BÁO CÁO THỨC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 2

Họ và tên: Vũ Đức Hoàng Anh

MSSV: 20235658

1. Assignment 1

- **Nhập chương trình :**

```
# Laboratory Exercise 3, Home Assignment 1
.data
        X: .word 2     # Biến X, kiểu word (4 bytes), giá trị khởi tạo = 2
        Y: .word 3     # Biến Y, kiểu word (4 bytes), giá trị khởi tạo = 3
        Z: .word 4     # Biến Z, kiểu word (4 bytes), giá trị khởi tạo = 4
.text
start:
li s1, 4                # Khởi tạo giá trị 4 cho s1
li s2, 5                # Khỏi tạo giá trị 5 cho s2

la t4, X                # Lấy địa chỉ của X trong vùng nhớ
lw t1, 0(t4)            # Gán giá trị của X vào t1
lw t2, 4(t4)            # Gán giá trị của Y vào t2
lw t3, 8(t4)            # Gán giá trị của Z vào t3

blt s2, s1, else        # if j < i then jump else
then:
        addi  t1, t1, 1         # x=x+1
        addi  t3, zero, 1       # z=1
        j  endif                # Bước nhảy đến endif
else:
        addi  t2, t2, -1    # y=y-1
        add   t3, t3, t3    # z=2*z
endif:
```

- **Các biến được khởi tạo:**

  + Biến t1 tương ứng với X (Trong ví dụ là 2)

  + Biến t2 tương ứng với Y (Trong ví dụ là 3)

  + Biến t3 tương ứng với Z (Trong ví dụ là 4)

- **Các bước thực hiện của chương trình :**

+ Câu lệnh *X: .word 2* để khởi tạo giá trị 2 cho X
+ Câu lệnh *Y: .word 3* để khởi tạo giá trị 3 cho Y
+ Câu lệnh *Z: .word 4* để khởi tạo giá trị 4 cho Z
+ Câu lệnh *li s1, 4* để khởi tạo giá trị cho s1 tương ứng với i
+ Câu lệnh *li s2, 5* để khởi tạo giá trị cho s2 tương ứng với j
+ Câu lệnh *la t4, X* để lấy địa chỉ của X lưu vào biến t4
+ Câu lệnh *lw t1, 0(t4)* gán giá trị của X vào t1
+ Câu lệnh *lw t2, 4(t4)* gán giá trị của Y vào t2
+ Câu lệnh *lw t3, 8(t4)* gán giá trị của Z vào t3
+ Câu lệnh *blt s2, s1, else* là câu lệnh điều kiện nếu s2 nhỏ hơn s1 thì nhảy đến else, khi điều kiện sai thực hiện tiếp tục các câu lệnh
+ Câu lệnh *addi t1, t1, 1* tương ứng với x = x + 1 (hoạt động khi j > i)
+ Câu lệnh *addi t3, zero, 1* tương ứng với z = 1 (hoạt động khi j > i)
+ Câu lệnh *j endif* để nhảy đến thẻ endif.
+ Câu lệnh addi *t2, t2, -1* tương ứng với y = y – 1 (hoạt động khi j < i)
+ Câu lệnh add *t3, t3, t3* tương ứng với z = z + z (hoạt đọng khi j < i)

- **Quan sát kết quả trên cửa sổ Register:**

+ Với giá trị của x = 2, y = 3, z =4, i = 4, j = 5 **(giá trị của j > i)** chương trình hoạt động với kết quả là :

Giá trị của s1 = 4 (đúng với chạy thử)

Giá trị của s2 = 5 (đúng với chạy thử)

Giá trị của t4 = 0x10010000 là địa chỉ của lưu giá trị của X

Giá trị của t1, t2, t3 lần lượt là 2, 3, 4 đúng với giá trị đã gán ban đầu

+ Với s1 = 4, s2 = 5 mà 4 < 5 nên điều kiện của câu lệnh *blt s2, s1, else* là không thỏa mãn lên chương trình tiếp tục thực hiện lệnh của thẻ *then*

Giá trị của t1 = t1 + 1 = 3 (x = x + 1)

Giá trị của t3 = 0 +1 = 1 (z = 1)

⇨ Đúng với kết quả chạy thử

Trước khi thực hiện if elsse	Sau khi thực hiện if else

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000004
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000003
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400038

+ Với s1 = 6, s2 = 5 mà 6 > 5 nên điều kiện của câu lệnh *blt s2, s1, else* là thỏa mãn lên chương trình tiếp tục thực hiện lệnh của thẻ *else*

Giá trị của t2 = t2 + (-1) = 2 (y = y – 1)

Giá trị của t3 = 4 + 4 = 8 (z = z + z)

⇨ Đúng với kết quả chạy thử

Với các thanh ghi đều có giá trị khởi đầu 0x00000000

Thanh ghi pc bắt đầu từ 0x00400000 và tăng thêm 4 giá trị với mỗi lệnh

Trước khi thực hiện if elsse	Sau khi thực hiện if else

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000006
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000004
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000002
s0	8	0x00000000
s1	9	0x00000006
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000008
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400038

2. Assignment 2

- **Nhập chương trình :**

```
 # Laboratory 3, Home Assigment 2
.data
A: .word 1, 2, 3, 4, 5
# khởi tạo mảng có 5 phần tử
.text
        li s1, 0        # gía trị bắt đầu duyệt
        la s2, A        # Lưu địa chỉ của A vào s2
        li s3, 5        # khởi tạo giá trị s3 là số phần tử
        li s4, 1        # khởi tạo bước nhảy
        li s5, 0        # Khởi tạo giá trị ban đầu cho tổng
loop:
        bge s1, s3, endloop         # Điều kiện dùng của vòng lặp
        add t1, s1, s1          # t1 = 2 * s1
        add t1, t1, t1          # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2          # t1 store the address of A[i]
        lw  t0, 0(t1)           # lấy giá trị của từng phần tử trong A
```

```
        add s5, s5, t0              # tính tổng từng phần tử trong A
        add s1, s1, s4              # tăng bước nhảy
        j  loop                 # quay lại loop để tiếp tục lặp
endloop:
```

- **Các bước thực hiện của chương trình :**

  + Câu lệnh *A: .word 1, 2, 3 ,4 ,5* để khởi tạo mảng có 5 phần tử lần lượt là 1, 2, 3, 4, 5 của A

  + Câu lệnh *li s1, 0* để khởi tạo biến đếm cho chương trình (i = 0)

  + Câu lệnh *la s2, A* lưu địa chỉ của A vào biến s2

  + Câu lệnh *li s3, 5* để khởi tạo giá trị s3 là lưu số lượng phần tử của mảng A

  + Câu lệnh *li s4, 1* để khởi tạo bước nhảy cho vòng lặp

  + Câu lệnh *li s5, 0* để khởi tạo giá trị ban đầu cho tổng

  + Câu lệnh *bge s1, s3, endloop* dùng để tạo điều kiện dừng cho vòng lặp. Vơi s1 lớn hơn hoặc bằng s2 thì chuyển tiếp đến thẻ endloop

  + Câu lệnh *add t1, s1, s1* và câu lệnh add t1, t1, t1 để khởi tạo giá trị cho t1 là 4*i . Tác dụng để lưu bước nhảy tiếp theo.

