BÁO CÁO THỨC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 4

Họ và tên: Vũ Đức Hoàng Anh

MSSV: 20235658

1. Assignment 1:

- **Nhập chương trình:**

```
# Laboratory Exercise 4, Home Assignment 1
.text
        li s1 1000000000   # Khởi tạo giá trị cho s1
        li s2 2000000000   # Khởi tạo giá trị cho s2
# Thuật toán xác định tràn số
        li    t0, 0            # Mặc định không có tràn số
        add   s3, s1, s2       # s3 = s1 + s2
        xor   t1, s1, s2       # Kiểm tra s1 với s2 có cùng dấu
        blt   t1, zero, EXIT    # Nếu t1 là số âm, s1 và s2 khác dấu
        blt   s1, zero, NEGATIVE   # Kiểm tra s1 và s2 là số âm hay không
        âm
        bge   s3, s1, EXIT        # s1 không âm, kiểm tra s3 nhỏ hơn s1
        không
        # Nếu s3 >= s1, không tràn số
        j     OVERFLOW
NEGATIVE:
        bge   s1, s3, EXIT    # s1 âm, kiểm tra s3 có lớn hơn s1 không
        # Nếu s1 >= s3, không tràn số
OVERFLOW:
        li    t0, 1         # The result is overflow
EXIT:
```

- **Các biến khởi tạo:**

  + Khởi tạo giá trị của s1 = 1000000000

  + Khởi tạo giá trị của s2 = 2000000000

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

  + Câu lệnh li s1 1000000000 để gán giá trị cho s1 = 1000000000

  + Câu lệnh li s2 2000000000 để gán giá trị cho s2 = 2000000000

  + Câu lệnh li t0, 0 để gán giá trị cho t0 = 0. Mặc định không có hiện tượng tràn số.

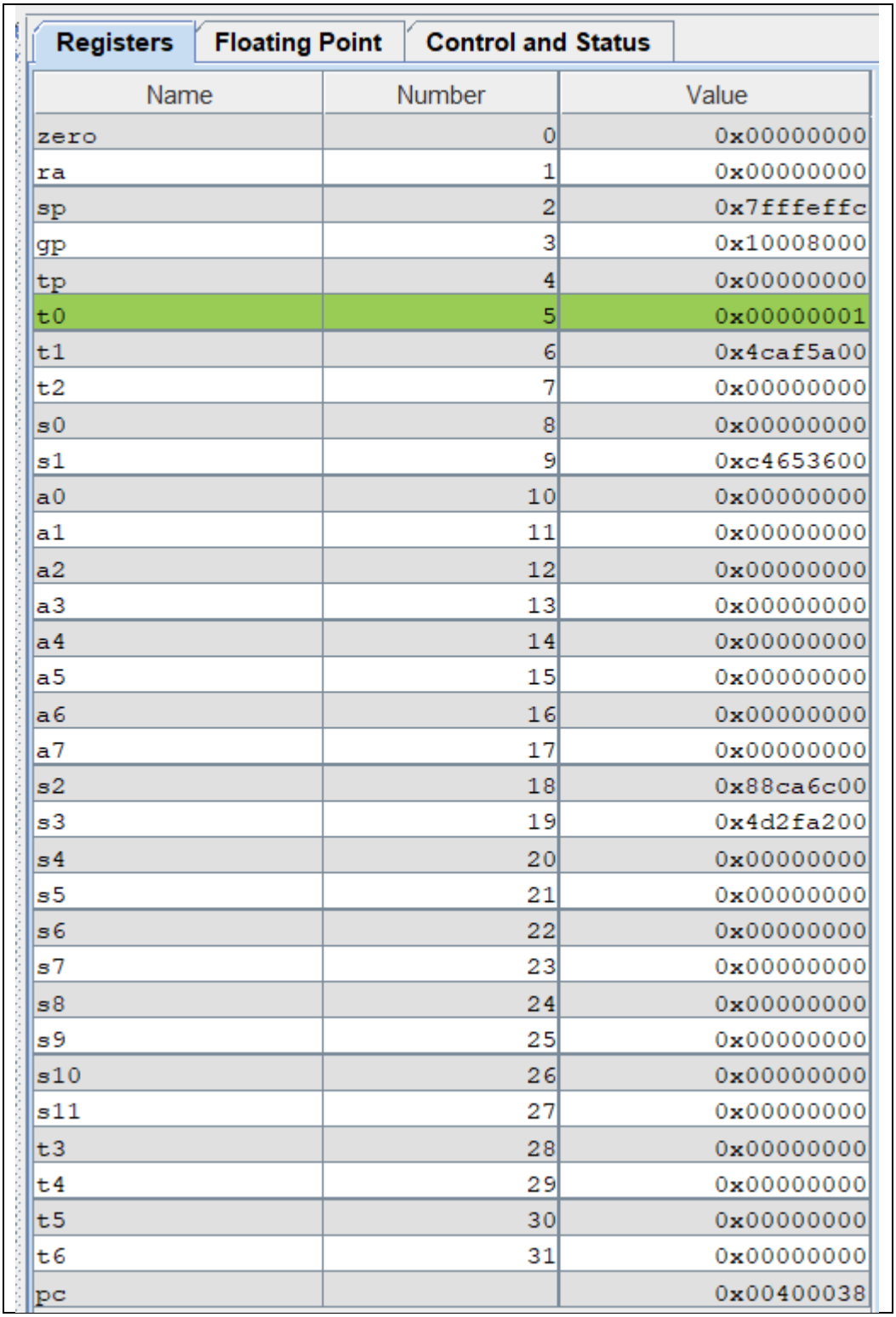
+ Câu lệnh add s3, s1, s2 để tính tổng s3 = s1 + s2;
+ Câu lệnh xor t1, s1, s2 để kiểm tra xem s1 và s2 có cùng dấu hay không. Nếu s1 và s2 khác dấu thì trả về t1 có kết quả âm.
+ Câu lệnh blt t1, zero, EXIT để so sánh t1 với zero. Nếu t1 < 0 thì s1 và s2 khác dấu điều kiện đúng thì nhảy đến thẻ EXIT.
+ Câu lệnh blt s1, zero, NEGATIVE để so sánh s1 với zero. Nếu s1 < 0 thì s1 và s2 cùng âm và nhảy đến thẻ NEGATIVE.
+ Câu lệnh bge s3, s1, EXIT để kiểm tra xem s3 có nhỏ hơn s1 hay không. Do đã kiểm tra điều kiện 2 số cùng dương nên nếu s3 nhỏ hơn s1 thì sảy ra hiện tượng tràn số.
+ Câu lệnh j OVERFLOW dùng để nhảy đến thẻ OVERFLOW. Khi trường hợp tràn số sảy ra sẽ nhảy đến OVERFLOW và thực hiện thay đổi giá trị của t0.
+ Câu lệnh bge s1, s3, EXIT dùng để so sánh s1 có nhỏ hơn s3 hay không. Do đã kiểm tra điều kiện 2 số cùng âm nên nếu s1 nhỏ hon s3 thì sảy ra hiện tượng tràn số.
+ Câu lệnh li t0, 1 dùng để thay đổi giá trị của t0 =1. Được đặt trong thẻ OVERFLOW, khi có hiện tượng tràn số sảy ra thì thay đổi giá trị của t0

