BÀI THỰC HÀNH TUẦN 3
KIẾN TRÚC MÁY TÍNH

Họ và tên: Đinh Huy Dương

MSSV: 20215020

Bài 1:

```
if (i<=j)
     x=x+1;
     z=1;
else
     y=y-1;
     z=2*z;
```

Gán giá trị của

- i tại thanh ghi $s1
- j tại thanh ghi $s2
- x tại thanh ghi $t1
- y tại thanh ghi $t2
- z tại thanh ghi $t3

```
start:
        addi    $s2, $zero, 0x00001
        addi    $s1, $zero, 0x00002
        slt     $t0,$s2,$s1       # j<i
        bne     $t0,$zero,else    # branch to else if j<i
        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:
```

TRƯỜNG HỢP 1:

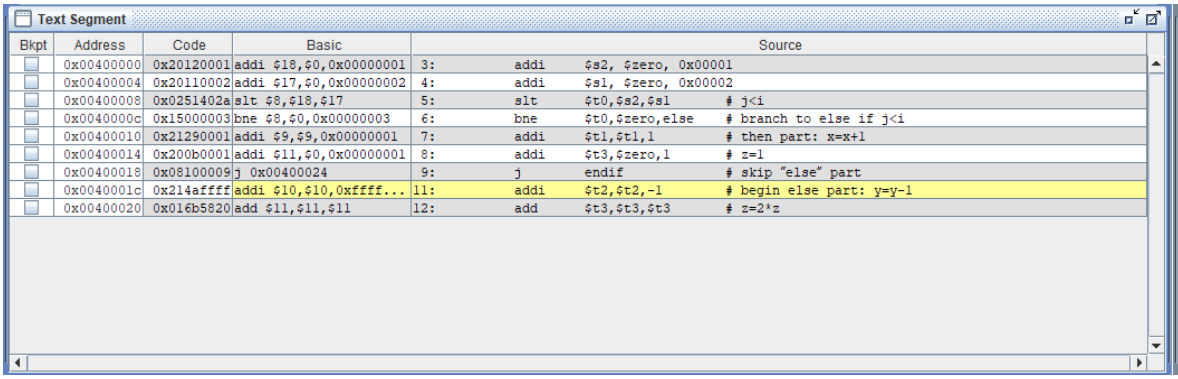
$s1 = i = 2

$s2 = j = 1

Tại trường hợp này i>j nên sẽ chuyển sẽ có được kết quả của phần else.

Máy sẽ đầu tiên sẽ so sánh $s2 và $s1 qua lệnh “slt” (nếu $2 nhỏ hơn $1 thì gán biến bằng 1) và gán vào thanh ghi $t0. Lệnh tiếp theo là lệnh rẽ nhánh qua nhãn “else” nếu $t0 khác 0, trong trường hợp này $t1 =1 nên sẽ rẽ nhánh tới nhãn “else”.

Quan sát thanh ghi $pc mang giá trị lần lượt theo trình tự của các lệnh trong file, chỉ tới khi phần rẽ nhánh thì $pc mang giá trị của địa chỉ của phần “else” (tại 0x004001c) và tiếp tục đến địa chỉ 0x0040020 và kết thúc ở “endif” tại 0x0040024



Trong phần lệnh “else” giá trị của y (tại $t2) sẽ bằng 0-1 = -1 =0xffffffff và giá trị của z (tại $t3) bằng 0 = 2*0 = 0x00000000

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x00000000
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000002
$s2	18	0x00000001
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400024
hi		0x00000000
lo		0x00000000

TRƯỜNG HỢP 2:

$s1 = i = 1

$s2 = j = 2

Tại trường hợp này máy sẽ không nhảy đến phần “else” mà tiếp tục cho đến khi cuối cùng gặp lệnh “j endif” để nhảy đến hết lệnh điều kiện. Đến cuối thanh ghi $pc cũng nhảy đến địa chỉ 0x004000024