  + Câu lệnh *add t1, t1, s2* để lấy địa chỉ của phần tử A[i]

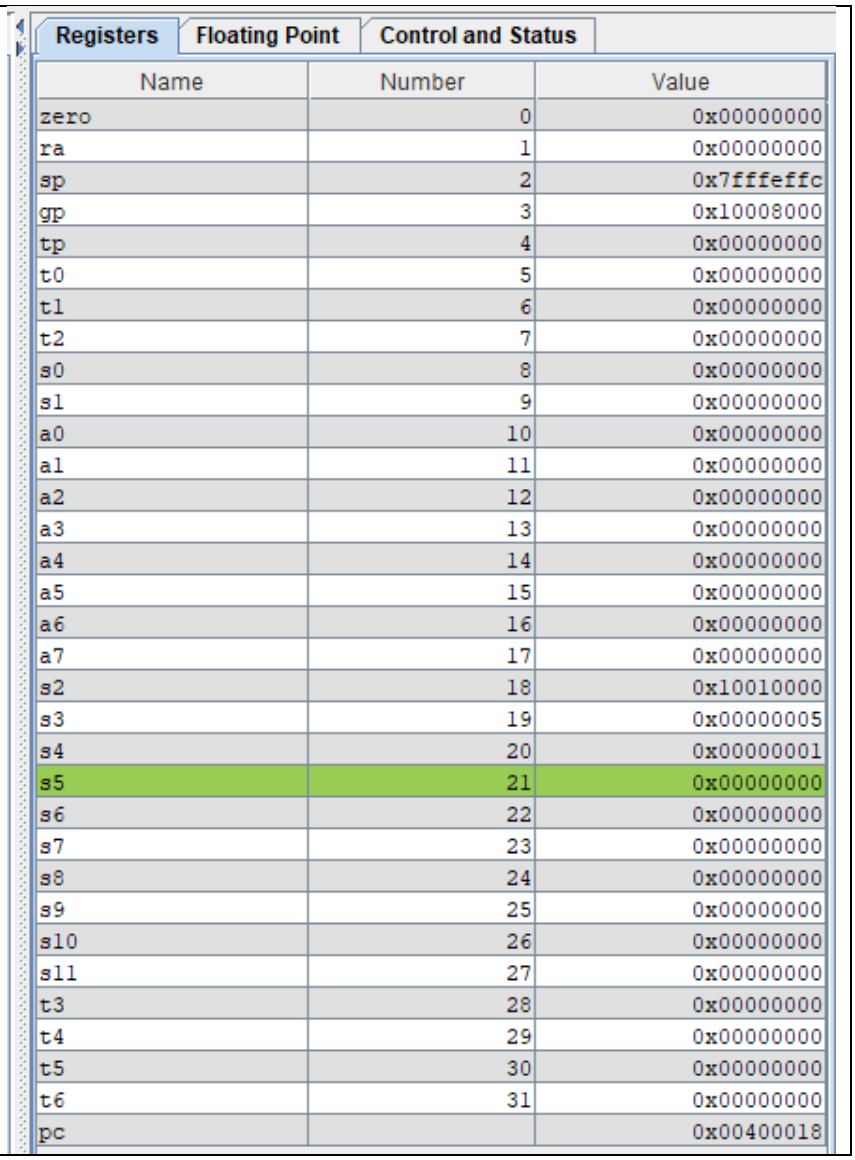
  + Câu lệnh *lw t0, 0(t1)* để lấy giá trị của phần tử A[i]

  + Câu lệnh *add s5, s5, t0* dùng để cộng tổng các phần tử của A.

  +Câu lệnh *add s1, s1, s4* dùng để tăng bước nhảy cho vòng lặp tiếp theo

  + Câu lệnh *j loop* dùng để nhảy đến thẻ loop và tiếp tục vòng lặp

- **Quan sát kết quả trên cửa sổ Register:**
  +Với các giá trị của bảng được khởi tạo lần lượt là 1, 2, 3, 4, 5 . Giá trị của biến đếm là s1 = 0, giá trị của s3 = 5 là số lượng phần tử của mảng, giá trị của s4 = 1 là bước nhảy của vòng lặp, s5 = 0 là giá trị khởi đầu của tổng.

Giá trị của các biến sau khi được khởi tạo	

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400018

+ Giá trị của t1, t0, s5, pc tăng qua mỗi vòng lặp

Vòng lặp	Giá trị của các biến

1

t0	5	0x00000001
t1	6	0x10010000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000001
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x00000001

2

t0	5	0x00000002
t1	6	0x10010004
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000002
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x00000003

3

t0	5	0x00000003
t1	6	0x10010008
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000003
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x00000006

4

t0	5	0x00000004
t1	6	0x1001000c
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x0000000a

5	t0	5	0x00000005
	t1	6	0x10010010
	t2	7	0x00000000
	s0	8	0x00000000
	s1	9	0x00000005
	a0	10	0x00000000
	a1	11	0x00000000
	a2	12	0x00000000
	a3	13	0x00000000
	a4	14	0x00000000
	a5	15	0x00000000
	a6	16	0x00000000
	a7	17	0x00000000
	s2	18	0x10010000
	s3	19	0x00000005
	s4	20	0x00000001
	s5	21	0x0000000f

+ Giá trị ban đầu của thanh pc = 0x00400000 và tăng 4 đơn vị với mỗi câu lệnh được thực hiện.