- **Quan sát kết quả trên cửa sổ Register:**
  + Giá trị của s1 = 1000000000 và s2 = 2000000000 sẽ sảy ra hiện tượng tràn số .

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	0x00000001
t1	6	0x4caf5e00
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x3b9aca00
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	0x77359400
s3	19	0xb2d05e00
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		0x00400038

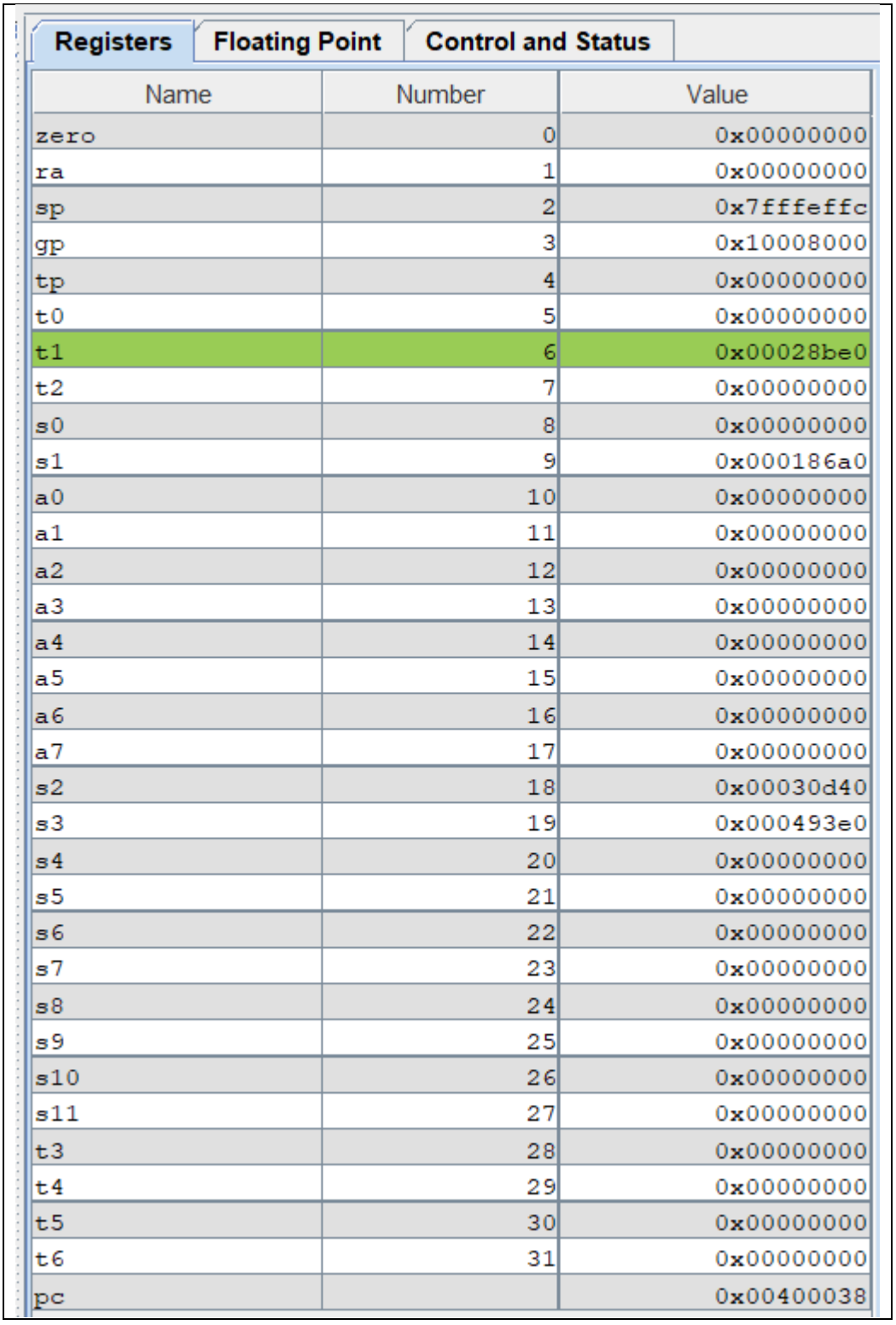
+ Thay đổi bộ giá trị s1 = -1000000000 và s2 = -1000000000 sẽ sảy ra hiện tượng tràn số.