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20120002	addi $18,$0,0x00000002	3: addi $s2, $zero, 0x00002
	0x00400004	0x20110001	addi $17,$0,0x00000001	4: addi $s1, $zero, 0x00001
	0x00400008	0x0251402a	slt $8,$18,$17	5: slt $t0,$s2,$s1 # j>i
	0x0040000c	0x15000003	bne $8,$0,0x00000003	6: bne $t0,$zero,else # branch to else if j<i
	0x00400010	0x21290001	addi $9,$9,0x00000001	7: addi $t1,$t1,1 # then part: x=x+1
	0x00400014	0x200b0001	addi $11,$0,0x00000001	8: addi $t3,$zero,1 # z=1
	0x00400018	0x08100009	j 0x00400024	9: j endif # skip "else" part
	0x0040001c	0x214affff	addi $10,$10,0xffff...	11: addi $t2,$t2,-1 # begin else part: y=y-1
	0x00400020	0x016b5820	add $11,$11,$11	12: add $t3,$t3,$t3 # z=2*z

Labels

Label	Address ▲
Week2_HomeAssignment	
start	0x00400000
else	0x0040001c
endif	0x00400024

Data Text

Khi đó, sau lệnh “bne”, chương trình sẽ không rẽ nhánh qua “else” tiếp tục đến 2 lệnh addi, thay đổi giá trị: x (tại $t1) = 0+1 = 1 và z =1 (tại $t3)

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000001
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000001
$s2	18	0x00000002
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400024
hi		0x00000000
lo		0x00000000

Bài 2:

With C language:

```
sum = 0;
for (int i = 0; i < n; i += step)
    sum += A[i];
```

With assembly language:

```
        sum = 0
        i = 0
loop:   if (i >= n) goto endloop
        sum = sum + A[i]
        i = i + step
        goto loop
endloop:
```

Gán: Giá trị biến chạy, i: $s1

Địa chỉ phần tử đầu của mảng A: $s2

Số các phần tử của mảng, n: $s3

Bước của biến chạy i, step: $s4

Tổng của các phần tử trong mảng A, sum: $s5

Đặt các giá trị của các tham số quan trọng vừa được nêu trên. Đặc biệt, lệnh "la" sử dụng để gán địa chỉ của mảng A trong Data Segment

```
16  .data
17          A:      .word 2,5,1,8,0
18  .text
19          addi    $s1, $zero, 0       # i = 0
20          la      $s2, A              # Store the Address of A into $s2
21          addi    $s3, $zero, 5       # n = 5
22          addi    $s4, $zero, 1       # step = 1
23          addi    $s5, $zero, 0       # sum = 0
24  loop:
25          slt     $t2, $s1, $s3       # $t2 = i < n ? 1 : 0
26          beq     $t2, $zero,endloop
27          add     $t1, $s1, $s1       # $t1 = 2 * $s1
28          add     $t1, $t1, $t1       # $t1 = 4 * $s1
29          add     $t1, $t1, $s2       # $t1 store the address of A[i]
30          lw      $t0, 0($t1)         # load value of A[i] in $t0
31          add     $s5, $s5, $t0       # sum = sum + A[i]
32          add     $s1, $s1, $s4       # i = i + step
33          j       loop                # goto loop
34  endloop:
```

Khởi tạo giá trị của mảng A với 5 phần tử kiểu word:{2,5,1,8,0} tại Data Segment. Ta có thể thấy ở hàng đầu tại địa chỉ 0x10010000, 5 giá trị đã được khởi tạo. Mỗi ô nhớ cách nhau 4 byte (+4)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000002	0x00000005	0x00000001	0x00000008	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

Sau đó, chương trình sẽ thực hiện nhãn "loop". Bước đầu tiên là thực hiện lệnh "slt" để so sánh xem $s1 (biến chạy i) có bé hơn $s3 (số

phần tử của mảng n). Lệnh rẽ nhánh beq để kết thúc vòng lặp khi i>n -> $t2 =0 -> nhảy đến endloop. Trong trường hợp này, i<n, gán giá trị cho thanh $t2 =1 và tiếp tục chạy.