- **Kiểm tra tính đúng đắn của chương trình:**

  + Với giá trị đã cho sẵn của mảng A gồm 5 phần tử có giá trị lần lượt là 1, 2 , 3, 4, 5 thì giá trị của s5 tăng dần qua mỗi vòng lặp là: 1, 3, 6, 10, 15 theo quy đổi ra hệ 10. Tổng = 1 + 2 + 3 + 4 + 5 = 15 thỏa mãn với kết quả tính toán .

- **Sự thay đổi của bộ nhớ :**

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

3. Assignment 3

- **Nhập chương trình:**

```
# Laboratory Exercise 3, Home Assignment 3
        .data
                test: .word 5
```

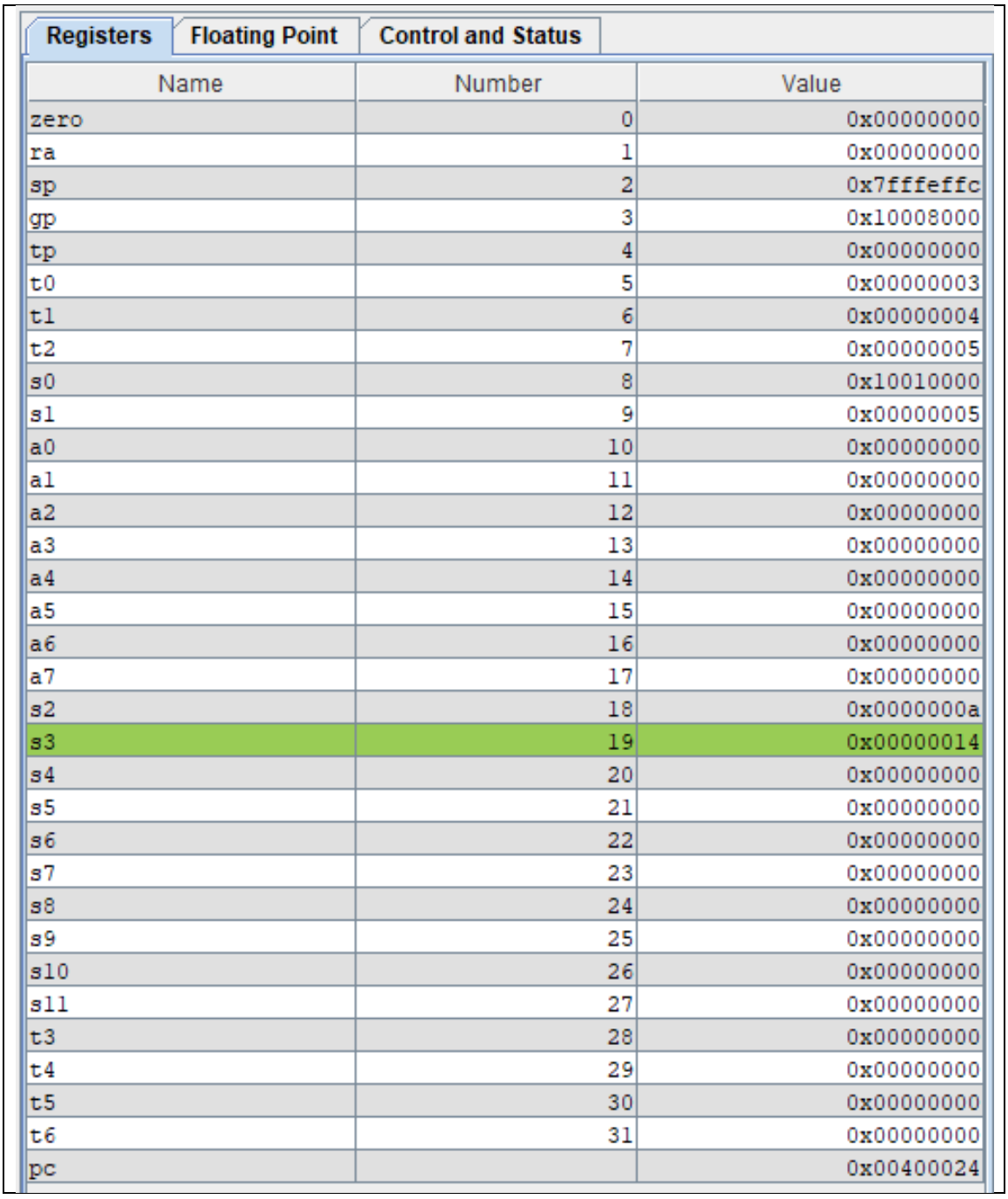
```
        .text
                la s0, test      # Nạp địa chỉ của biến test vào s0
                lw s1, 0(s0)   # Nạp giá trị của biến test vào s1
                li t0, 3       # Nạp giá trị cần kiểm tra
                li t1, 4       # Nạp giá trị cần kiểm tra
                li t2, 5       # Nạp giá trị cần kiểm tra
                li s2, 10     # Khởi tạo giá trị cho a = 10
                li s3, 20     # Khởi tạo giá trị cho b = 20
                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 - t1
                j continue
        case_2:
                add s3, s3, s3 # b = 2 * b
                j continue
        default:
continue:
```

- **Các bước thực hiện của chương trình :**

  + Câu lệnh *test: .word 5* để khởi tạo giá trị bằng 5 cho test
  + Câu lệnh *la s0,* test để nạp địa chỉ của test vào s0
  + Câu lệnh *lw s1, 0(s0)* để nạp giá trị của test vào s1
  + Câu lệnh *li t0, 3* để gián giá trị bằng 3 cho t0
  + Câu lệnh *li t1, 4* để gián giá trị bằng 4 cho t1
  + Câu lệnh *li t2, 5* để gián giá trị bằng 5 cho t2
  + Câu lệnh *li s2, 10* để gián giá trị bằng 10 cho s2 (là a)
  + Câu lệnh *li s3, 20* để gián giá trị bằng 20 cho s3 (là b)
  + Câu lệnh *beq s1, t0, case_0* để so sánh giá trị của s1 và t0. Nếu s1 = t0 thì nhảy đến thẻ case_0;
  + Câu lệnh *beq s1, t1, case_1* để so sánh giá trị của s1 và t1. Nếu s1 = t1 thì nhảy đến thẻ case_1;

+ Câu lệnh *beq s1, t2* case_2 để so sánh giá trị của s1 và t2. Nếu s1 = t1 thì nhảy đến thẻ case_2;
+ Câu lệnh *addi s2, s2, 1* được đặt sau thẻ case_0 để thực hiện a = a + 1. Nếu s1 = t0 thì câu lệnh được thực hiện
+ Câu lệnh *sub s2, s2, t1* được đặt sau thẻ case_1 để thực hiện a = a - t1. Nếu s1= t1 thì câu lệnh được thực hiện
+ Câu lệnh *add s3, s3, s3* được đặt sau thẻ case_2 để thực hiện b = 2 * b. Nếu s1 = t2 thì câu lệnh trên được thực hiện
+ Câu lệnh *j continue* để nhảy đến thẻ continue làm kết thúc cấu trúc switch/case .
+ Thẻ continue đặt ở cuối cùng để kết thúc cấu trúc.