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	0x00000001
t1	6	0x4caf5a00
t2	7	0x00000000
s0	8	0x00000000
s1	9	0xc4653600
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	0x88ca6c00
s3	19	0x4d2fa200
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		0x00400038

+ Giá trị của s1 = 100000 và s2 = 200000 sẽ không sảy ra hiện tượng tràn số.



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	0x00028be0
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x000186a0
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	0x00030d40
s3	19	0x000493e0
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		0x00400038

2. Assignment 2

- **Nhập chương trình:**

```
.text
        li s0, 0x20235658       # Khởi tạo giá trị cho s0 = 0x20235658

# 1. Trích xuất MSB (Most Significant Byte)
        srli t0, s0, 24     # Dịch phải 24 bit để lấy byte cao nhất (MSB)

# 2. Xóa LSB (Least Significant Byte)
        li t1, 0xFFFFFF00   # Khởi tạo giá trị để xóa LSB
        and s0, s0, t1      # Xóa LSB của s0

# 3. Thiết lập LSB thành 0xFF (tất cả bit của byte thấp nhất là 1)
        ori s0, s0, 0xFF    # OR với 0xFF để thiết lập byte thấp nhất

# 4. Xóa thanh ghi s0 bằng cách sử dụng các lệnh logic
      xor s0, s0, s0      # Xóa toàn bộ nội dung của s0 (s0 = 0)
```