2 lệnh add tiếp theo dùng để gán $t1 = 4*i =4*$s1 với mục đích dùng để sang các ô tiếp theo trong vùng Data Segment (mỗi ô cách nhau 4 byte). Tiếp tục cộng thêm địa chỉ phần tử đầu của mảng A ($s2) vào $t1 để có thể xác định được địa chỉ ô nhớ chính xác của phần tử trong mảng với vị trí i tương ứng.

Khi địa chỉ của phần từ thứ i đã được lưu vào $t1, sử dụng thanh ghi $t0 để lưu GIÁ TRỊ của phần tử i qua địa chỉ tại $t1 bằng lệnh "lw". Từ đó có thể tính tổng sum ($s5) cộng với thanh $t0, và tăng bước chạy của i (step - $s4), trong trường hợp này, step =1, giá trị i sẽ tăng lần lượt từ 0-1-2-3-4.

Kết thúc lệnh lặp, sử dụng lệnh "j" để nhảy về đầu hàm loop cho lần lặp tiếp theo. Lệnh "j" làm cho giá trị $pc thay đổi từ địa chỉ của "loop" tới kết thúc vòng lặp tại "endloop".

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	0x00000000
$t1	9	0x10010010
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000005
$s2	18	0x10010000
$s3	19	0x00000005
$s4	20	0x00000001
$s5	21	0x00000010
$s6	22	0x00000000
$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		0x0040003c
hi		0x00000000
lo		0x00000000

Kết quả của tổng sum tại $s5 = 2+5+1+8+0=16=0x0000 0010

Bài 3:

```
switch(test) {
      case 0:
            a=a+1; break;
      case 1:
            a=a-1; break;
      case 2:
            b=2*b; break;
      }
```

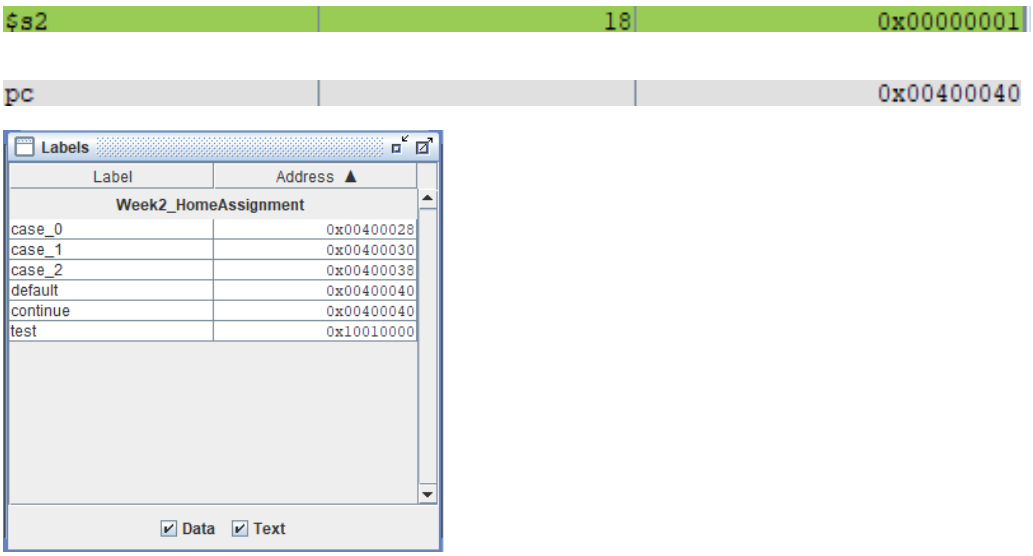
```
.data
        test: .word 0     #The choice variable

.text
        la $s0, test      #load the address of test variable
        lw $s1, 0($s0)    #load the value of test to register $t1
        li $t0, 0         #load value for test case
        li $t1, 1
        li $t2, 2
        beq $s1, $t0, case_0
        beq $s1, $t1, case_1
        beq $s1, $t2, case_2
        j default
case_0: addi $s2, $s2, 1   #a=a+1
        j continue
case_1: sub $s2, $s2, $t1  #a=a-1
        j continue
case_2: add $s3, $s3, $s3  #b=2*b
        j continue
default:
continue:
```