- **Quan sát kết quả trên cửa sổ Register:**
  + Giá trị của các biến sau khi được khởi tạo:



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000003
t1	6	0x00000004
t2	7	0x00000005
s0	8	0x10010000
s1	9	0x00000005
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x0000000a
s3	19	0x00000014
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400024

+ Với giá trị của t0, t1, t2 lần lượt là 3, 4, 5. Giá trị của a = 10, b = 20 và giá trị của **s1 = 5**.Thì kết quả trả về của chương trình là:

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000003
t1	6	0x00000004
t2	7	0x00000005
s0	8	0x10010000
s1	9	0x00000005
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x0000000a
s3	19	0x00000028
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400044

⇨ Giá trị của s3 thay đổi từ 0x00000014 thàng 0x00000028 quy đổi ra hệ thập phân là 40. Do s1 = 5 và t2 = 5 nên chương trình nhảy đến case_2 và thực hiện s3 = s3 + s3 = 20 + 20 đúng với kết quả chương trình.

+ Thay đổi giá trị của **s1 = 3** và giữ nguyên các giá trị khác ta thu được :

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000003
t1	6	0x00000004
t2	7	0x00000005
s0	8	0x10010000
s1	9	0x00000003
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x0000000b
s3	19	0x00000014
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c

⇨ Giá trị của s2 thay đổi từ 0x0000000a thành 0x000000b quy đổi thành hệ thập phân có giá trị bằng 11. Do s1 = 3 và t0 =3 nên chương trình nhảy case_0 và thực hiện a = a + 1 = 10 + 1 = 11 đúng với kết quả chương trình.

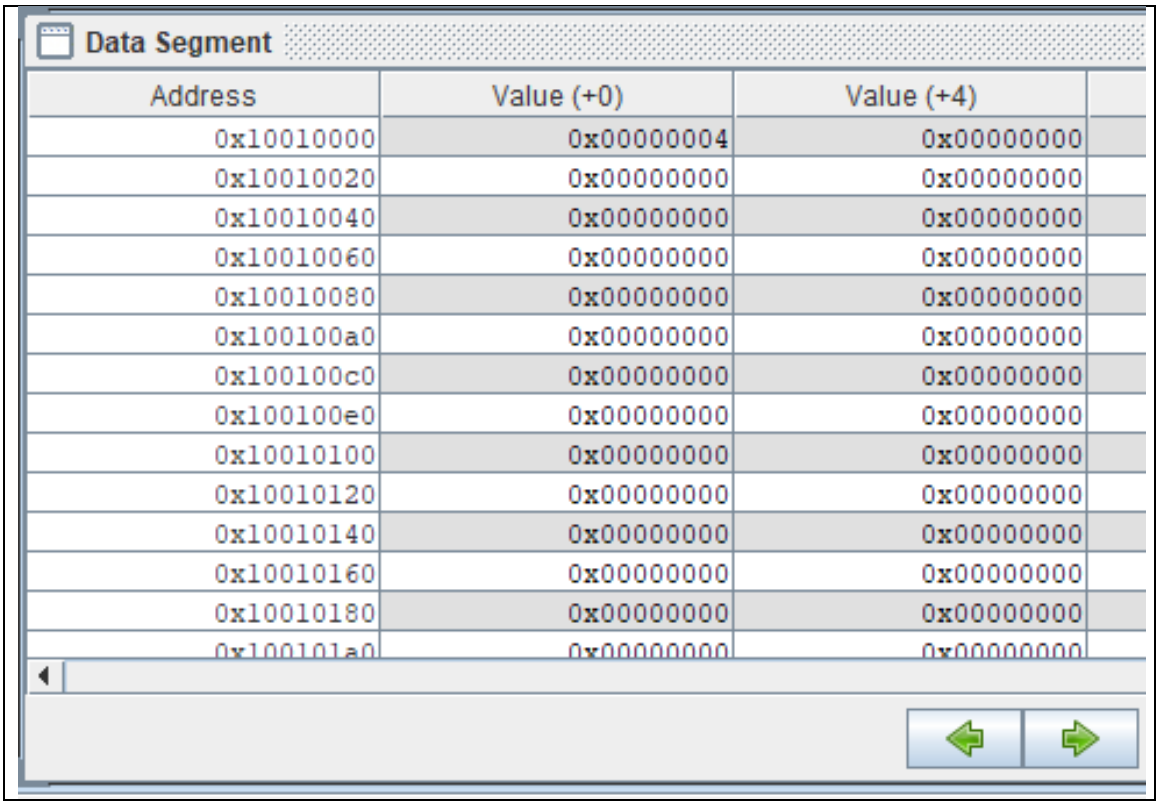
+ Thay đổi giá trị của s1 = 4 và giữ nguyên các giá trị khác ta được :

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000003
t1	6	0x00000004
t2	7	0x00000005
s0	8	0x10010000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000006
s3	19	0x00000014
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c

⇨ Giá trị của s2 thay đổi từ 0x0000000a thành 0x0000006 quy đổi thành hệ thập phân có giá trị bằng 6. Do s1 = 4 và t0 = 4 nên chương trình nhảy case_1 và thực hiện a = a – t1 = 10 – t1 = 6 đúng với kết quả chương trình.

