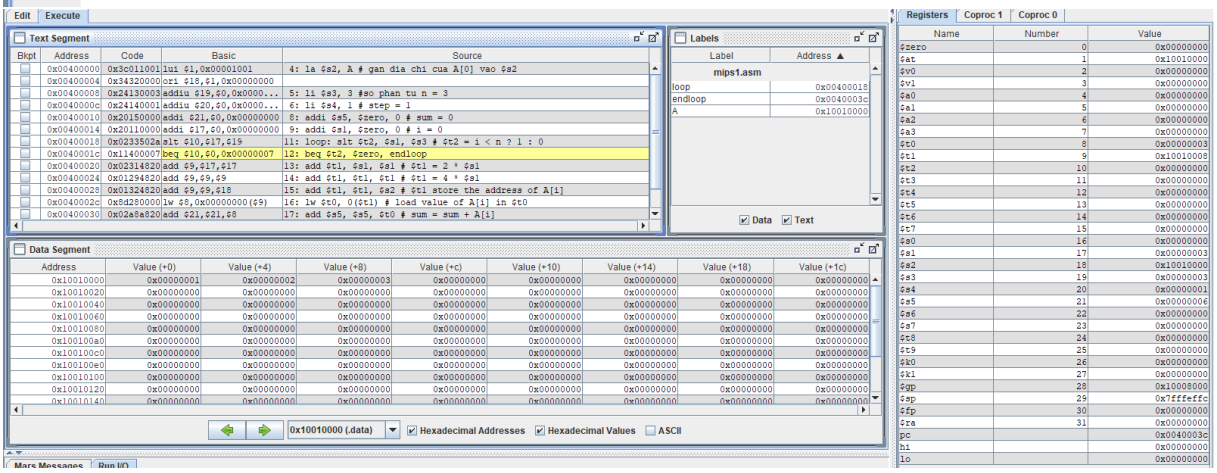
Lệnh bne kiểm tra $t0 khác 0 hay không. Vì $t0 bằng 1, khác giá trị thanh ghi $zero nên sẽ nhảy đến câu lệnh ở nhãn else.

Lệnh addi $t2,$t2,-1 sẽ cộng giá trị thanh ghi $t2 và -1. Do giá trị $t2 bằng 0 nên sau khi kết thúc câu lệnh, giá trị $t2 bằng -1(0xffffffff).

Lệnh add $t3,$t3,$t3 cộng giá trị $t3 và $t3 rồi lưu lại vào $t3. Do $t3 bằng 0 nên giá trị $t3 không thay đổi.

Bài 2

```
1  .data
2  A: .word 1,2,3,4,5,6,7,8,9
3  .text
4  la $s2, A # gan dia chi cua A[0] vao $s2
5  li $s3, 9 #so phan tu n = 9
6  li $s4, 1 # step = 1
7
8  addi $s5, $zero, 0 # sum = 0
9  addi $s1, $zero, 0 # i = 0
10
11 loop: slt $t2, $s1, $s3 # $t2 = i < n ? 1 : 0
12 beq $t2, $zero, endloop
13 add $t1, $s1, $s1 # $t1 = 2 * $s1
14 add $t1, $t1, $t1 # $t1 = 4 * $s1
15 add $t1, $t1, $s2 # $t1 store the address of A[i]
16 lw $t0, 0($t1) # load value of A[i] in $t0
17 add $s5, $s5, $t0 # sum = sum + A[i]
18 add $s1, $s1, $s4 # i = i + step
19 j loop # goto loop
20 endloop:
21
```



Bước khai báo:

Lệnh la $s2, A: Gán địa chỉ của A[0] vào $s2.

$s2: chứa địa chỉ của A[0].

Lệnh li $s3, 3: Gán giá trị tổng số phần tử 9 vào $s3.

Lệnh li $s4, 1: Gán giá trị bước nhảy 1 vào $s4.

Lệnh addi $s5, $zero, 0: Gán giá trị tổng ban đầu 0 vào $s5.

Lệnh addi $s1, $zero, 0: Gán giá trị 0 vào $s1 (biến lặp).

Bước vòng lặp:

Vòng lặp sẽ được thực thi cho đến khi giá trị của $s1 không còn nhỏ hơn 3.

Trong mỗi vòng lặp, chúng ta sẽ thực hiện các bước sau:

Tính toán địa chỉ của phần tử hiện tại trong mảng A và load giá trị của nó vào $t0.

Cộng giá trị của phần tử này vào tổng ($s5).

Tăng giá trị của biến lặp $s1 lên 1.

Sau khi vòng lặp kết thúc, giá trị của tổng được lưu trong $s5 (0x0000002d). Mà tổng từ 1 đến 9 bằng 45, đúng với kết quả thực thi chương trình.