Đặt a tại $s2, b tại $s3

Khai báo biến test dưới kiểu word. Đây sẽ là biến biểu thị lựa chọn vào các case của chương trình. Khi đó ta sẽ lưu địa chỉ và giá trị của biến test vào thanh ghi qua 2 lệnh “la” và “lw” qua lần lượt thanh $s0 và $s1.

Khi đó ta gán các giá trị của các trường hợp của các giá trị test: 0,1,2 vào trong 3 thanh ghi $t0, $t1, $t2. Khi đó 3 lệnh so sánh và rẽ nhánh “beq” tiếp theo sẽ so sánh và dịch chuyển về các case đã có sẵn. Trường hợp test không bằng bất cứ case nào, chương trình sẽ chạy qua 3 lệnh “beq” và đi thẳng vào lệnh “j” nhảy đến default và kết thúc chương trình ở nhãn “continue”.

Cụ thể, trong trường hợp như trong hình, test = 0, chương trình sẽ nhảy tới case_0, thay đổi giá trị a (tại $s2) = a+1 =1, và kết thúc ở “continue” với $pc mang địa chỉ “continue”



Trong trường hợp test = 1, chương trình nhảy tới case_1, thay đổi $s2 thành a-1 = -1 = 0xffffffff qua lệnh “sub” để trừ. Tương tự với test =2, giá trị của $s3 sẽ được nhân đôi.

Bất cứ trường hợp test nào khác 3 trường hợp trên sẽ nhảy xuống “default” và kết thúc ở “continue”.

Bài 4: Mặc định x=y=z=0

a, i<j

```
start:
      addi  $s2, $zero, 0x00001
      addi  $s1, $zero, 0x00003
      slt   $t0, $s1, $s2     # i<j -> $st0 ?1:0
      beq   $t0, $zero, else  # branch to “else if j>=i
      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=z*2
endif:
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000003
$s2	18	0x00000001
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400024
hi		0x00000000
lo		0x00000000

Trường hợp này i>j nên bị rẽ nhánh sang “else”, thay đổi y=$t2= 0xffffffff, z=$t3= 0

b, i>=j

```
start:
        addi    $s2, $zero, 0x00001
        addi    $s1, $zero, 0x00003
        slt     $t0, $s1, $s2       # i<j -> $st0 ?1:0
        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=z*2
```

```
endif:
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000001
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000003
$s2	18	0x00000001
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400024
hi		0x00000000
lo		0x00000000

Trong trường hợp này, i>j, nên sẽ thay đổi thanh ghi của x ($t1) và z($t3) =1

c, i+j<=0

```
start:
      addi  $s2, $zero, 0x00001
      addi  $s1, $zero, 0x00003
      add   $t0, $s1, $s2     # $t0= i+j
      slt   $t5, $zero, $t0   # i+j >0 -> $t5 ? 1:0
      bne   $t5, $zero, 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=z*2
endif:
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000004
$t1	9	0x00000000
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000001
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000003
$s2	18	0x00000001
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400028
hi		0x00000000
lo		0x00000000

Trường hợp này có i+j ($t5) > 0 nên rẽ nhánh sang nhãn “else”

khiến cho thay đổi thanh ghi của y và z, y=y-1=0-1=-1=0xffffffff

z=z*2 = 0*2=0

d, i+j > m+n

Đặt giá trị của m và n lần lượt tại các thanh ghi $s3 và $s4