- **Sự thay đổi của bộ nhớ :**



4. Assignment 4

a. i < j

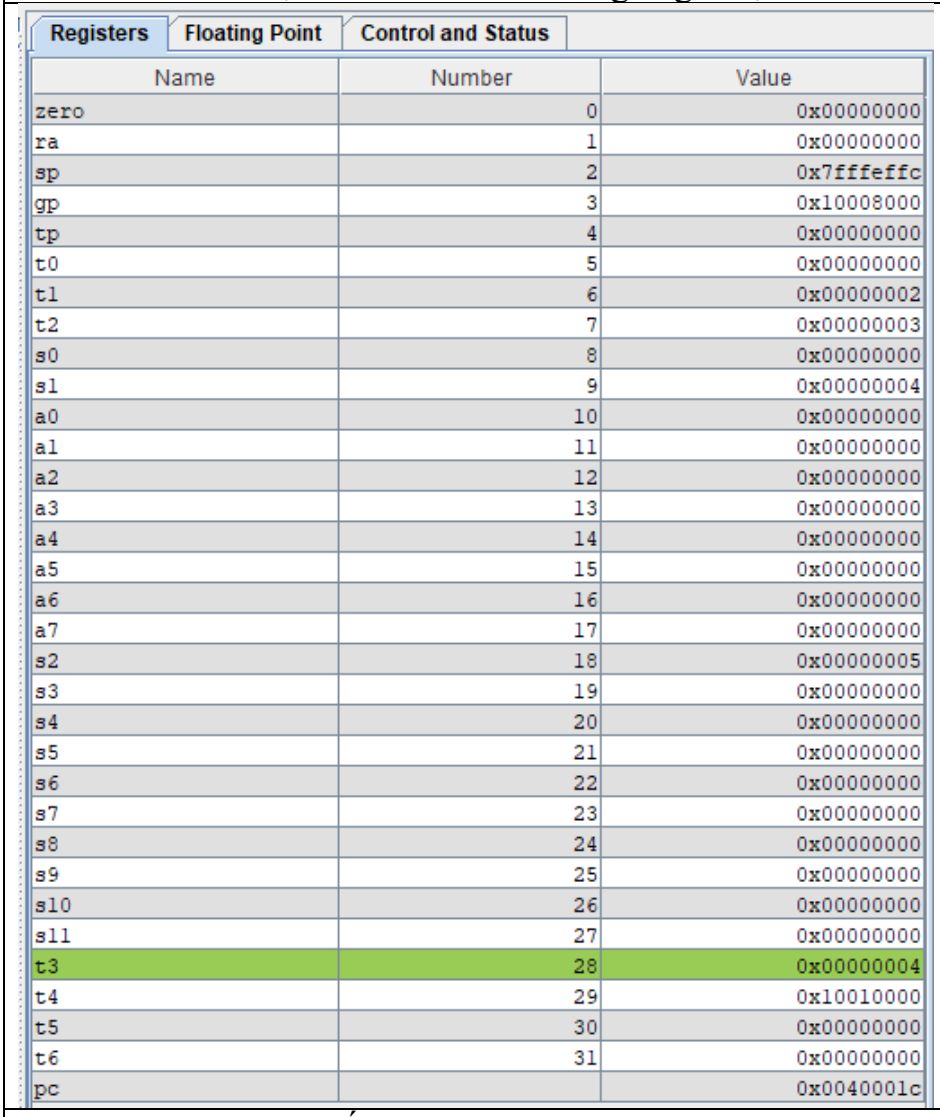
- **Nhập chương trình**

```
# Laboratory Exercise 3,
        .data
                X: .word 2     # Biến X, kiểu word (4 bytes), giá trị khởi tạo = 2
                Y: .word 3     # Biến Y, kiểu word (4 bytes), giá trị khởi tạo = 3
                Z: .word 4     # Biến Z, kiểu word (4 bytes), giá trị khởi tạo = 4
        .text
        start:
        li s1, 4               # Khởi tạo giá trị 4 cho s1
        li s2, 5               # Khỏi tạo giá trị 5 cho s2

        la t4, X               # Lấy địa chỉ của X trong vùng nhớ
        lw t1, 0(t4)           # Gán giá trị của X vào t1
        lw t2, 4(t4)           # Gán giá trị của Y vào t2
        lw t3, 8(t4)           # Gán giá trị của Z vào t3

        blt s1, s2, else       # if i < j then jump else
```

```
        then:
                addi  t1, t1, 1         # x=x+1
                addi  t3, zero, 1       # z=1
                j  endif                # Bước nhảy đến endif
        else:
                addi  t2, t2, -1    # y=y-1
                add   t3, t3, t3    # z=2*z
endif:
```

- **Kết quả của chương trình :**

Giá trị của các biến sau khi gán giá trị



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000004
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

Giá trị của các biến sau khi hoàn thành chương trình

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000002
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000008
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400038

- **Giải thích :**

  + Giá trị của i = 4 và giá trị của j = 5 . Sau khi thực hiện câu lệnh blt *s1, s2, else* tương ứng với nếu s1 nhỏ hơn s2 thì nhảy đến else. Do 5 > 4 nên nhảy đến else và thực hiện y = y -1 = t2 -1 = 3 -1 = 2 và z = z * 2 = t3 * 2 = 4 * 2 = 8. Đúng với kết quả của chương trình.