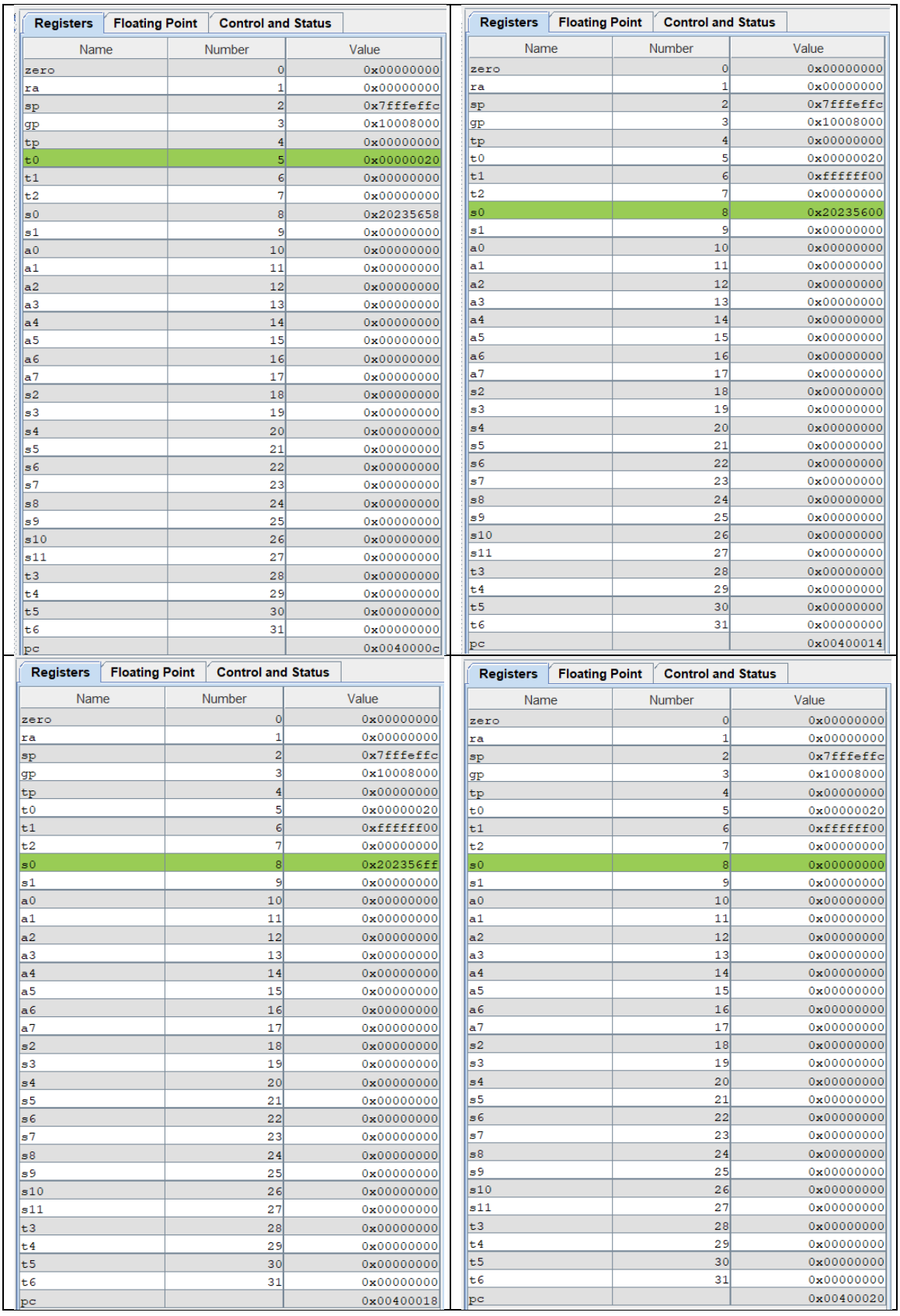
- **Khởi tạo giá trị:**

  + Giá trị của s0 = 0x20235658
  + Giá trị của t1 = 0xFFFFFF00

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

  + Câu lệnh li s0, 0x20235658 để khởi tạo giá trị cho s0 = 0x20235658
  + Câu lệnh srli t0, s0, 24 dùng để dịch phải 24 bit để lấy byte cao nhất. Do dịch phải 24 bit để giữ ại 2 bit cao nhất lưu vào t0;
  + Câu lệnh li t1, 0xFFFFFF00 để khởi tạo giá trị cho t1 = 0xFFFFFF00
  + Câu lệnh and s0, s0, t1 để xóa LSB của s0
  + Câu lệnh ori s0, s0, 0xFF để thiết lập byte bé nhất
  + Câu lệnh xor s0, s0, s0 dùng để xóa toàn bộ nội dung của s0 (s0 = 0) thông qua phép logic xor.

- **Kết quả chạy của chương trình của mỗi phần:**



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	0x00000020
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x20235658
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	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		0x0040000c

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	0x00000020
t1	6	0xffffff00
t2	7	0x00000000
s0	8	0x20235600
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	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		0x00400014

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	0x00000020
t1	6	0xffffff00
t2	7	0x00000000
s0	8	0x202356ff
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	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		0x00400018

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	0x00000020
t1	6	0xffffff00
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	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		0x00400020

3. **Assignment 3**

a. neg s0, s1 (s0 = -s1)

- **Nhập chương trình:**

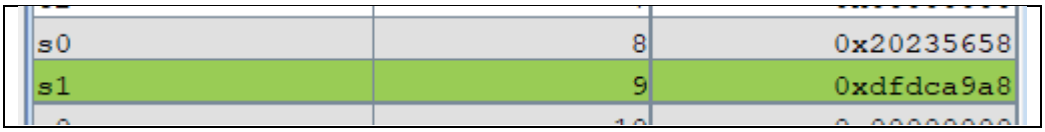
```
.text
li s1 0x20235658
sub s0, zero, s1
```

- **Các bước thực hiện**

+ Câu lệnh li s0 0x20235658 để khởi tạo giá trị cho s0

+ Câu lệnh sub s0, zero, s1 để thực hiện phép trừ s0 = 0 – s1

- **Kết quả thực hiện**



s0	8	0x20235658
s1	9	0xdfdca9a8

b. mv s0, s1 (s0 = s1)

- **Nhập chương trình:**

```
.text
li s1 0x20235658
add s0, zero, s1
```

- **Các bước thực hiện**

+ Câu lệnh li s0 0x20235658 để khởi tạo giá trị cho s0

+ Câu lệnh add s0, zero, s1 để thực hiện phép trừ s0 = 0 + s1

- **Kết quả thực hiện:**

s0	8	0x20235658
s1	9	0x20235658

c. not s0 (s0 = bit_invert(s0))

- **Nhập chương trình:**

```
.text
li s0  0x20235658
xori s1, s0, 0xFFFFFFFF
```

- **Các bước thực hiện:**

+ Câu lệnh li s0 0x20235658 để khởi tạo giá trị cho s0

+ Câu lệnh xori s1, s0, 0xFFFFFFFF để xor s0 với 0xFFFFFFFF

- **Kết quả thực hiện:**

s0	8	0x20235658
s1	9	0xdfdca9a7

d. ble s1, s2, label (if s1 <= s2 j label)