```
start:
      addi     $s1, $zero, 0x00000000  # i = ?
      addi     $s2, $zero, 0x00000001  # j = ?
      addi     $s3, $zero, 0x00000001  # m = ?
      addi     $s4, $zero, 0x00000000  # n = ?
      add   $t0,$s1,$s2       # $t0 = i+j
      add   $t4,$s3,$s4       # $t4 = m+n
      slt   $t5, $t4, $t0     # i+j >m+n -> $t5 ? 1:0
      beq   $t5,$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:
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x00000000
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000001
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000001
$s3	19	0x00000001
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$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		0x00400034
hi		0x00000000
lo		0x00000000

Trường hợp này có x+y = m+n nên thanh ghi $t2, $t3 được thay đổi thành:

y=y-1 =0-1 =-1 =0xffffffff

z=z*2 = 0*2 = 0x000000000

Bài 5:

a, i<n: Giống với Bài 2

b, i<=n

```
.data
        A:      .word 2,5,1,8,0
.text
        addi    $s1, $zero, 0       # i = 0
        la      $s2, A              # Store the Address of A into $s2
```

```
        addi  $s3, $zero, 5     # n = 5
        addi  $s4, $zero, 1     # step = 1
        addi  $s5, $zero, 0     # sum = 0
loop:
        slt   $t2, $s3, $s1     # $t2 = i > n ? 0 : 1
        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:
```

Registers | Coproc 1 | Coproc 0

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	0x00000000
$t1	9	0x10010014
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000006
$s2	18	0x10010000
$s3	19	0x00000005
$s4	20	0x00000001
$s5	21	0x00000010
$s6	22	0x00000000
$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		0x0040003c
hi		0x00000000
lo		0x00000000

Tương tự với Bài 2, tuy nhiên giá trị của i tại thanh $s1 =6 vì điều kiện để thoát vòng lặp là i>n

c, sum >=0

```
.data
```

```
      A:    .word 2,5,1,-9,0
.text
      addi  $s1, $zero, 0     # i = 0
      la    $s2, A            # Store the Address of A into $s2
      addi  $s3, $zero, 5     # n = 5
      addi  $s4, $zero, 1     # step = 1
      addi  $s5, $zero, 0     # sum = 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
      add   $s5, $s5, $t0     # sum = sum + A[i]
      bltz  $s5, endloop
      add   $s1, $s1, $s4     # i = i + step
      j     loop              # goto loop
endloop:
```

Registers | Coproc 1 | Coproc 0

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	0xfffffff7
$t1	9	0x1001000c
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000003
$s2	18	0x10010000
$s3	19	0x00000005
$s4	20	0x00000001
$s5	21	0xffffffff
$s6	22	0x00000000
$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		0x00400038
hi		0x00000000
lo		0x00000000

Trường hợp này, tại a[4]=-9, sum =-1, nên chương trình rẽ nhánh tới endloop, thanh ghi sum tại $s5 = 0xffffffff và thanh ghi của i tại $1 = 0x3 và thanh ghi tạm thời $t0 chứa giá trị cuối cùng của mảng trước khi kết thúc lặp: -9=0xfffffff7

d, A[i] = 0

```
.data
	A:	.word 2,5,1,8,0,4
.text
	addi	$s1, $zero, 0		# i = 0
	la	$s2, A			# Store the Address of A into $s2
	addi	$s3, $zero, 6		# n = 6
	addi	$s4, $zero, 1		# step = 1
	addi	$s5, $zero, 0		# sum = 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
        beq   $t0, $zero, endloop
        add   $s5, $s5, $t0     # sum = sum + A[i]
        add   $s1, $s1, $s4     # i = i + step
        j     loop              # goto loop
endloop:
```

Registers | Coproc 1 | Coproc 0

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	0x00000000
$t1	9	0x10010010
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000004
$s2	18	0x10010000
$s3	19	0x00000005
$s4	20	0x00000001
$s5	21	0x00000010
$s6	22	0x00000000
$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		0x00400038
hi		0x00000000
lo		0x00000000