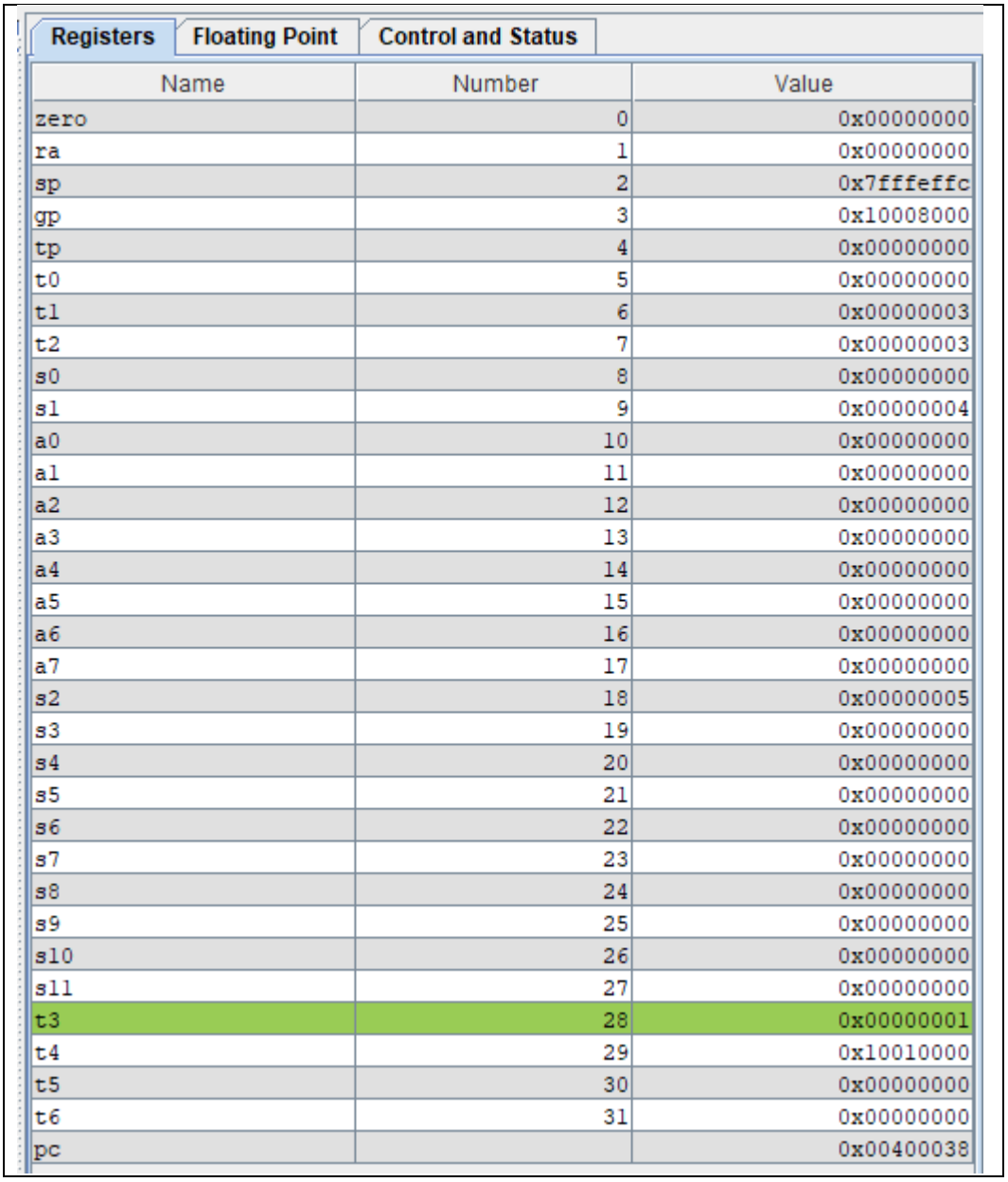
b. i >= j

- **Nhập chương trình:**

```
# Laboratory Exercise 3,
        .data
        X: .word 2     # Biến X, kiểu word (4 bytes), giá trị khởi tạo = 2
        Y: .word 3     # Biến Y, kiểu word (4 bytes), giá trị khởi tạo = 3
```

```
        Z: .word 4     # Biến Z, kiểu word (4 bytes), giá trị khởi tạo = 4
.text
start:
        li s1, 4                # Khởi tạo giá trị 4 cho s1
        li s2, 5                # Khỏi tạo giá trị 5 cho s2

        la t4, X                # Lấy địa chỉ của X trong vùng nhớ
        lw t1, 0(t4)            # Gán giá trị của X vào t1
        lw t2, 4(t4)            # Gán giá trị của Y vào t2
        lw t3, 8(t4)            # Gán giá trị của Z vào t3

        bge s1, s2, else        # if i>=j then jump else
        then:
                addi  t1, t1, 1         # x=x+1
                addi  t3, zero, 1       # z=1
                j  endif                # Bước nhảy đến endif
        else:
                addi  t2, t2, -1    # y=y-1
                add   t3, t3, t3    # z=2*z
endif:
```

- **Kết quả của chương trình:**



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000003
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400038

- **Giải thích :**

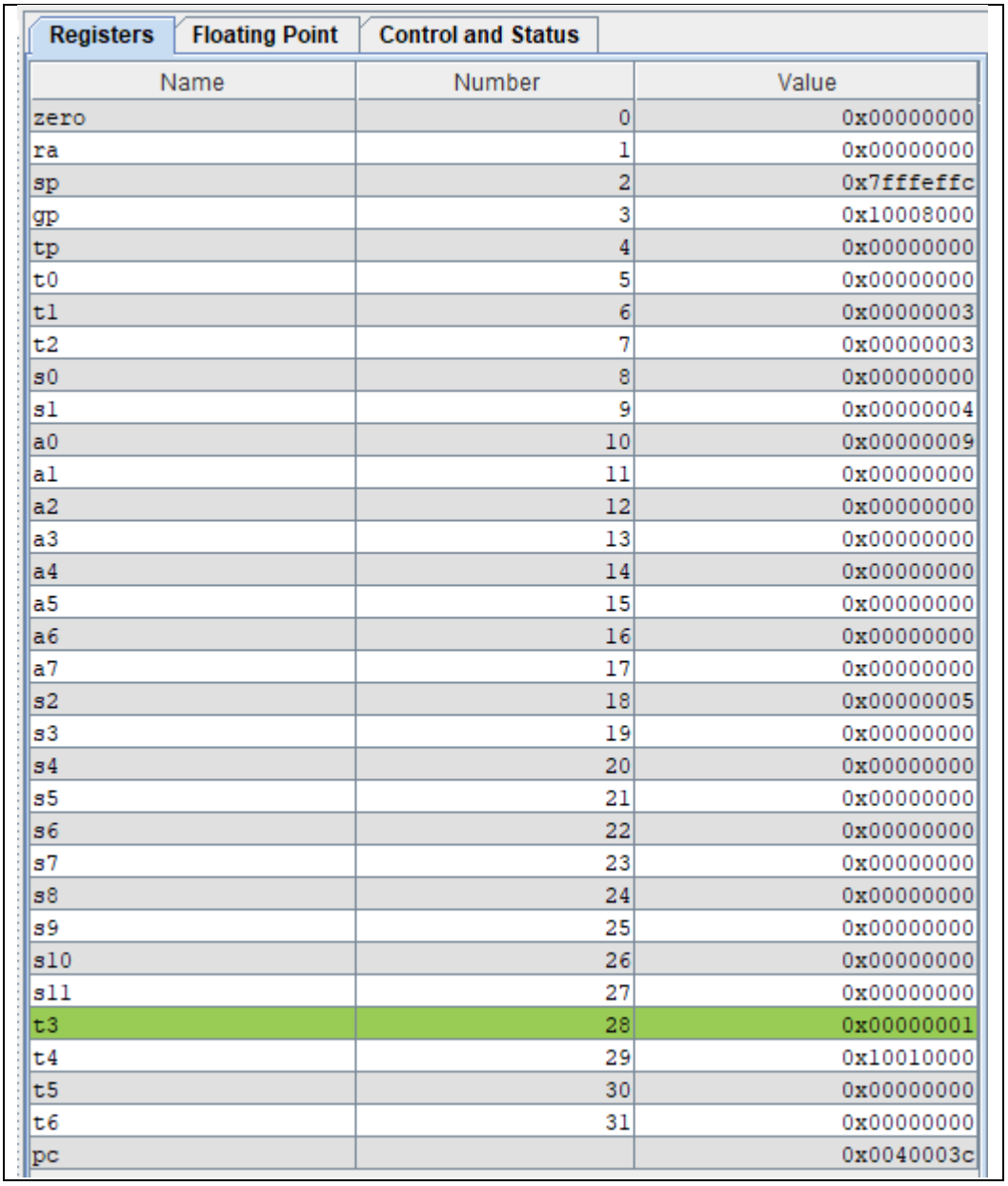
  + Giá trị của i = 4 và giá trị của j = 5 . Sau khi thực hiện câu lệnh *bge s1, s2, else* tương ứng với nếu s1 lớn hơn hoặc bằng s2 thì nhảy đến else. Do 4 < 5 nên chương trình tiếp tục thực hiện x = x + 1 = t1 + 1 = 2 +1 = 3 và z = t3=1. Đúng với kết quả chương trình.