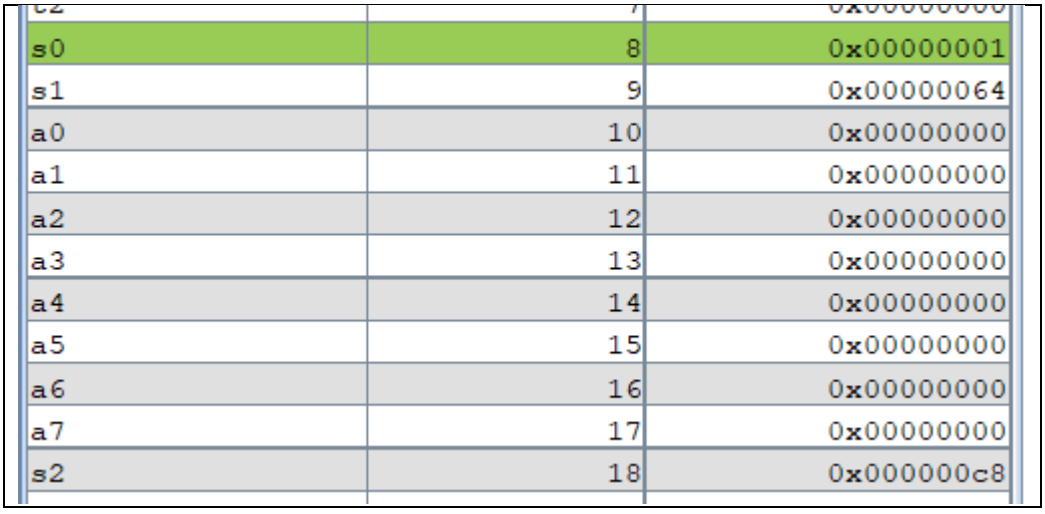
- **Nhập chương trình:**

```
.text
        li s1 100
        li s2 200
        li s0 0
        bge s2, s1, else
then:
        j end
else:
        li s0 1
end:
```

- **Các bước thực hiện:**

+ Câu lệnh li s1 100 để gán giá trị s1 = 100
+ Câu lệnh li s2 200 để gán giá trị s2 = 200
+ Câu lệnh li s0 0 để gán giá trị s0 = 0
+ Câu lệnh bge s2, s1, else để so sánh nếu s2 >= s1 thì nhảy đến thẻ else
+ Câu lênh j end để nhay đến thẻ end
+ Câu lệnh li s0 1 để gán giá trị của s0 =1 nhằm xác định xem nếu s2 >= s1 thì đổi giá trị của s0 = 1;

- **Kết quả thực hiện của chương trình:**



s0	8	0x00000001
s1	9	0x00000064
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	0x000000c8

4. Assignment 4

- **Nhập chương trình:**

```
.text
        li s1 1000000000   # Gán giá trị cho s1
        li s2 2000000000   # Gán giá trị cho s1
        li t0 0            # Mặc định là không tràn số

        xor t1, s1, s2     # Kiểm tra s1 và s2 có cùng dấu hay không
        blt  t1, zero, EXIT# Kiểm tra dấu của t1
        add s3, s1, s2          # Tính tổng s1, s2
        xor t1, s3, s1          # Kiểm tra dấu của s3, s1
        blt  zero, t1, EXIT# Kiểm tra dấu của t1
then:
        # Nếu không nhảy là sảy ra hiện tượng tràn số
        li t0 1            # Thay đổi giá trị cho t0 = 1
EXIT:
```