Bài 3:

la $s0, test: Gán địa chỉ biến test vào $s0

lw $s1, 0($s0): Lấy dữ liệu biến test và lưu vào $s1

li $t0, 0: Gán $t0 bằng 0

li $t1, 1: Gán giá trị $t1 bằng 1

li $t2, 2: Gán giá trị $s2 bằng 2

Code:

mips1.asm | mips2.asm

```
1  .data
2  test: .word 1
3  .text
4   la $s0, test #load the address of test variable
5   lw $s1, 0($s0) #load the value of test to register $t1
6   li $t0, 0 #load value for test case
7   li $t1, 1
8   li $t2, 2
9   beq $s1, $t0, case_0
10  beq $s1, $t1, case_1
11  beq $s1, $t2, case_2
12  j default
13 case_0: addi $s2, $s2, 1 #a=a+1
14  j continue
15 case_1: sub $s2, $s2, $t1 #a=a-1
16  j continue
17 case_2: add $s3, $s3, $s3 #b=2*b
18  j continue
19 default:
20 continue:
```

Với trường hợp test = 0:

Lệnh beq $s1, $t0, case_0 sẽ so sánh giá trị $s1 bằng $t0 hay không, do $t0 và giá trị biến test bằng nhau nên sẽ thực hiện nhảy đến case_0.

case_0: addi $s2, $s2, 1: Cộng giá trị $s2 thêm 1 và lưu lại vào $s2 nên $s2 sẽ lưu giá trị 1.

j continue: Nhảy đến continue.

Với trường hợp test = 1:

Lệnh beq $s1, $t0, case_0 sẽ so sánh giá trị $s1 bằng $t0 hay không, do $t0 khác giá trị biến test nên sẽ bỏ qua.

Lệnh beq $s1, $t1, case_1 sẽ so sánh giá trị $s1 bằng $t1 hay không, do $t1 bằng giá trị biến test nên sẽ sẽ thực hiện nhảy đến case_1.

case_1: sub $s2, $s2, $t1: Trừ giá trị $s2 1 đơn vị và lưu lại vào $s2 nên $s2 sẽ lưu giá trị -1(0xffffffff).

Với trường hợp test = 2:

Lệnh beq $s1, $t0, case_0 sẽ so sánh giá trị $s1 bằng $t0 hay không, do $t0 khác giá trị biến test nên sẽ bỏ qua.

Lệnh beq $s1, $t1, case_1 sẽ so sánh giá trị $s1 bằng $t0 hay không, do $t1 khác giá trị biến test nên sẽ bỏ qua.

Lệnh beq $s1, $t2, case_2 sẽ so sánh giá trị $s1 bằng $t2 hay không, do $t2 bằng giá trị biến test nên sẽ sẽ thực hiện nhảy đến case_2.

case_2: add $s3, $s3, $s3: Cộng giá trị $s3 và $s3 và lưu lại vào $s3 nên $s3 sẽ không thay đổi (Vẫn bằng 0).

Bài 4:

Câu a: i < j

```
.text
li $s1, 3  # i=3
li $s2, 4     # j=4
start:
   slt $t0, $s1, $s2      # Check if i < j
   beq $t0, $zero, else    # Branch to else if i >= j
   addi $t1, $t1, 1       # Then part: x = x + 1
   addi $t3, $zero, 1     # z = 1
   j endif              # Skip "else" part
else:
   addi $t2, $t2, -1      # Begin else part: y = y - 1
   add $t3, $t3, $t3      # z = 2 * z
```

```
endif:
```

Câu b: i>=j

```
.text
li $s1, 3     # i=3
li $s2, 4     # j=4
start:
   slt $t0, $s1, $s2          # Check if i < j
   bne $t0, $zero, else    # Branch to else if i < j
   addi $t1,$t1,1     # then part: x=x+1
   addi $t3,$zero,1     # z=1
   j endif # skip “else” part
else: addi $t2,$t2,-1       # begin else part: y=y-1
    add $t3,$t3,$t3 # z=2*z
endif:
```

Câu c: i+j <= 0

```
.text
li $s1, -10 # i=-10
li $s2, 4 # j=4
start:
   add $t0, $s1, $s2      # Calculate i + j
   bgtz $t0, else         # Branch to else if i + j > 0
   addi $t1, $t1, 1       # Then part: x = x + 1
   addi $t3, $zero, 1     # z = 1
   j endif              # Skip "else" part
else:
   addi $t2, $t2, -1      # Begin else part: y = y - 1
   add $t3, $t3, $t3      # z = 2 * z
endif:
```

Câu d: i+j > m+n (m and n stored in other registers)

```
.text
li $s1, 10 # i=10
```

```
li $s2, 4 # j=4
li $s3, 5 # m=5
li $s4, 7 #n=7
start:
   add $t0, $s1, $s2       # Calculate i + j
   add $t4, $s3, $s4       # Calculate m + n
   slt $t5, $t0, $t4       # t0 < t4? 1:0
   bne $t5, $zero, else    # Branch to else if i + j <= m + n
   addi $t1, $t1, 1        # Then part: x = x + 1
   addi $t3, $zero, 1      # z = 1
   j endif                 # Skip "else" part
else:
   addi $t2, $t2, -1       # Begin else part: y = y - 1
   add $t3, $t3, $t3       # z = 2 * z
endif:
```