Trong trường hợp này gặp a[4] =0 nên chương trình rẽ sang endloop và dừng lại không tính thêm a[5] =4 vào tổng nên giá trị sum ở thanh $s5 = 0x10 =16

Bài 6:

```
.data
        arr: .word 1,2,-4,5,-7
.text
        addi  $s1, $zero, 0     # i = 0
        la    $s2, arr          # Store the Address of arr into $s2
        addi  $s3, $zero, 5     # n = 5
```

```
      addi  $s4, $zero, 1     # step = 1
      addi  $s5, $zero, 0     # max = 0
      j     loop
absolute:
      sub   $t0, $zero, $t0   # |n| = 0 - n, n<0
      j     compare
loop:
      slt   $t2, $s1, $s3     # $t2 = i < n ? 1 : 0
      beq   $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 arr[i]
      lw    $t0, 0($t1)       # load value of arr[i] in $t0
      add   $t4, $zero, $t0   # the original value of arr[i] is stored in $t4
      bltz  $t0, absolute     # if arr[i] <0 then branch to absolute
compare:
      slt   $t3, $s5, $t0     # arr[i] > max -> $t3 ? 1:0
      bne   $t3, $zero, maximize
      add   $s1, $s1, $s4     # i = i + step
      j     loop              # goto loop
maximize:
      add   $s5, $zero, $t0   # max = arr[i]
      add   $s6, $zero, $t4   # the value of max
      add   $s7, $zero, $s1   # the index of max in the array
      add   $s1, $s1, $s4     # i = i + step
      j     loop              # goto loop
endloop:
```

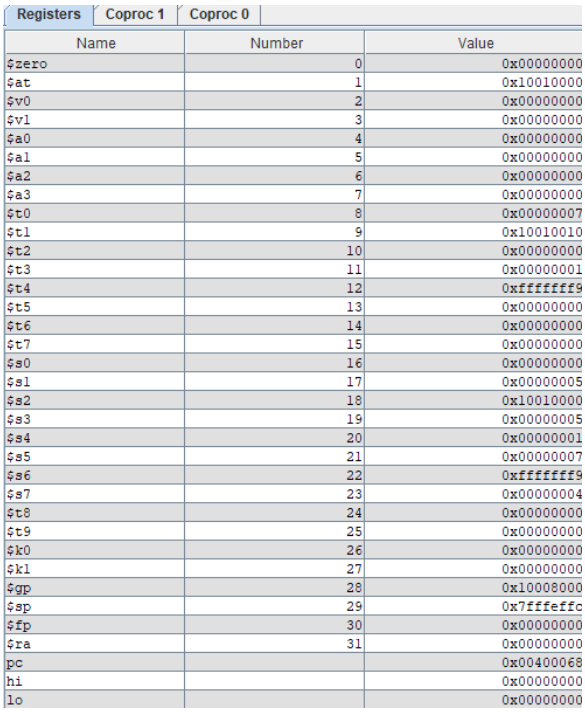
Giá trị của biến max để so sánh được lưu vào thanh $s5

Giá trị của phần tử có trị tuyệt đối lớn nhất được lưu vào thanh $s6

Giá trị của vị trí phần tử đấy trong mảng được lưu vào thanh $s7

Mảng arr = {1,2,-4,5,-7}

Kết quả:



Registers | Coproc 1 | Coproc 0

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	0x00000007
$t1	9	0x10010010
$t2	10	0x00000000
$t3	11	0x00000001
$t4	12	0xfffffff9
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000005
$s2	18	0x10010000
$s3	19	0x00000005
$s4	20	0x00000001
$s5	21	0x00000007
$s6	22	0xfffffff9
$s7	23	0x00000004
$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		0x00400068
hi		0x00000000
lo		0x00000000

Tại $s5, giá trị max =7 = |-7|

$s6 = 0xfffffff9 = -7

$s7 = 4 là vị trí của -7 trong mảng arr (arr[4])