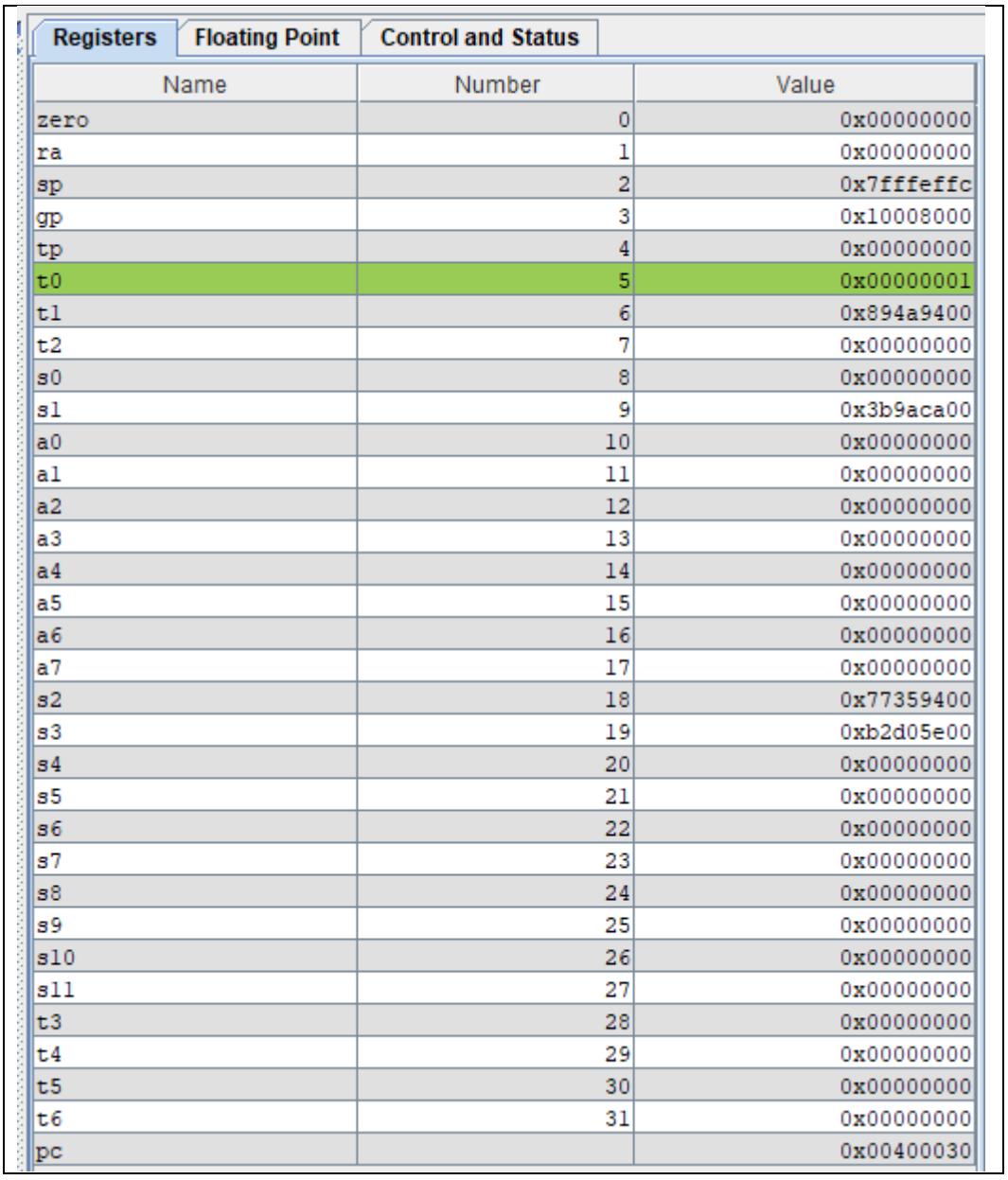
- **Khởi tạo giá trị**

  + Giá trị của s1 = 1000000000
  + Giá trị của s2 = 2000000000
  + Giá trị của t0 = 0

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

  + Các câu lệnh khởi tạo giá trị của s1, s2, t0
  + Câu lệnh xor t1, s1, s2 để kiểm tra xem s1 và s2 có cùng dấu hay không. Vì khi xor 2 số bit đầu tiên là bit dấu, nếu 2 số dùng dấu xor sẽ trả về 0 ở bit cao nhất hay là số dương, và ngược lại nếu trái dấu trả về 1 là số âm.
  + Câu lệnh blt t1, zero, EXIT để so sánh giá trị của t1 với 0. Nếu t1 nhỏ hơn 0 thì nhảy đến thẻ EXIT có nghĩa là s1 và s2 khác dấu sẽ nhảy đến EXIT
  + Câu lệnh add s3, s1, s2 để tính tổng s3 = s1 + s2
  + Câu lệnh xor t1, s3, s1 để kiểm tra dấu của s3 và s1
  + Câu lệnh blt zero, t1, EXIT để nếu t1 lớn hơn 0 thì nhảy đến thẻ EXIT. Có nghĩa là khi t1 > 0 thì s3 và s1 khác dấu không xảy ra hiện tượng tràn số và nhảy đến thẻ EXIT.
  + Câu lệnh li t0, 1 để gán giá trị cho t0 =1 khi xảy ra hiện tượng tràn số