c. i + j <= 0

- **Nhập chương trình:**

```
# Laboratory Exercise 3,
        .data
        X: .word 2     # Biến X, kiểu word (4 bytes), giá trị khởi tạo = 2
        Y: .word 3     # Biến Y, kiểu word (4 bytes), giá trị khởi tạo = 3
        Z: .word 4     # Biến Z, kiểu word (4 bytes), giá trị khởi tạo = 4
.text
start:
        li s1, 4                # Khởi tạo giá trị 4 cho s1
        li s2, 5                # Khỏi tạo giá trị 5 cho s2

        la t4, X                # Lấy địa chỉ của X trong vùng nhớ
        lw t1, 0(t4)            # Gán giá trị của X vào t1
        lw t2, 4(t4)            # Gán giá trị của Y vào t2
        lw t3, 8(t4)            # Gán giá trị của Z vào t3
        add a0, s1, s2                  # Gán a0 = s1 + s2
        bge zero, a0, else   # if 0 >= i + j then jump else
        then:
                addi  t1, t1, 1         # x=x+1
                addi  t3, zero, 1       # z=1
                j  endif                # Bước nhảy đến endif
        else:
                addi  t2, t2, -1    # y=y-1
                add   t3, t3, t3    # z=2*z
endif:
```

- **Kết quả chương trình**



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000003
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000009
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040003c

+ Giá trị của i = 4 và giá trị của j = 5 . Sau khi thực hiện câu lệnh *bge zero, a0, else* tương ứng với nếu 0 >= i + j thì nhảy đến else. Do 9 > 0 nên chương trình tiếp tục thực hiện x = x + 1 = t1 + 1 = 2 +1 = 3 và z = t3=1. Đúng với kết quả chương trình.