Bài 5:

Câu a: i<=n

```
.data
A: .word 1,2,3,4,5,6,7,8,9,10
.text
 addi $s3, $zero, 10 #n=10
 addi $s4, $zero, 1 #step=1
 la $s2, A #load address of A[0]
 addi $s5, $zero, 0 #sum = 0
 addi $s1, $zero, 0 #i = 0
loop:
 slt $t2, $s3, $s1 # $t2 = n < i? 1 : 0
 bne $t2, $zero, endloop
 add $t1,$s1,$s1 #$t1=2*$s1
 add $t1,$t1,$t1 #$t1=4*$s1
 add $t1,$t1,$s2 #$t1 store the address of A[i]
```

```
 lw $t0,0($t1) #load value of A[i] in$t0
 add $s5,$s5,$t0 #sum=sum+A[i]
 add $s1,$s1,$s4 #i=i+step
 j loop #goto loop
endloop:
```

Câu b: sum>=0

```
.data
A: .word 1,2,3,-10,5,6,7,-8,9,-5
.text
li $s3, 10       #n=10
li $s4, 1        #step=1
la $s2,A         # address A[0]
addi $s5, $zero, 0      #sum = 0
addi $s1, $zero, 0      #i = 0
loop:
    bltz $s5, endloop   # endloop if sum < 0
    add $t1,$s1,$s1     #$t1=2*$s1
    add $t1,$t1,$t1      #$t1=4*$s1
    add $t1,$t1,$s2     #$t1 store the address of A[i]
    lw $t0,0($t1)     #load value of A[i]
    add $s5,$s5,$t0     #sum=sum+A[i]
    add $s1,$s1,$s4     #i=i+step
    j loop    #goto loop
endloop:
```

Câu c: A[i] != 0

```
.data
A: .word 1,3,3,0,5,6,7,-4,9,10
.text
li $s3, 10       #n=10
li $s4, 1        #step=1
la $s2,A         # address A[0]
```

```
addi $s5, $zero, 0      #sum = 0
addi $s1, $zero, 0      #i = 0
loop:
    add $t1,$s1,$s1         #$t1=2*$s1
    add $t1,$t1,$t1     #$t1=4*$s1
    add $t1,$t1,$s2         #$t1 store the address of A[i]
    lw $t0,0($t1)     #load value of A[i] in $t0
    beqz $t0, endloop       # endloop if A[i] == 0
    add $s5,$s5,$t0         #sum=sum+A[i]
    add $s1,$s1,$s4         #i=i+step
    j loop         #goto loop
endloop:
```

Bài 6:

```
.data
A: .word 1,2,3,4,5,6,-7,3,-5,1
.text
    li $s3, 10   #n=10
    li $s4, 1         #step=1
    la $s2, A   #address of A[0]
    li $s5, 0         #gt = 0
    li $s6, 0        #address = 0
    li $s1, 0    #i = 0
loop:
    slt $t2, $s1, $s3 # $t2 = i < n? 1 : 0
    beq $t2, $zero, endloop
    sll $t1, $s1,2     #t1=s1*4
    add $t1,$t1,$s2     #$t1 store the address of A[i]
    lw $t0,0($t1)    #load value of A[i] in$t0
    slt $t4, $t0, $zero      #check abs
    beq $t4, $zero, positive
    sub $t0,$zero,$t0
```

```
positive:
    slt $t5, $s5, $t0      #$s5 chua gia tri max
    beq $t5, $zero, negative        #Neu |a[i]| <= max thi nhay qua nhan “sai”
    add $s5,$zero,$t0     #max = |a[i||
    add $s6,$zero,$s1     #Cap nhat vi tri moi
    j cont
negative:
    addi $s5, $s5, 0
    addi $s6, $s6, 0
    j cont
cont:
    add $s1,$s1,$s4 #i=i+step
    j loop #goto loop
endloop:
```

TEST:

TH1: 1,2,3,4,5,6,-7,3,-5,1

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010024
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000000a
$s2	18	0x10010000
$s3	19	0x0000000a
$s4	20	0x00000001
$s5	21	0x00000007
$s6	22	0x00000006
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400064
hi		0x00000000
lo		0x00000000

TH2: 1,2,4,5,6,8,9,-10,5,6

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000006
$t1	9	0x10010024
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000000a
$s2	18	0x10010000
$s3	19	0x0000000a
$s4	20	0x00000001
$s5	21	0x0000000a
$s6	22	0x00000007
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400064
hi		0x00000000
lo		0x00000000

TH3: 4,9,8,2,4,7,3,-45,21,10

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000000a
$t1	9	0x10010024
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000000a
$s2	18	0x10010000
$s3	19	0x0000000a
$s4	20	0x00000001
$s5	21	0x0000002d
$s6	22	0x00000007
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400064
hi		0x00000000
lo		0x00000000