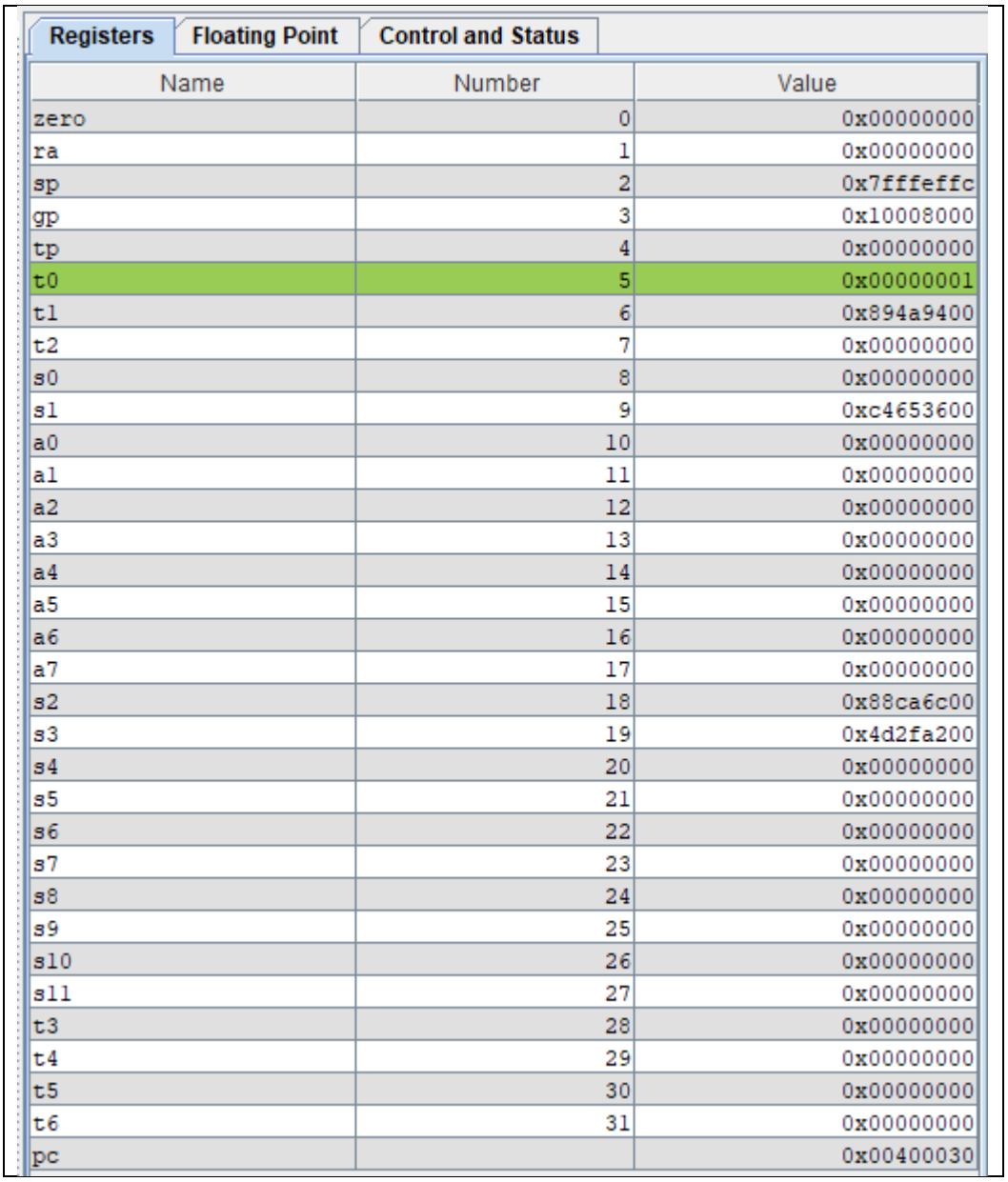
- **Quan sát kết quả trên cửa sổ Register:**
  + Giá trị của s1 = 1000000000 và s2 = 2000000000 sẽ sảy ra hiện tượng tràn số .



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	0x00000001
t1	6	0x894a9400
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x3b9aca00
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	0x77359400
s3	19	0xb2d05e00
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		0x00400030

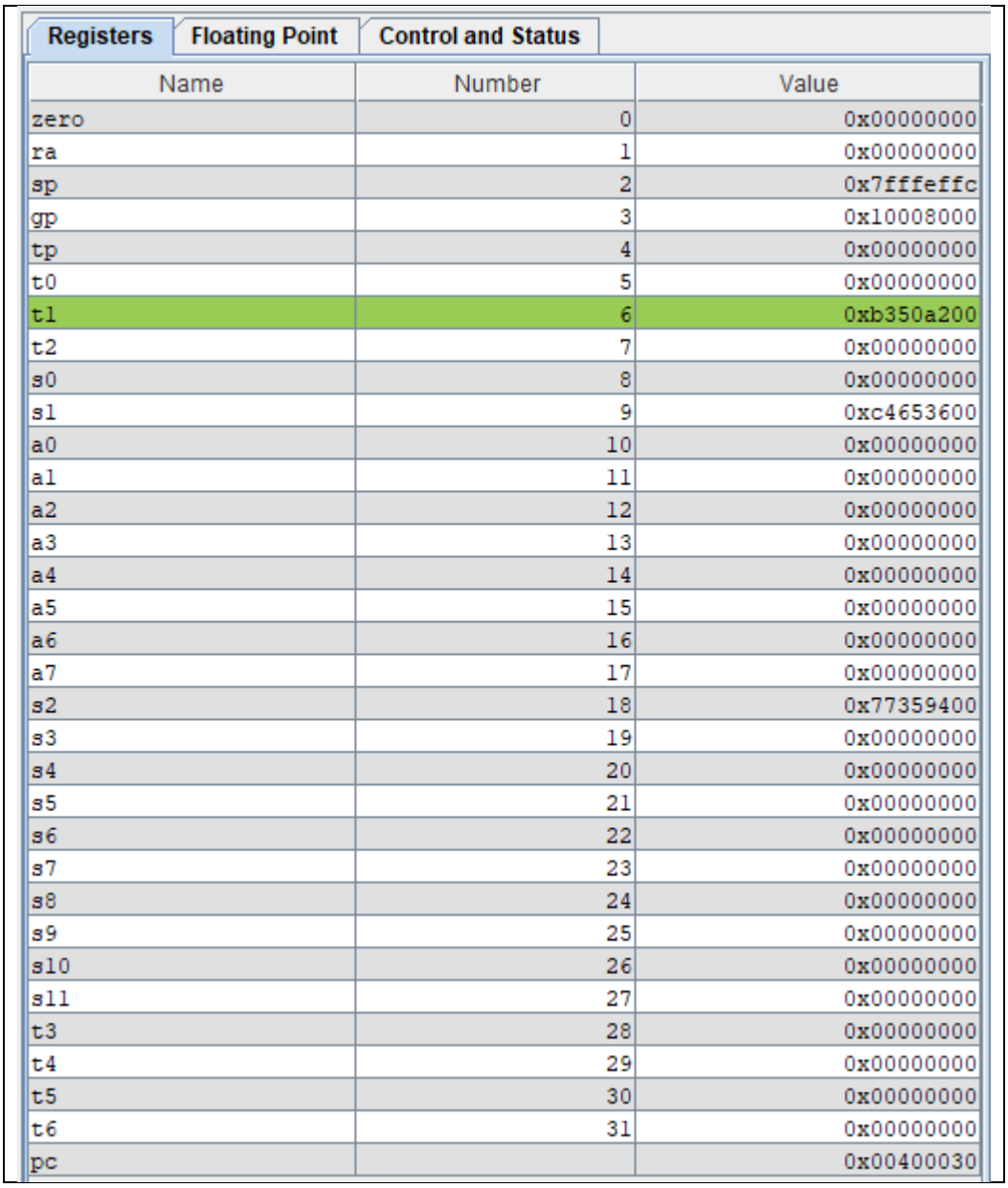
  + Giá trị của s1 = -1000000000 và s2 = -2000000000 sẽ sảy ra hiện tượng tràn số .