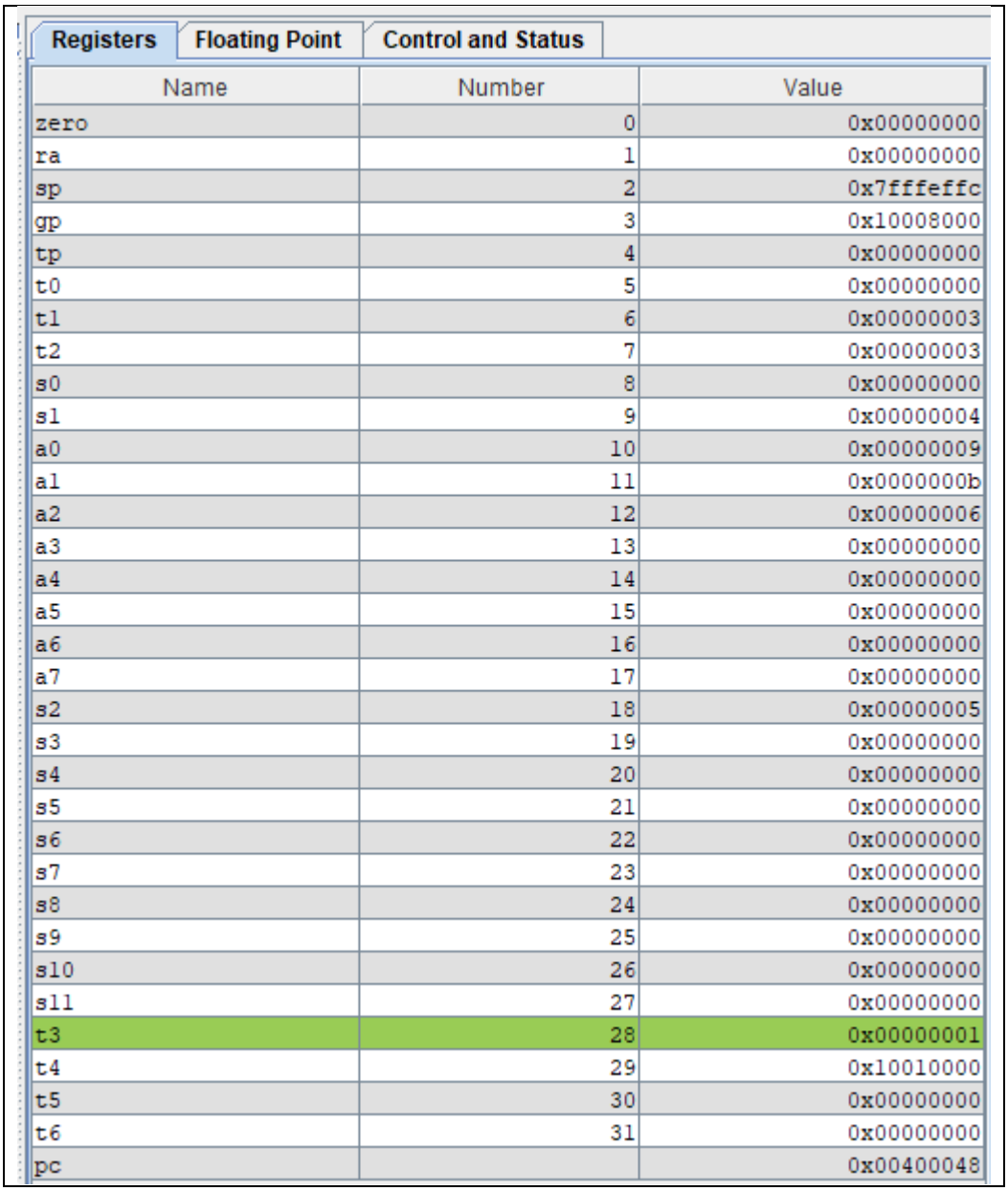
d. i + j > m + n

- **Nhập chương trình:**

```
# Laboratory Exercise 3,
.data
```

```
        X: .word 2     # Biến X, kiểu word (4 bytes), giá trị khởi tạo = 2
        Y: .word 3     # Biến Y, kiểu word (4 bytes), giá trị khởi tạo = 3
        Z: .word 4     # Biến Z, kiểu word (4 bytes), giá trị khởi tạo = 4
        A: .word 5     # Biến A, kiểu word (4 bytes), giá trị khởi tạo = 5
        B: .word 6     # Biến B, kiểu word (4 bytes), giá trị khởi tạo = 6
.text
start:
        li s1, 4               # Khởi tạo giá trị 4 cho s1
        li s2, 5               # Khỏi tạo giá trị 5 cho s2

        la t4, X               # Lấy địa chỉ của X trong vùng nhớ
        lw t1, 0(t4)           # Gán giá trị của X vào t1
        lw t2, 4(t4)           # Gán giá trị của Y vào t2
        lw t3, 8(t4)           # Gán giá trị của Z vào t3
        lw a1, 12(t4)          # Gán giá trị của A vào m
        lw a2, 16(t4)          # Gán giá trị của B vào n
        add a1, a1, a2                 # Gán a1 = m + n
        add a0, s1, s2                 # Gán a0 = s1 + s2
        blt a1, a0, else       # if i+j > m+n then jump else
        then:
                addi  t1, t1, 1        # x=x+1
                addi  t3, zero, 1      # z=1
                j  endif               # Bước nhảy đến endif
        else:
                addi  t2, t2, -1    # y=y-1
                add   t3, t3, t3    # z=2*z
endif:
```

- **Kết quả**



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000003
t2	7	0x00000003
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000009
a1	11	0x0000000b
a2	12	0x00000006
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000005
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x10010000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400048

- **Giải thích:**

  Giá trị của i = 4, j = 5, m = 5, n = 6. Sau khi thực hiện câu lệnh *blt a1, a0, else* tương ứng với nếu i + j > m + n thì nhảy đến else. Do 4 + 5 < 5 + 6 nên chương trình tiếp tục thực hiện x = x + 1 = t1 + 1 = 2 +1 = 3 và z = t3=1. Đúng với kết quả chương trình.

5. Assignment 5

a. i > n

- **Nhập chương trình:**

```
# Laboratory 3,
        .data
        A: .word 1, 2, 3, 4, 5
        # khởi tạo mảng có 5 phần tử
        .text
                li s1, 0        # gía trị bắt đầu duyệt
                la s2, A        # Lưu địa chỉ của A vào s2
                li s3, 5        # khởi tạo giá trị s3 là số phần tử
                li s4, 1        # khởi tạo bước nhảy
                li s5, 0        # Khởi tạo giá trị ban đầu cho tổng
        loop:
                blt s3, s1, endloop          # Điều kiện dùng của vòng lặp
                add t1, s1, s1          # t1 = 2 * s1
                add t1, t1, t1          # t1 = 4 * s1 => t1 = 4*i
                add t1, t1, s2          # t1 store the address of A[i]
                lw  t0, 0(t1)           # lấy giá trị của từng phần tử trong A
                add s5, s5, t0                  # tính tổng từng phần tử trong A
                add s1, s1, s4                  # tăng bước nhảy
                j  loop                 # quay lại loop để tiếp tục lặp
endloop:
```

- **Kết quả**

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x10010014
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000006
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0x0000000f
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040003c

b. sum < 0

- **Nhập chương trình:**

```
# Laboratory 3,
.data
        A: .word 1, 2, 3, -4, -5
        # khởi tạo mảng có 5 phần tử
        .text
                li s1, 0        # gía trị bắt đầu duyệt
                la s2, A        # Lưu địa chỉ của A vào s2
```

```
        li s3, 5        # khởi tạo giá trị s3 là số phần tử
        li s4, 1        # khởi tạo bước nhảy
        li s5, 0        # Khởi tạo giá trị ban đầu cho tổng
    loop:
        blt s5, zero, endloop   # Điều kiện dùng của vòng lặp
        add t1, s1, s1          # t1 = 2 * s1
        add t1, t1, t1          # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2          # t1 store the address of A[i]
        lw  t0, 0(t1)           # lấy giá trị của từng phần tử trong A
        add s5, s5, t0                  # tính tổng từng phần tử trong A
        add s1, s1, s4                  # tăng bước nhảy
        j  loop                 # quay lại loop để tiếp tục lặp
endloop:
```

-Kết quả :

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0xfffffffb
t1	6	0x10010010
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000005
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000005
s4	20	0x00000001
s5	21	0xfffffffd
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040003c

6. Assignment 6

- **Nhập chương trình:**

```
.data
        A: .word -10, 25, -40, 30, -50, 15, 5
        n: .word 7
        max_abs: .word 0
.text
        la a0, A
        lw a1, n
```

```
        li a2, 0         # index i = 0
        li a3, 0        # max_abs = 0
loop:
        bge a2, a1, endloop          # If i >= n, exit loop
        slli t0, a2, 2           # t0 = i * 4 (each .word is 4 bytes)
        add t0, a0, t0           # t0 = &A[i]
        lw t1, 0(t0)             # t1 = A[i]
        blt t1, zero, rev        # if A[i] < 0, take -A[i]
        j comp
rev:
        sub t1, zero, t1         # abs(A[i])
comp:
        bge t1, a3, upd_max          # if abs(A[i]) >= max_abs, upd
        j next
upd_max:
        mv a3, t1  # max_abs = abs(A[i])
next:
        addi a2, a2, 1  # i++
        j loop
endloop:
```

- **Kết quả:**

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x10010018
t1	6	0x00000005
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10010000
a1	11	0x00000007
a2	12	0x00000007
a3	13	0x00000032
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c