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	0x00000001
t1	6	0x894a9400
t2	7	0x00000000
s0	8	0x00000000
s1	9	0xc4653600
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	0x88ca6c00
s3	19	0x4d2fa200
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		0x00400030

+ Giá trị của s1 = 1000000000 và s2 = -2000000000 sẽ không sảy ra hiện tượng tràn số .



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	0xb350a200
t2	7	0x00000000
s0	8	0x00000000
s1	9	0xc4653600
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	0x77359400
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		0x00400030

5. Assignment 5

- **Nhập chương trình:**

```
.text
        li s1 6         # Gán giá trị cho s1
        li s2 8         # Gán giá trị cho s2
        li t0, 0        # Gán giá trị cho biến đến t0
        li s0, 0        # Gán giá trị cho s0 = 0
        li t1, 1        # Tạo hằng t1 để so sánh
        li s3, 32       # Khởi tạo số vòng lặp
```

```
LOOP:
        bge t0, s3, ENDLOOP  # Nếu đã lặp 32 lần, kết thúc

        and t2, s2, t1       # Kiểm tra bit thấp nhất của s2 (s2 & 1)
        beqz t2, SKIP_ADD    # Nếu bit là 0, bỏ qua cộng
        add s0, s0, s1       # Nếu bit là 1, cộng s1 vào kết quả

SKIP_ADD:
        slli s1, s1, 1       # Dịch trái s1 (nhân đôi)
        srli s2, s2, 1       # Dịch phải s2 (chia đôi)
        beq s2, zero, ENDLOOP # Nếu s2 = 0, kết thúc sớm
        addi t0, t0, 1       # Tăng biến đếm
        j LOOP               # Quay lại vòng lặp

ENDLOOP:
```

- **Các giá trị khởi tạo:**
  + Khởi tạo giá trị cho s1, s2
  + Khởi tạo biến đếm t0 = 0
  + Khởi tạo giá trị tích s0 = 0
  + Khởi tạo giá trị hằng t1 = 1 và s3 = 32 là giá trị để so sánh và số vòng lặp
- Các bước thực hiện chương trình
  + Khởi tạo các giá trị cần thiết
  + Câu lênh bge t0, s3, ENDLOOP để kiểm tra điều kiện vòng lặp, nếu biến đếm = 32 thì nhảy đến ENDLOOP
  + Câu lệnh and t2, s2, t1 để kiểm tra bit thấp nhất của s2 có phải là 1 hay không
  + Câu lệnh beqz t2, SKIP_ADD để kiểm tra nếu biến thấp nhất của s2 là 0 thì nhảy sang thẻ SKIP_ADD và không thực hiện phép cộng
  + Câu lệnh add s0, s0, s1 để tính tổng của s0 và s1
  + Câu lệnh slli s1, s1, 1 để dịch trái s1 1 bit (tương tự nhân 2)
  + Câu lệnh srli s2, s2, 1 để dịch phải s2 1 bit (tương tự chia 2)
  + Câu lệnh beq s2, zero, ENDLOOP để kiểm tra xem nếu s2 = 0 thì nhảy sang ENDLOOP và kết thúc vòng lặp
  + Câu lệnh addi t0, t0, 1 để tăng biến đếm lên 1 đơn vị

+ Câu lệnh j LOOP để nhảy đến thẻ LOOP và tiếp tục thực hiện chương trình

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

+ Với giá trị của s1 = 6 và s2 = 8 ta có:

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	0x00000001
t2	7	0x00000001
s0	8	0x00000030
s1	9	0x00000060
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	0x00000000
s3	19	0x00000020
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		0x00400040

⇨ Sau khi thực hiện chương trình giá trị của s0 = 0x30 = 48 = 6 * 8 đúng với kết quả tính toán