

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 3

Họ và tên: Hoàng Văn Thắng

MSSV: 20235828

Assignment 1

Tạo project để thực hiện đoạn mã trong Home Assignment 1. Khởi tạo các biến cần thiết. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, kiểm tra sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy.

Nhập chương trình

```
1  # Laboratory Exercise 3, Home Assignment 1
2  .data
3      i: .word 5      # Khai báo biến i với giá trị 5
4      j: .word 3      # Khai báo biến j với giá trị 3
5      x: .word 0      # Khai báo biến x với giá trị 0
6      y: .word 0      # Khai báo biến y với giá trị 0
7      z: .word 0      # Khai báo biến z với giá trị 0
8  .text
9  start:
10     # TODO:
11     # Khởi tạo giá trị i vào thanh ghi s1
12     # Khởi tạo giá trị j vào thanh ghi s2
13     la t0, i          # Load address i in t0
14     lw s1, 0(t0)      # load value of i in s1
15     lw s2, 4(t0)      # load value of j in s2
16     # Khởi tạo x, y, z (t1, t2, t3)
17     li t1, 0 # x = 0
18     li t2, 0 # y = 0
19     li t3, 0 # z = 0
20     # Cách 1:
21     # blt s2, s1, else # if j < i then jump else
22     # Cách 2:
23     slt t0, s2, s1     # set t0 = 1 if j < i else clear t0 = 0
24     bne t0, zero, else # t0 != 0 means t0 = 1, jump else
25
26 then:
27     addi t1, t1, 1     # then part: x = x + 1
28     addi t3, zero, 1   # z = 1
29     j     endif        # skip else part
30 else:
31     addi t2, t2, -1    # begin else part: y = y - 1
32     add  t3, t3, t3     # z = 2 * z
33 endif:
```

Laboratory Exercise 3, Home Assignment 1

```
.data
    i: .word 5    # Khai báo biến i với giá trị 5
    j: .word 3    # Khai báo biến j với giá trị 3
    x: .word 0    # Khai báo biến x với giá trị 0
    y: .word 0    # Khai báo biến y với giá trị 0
    z: .word 0    # Khai báo biến z với giá trị 0

.text
start:
    # TODO:
    # Khởi tạo giá trị i vào thanh ghi s1
    # Khởi tạo giá trị j vào thanh ghi s2
    la t0, i # Load address i in t0
    lw s1, 0(t0) # load value of i in s1
    lw s2, 4(t0) # load value of j in s2
    # Khởi tạo x, y, z (t1, t2, t3)
    li t1, 0 # x = 0
    li t2, 0 # y = 0
    li t3, 0 # z = 0
    # Cách 1:
    # blt s2, s1, else # if j < i then jump else
    # Cách 2:
    slt t0, s2, s1    # set t0 = 1 if j < i else clear t0 = 0
    bne t0, zero, else # t0 != 0 means t0 = 1, jump else

then:
    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:
```

Nội dung các thanh ghi ở mỗi bước chạy

Ở trạng thái ban đầu:

- t0: Giá trị không xác định
- s1: Giá trị không xác định
- s2: Giá trị không xác định

- t1: Giá trị không xác định
- t2: Giá trị không xác định
- t3: Giá trị không xác định
- pc: Giá trị không xác định (trở đến lệnh la t0, i)
- Bộ nhớ:
 - Địa chỉ i: 5
 - Địa chỉ j: 3
 - Địa chỉ x, y, z: 0

Sau lệnh *la t0, i*

t0	5	0x10010000
----	---	------------

- t0: Chứa địa chỉ của biến i (ví dụ: 0x10010000).
- s1: Giá trị không thay đổi (vẫn không xác định).
- s2: Giá trị không thay đổi (vẫn không xác định).
- t1: Giá trị không thay đổi
- t2: Giá trị không thay đổi.
- t3: Giá trị không thay đổi.
- pc: Tăng lên (trở đến lệnh lw s1, 0(t0)).

Sau lệnh *lw s1, 0(t0)*

s1	9	0x00000005
----	---	------------

- t0: Giá trị không thay đổi (vẫn là địa chỉ của i).
- s1: 5 (giá trị của i).
- s2: Giá trị không thay đổi (vẫn không xác định).
- t1: Giá trị không thay đổi
- t2: Giá trị không thay đổi.
- t3: Giá trị không thay đổi.
- pc: Tăng lên (trở đến lệnh lw s2, 4(t0)).

Sau lệnh *lw s2, 4(t0)*

s2	18	0x00000003
----	----	------------

- t0: Giá trị không thay đổi (vẫn là địa chỉ của i).
- s1: Giá trị không thay đổi (vẫn là 5).
- s2: 3 (giá trị của j).
- t1: Giá trị không thay đổi
- t2: Giá trị không thay đổi.
- t3: Giá trị không thay đổi.
- pc: Tăng lên (trở đến lệnh li t1, 0).

Sau lệnh *li t1, 0*

t1	6	0x00000000
----	---	------------

- t0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: 0 (giá trị của x).
- t2: Giá trị không thay đổi.
- t3: Giá trị không thay đổi.
- pc: Tăng lên.

Sau lệnh *li t2, 0*

t2	7	0x00000000
----	---	------------

- t0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: 0 (giá trị của y).
- t3: Giá trị không thay đổi.

- pc: Tăng lên.

Sau lệnh *li t3, 0*

t3	28	0x00000000
----	----	------------

- t0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- t3: 0 (giá trị của z).
- pc: Tăng lên.

Sau lệnh *slt t0, s2, s1*

t0	5	0x00000001
----	---	------------

- t0: 1 (vì s2 (3) < s1 (5)).
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- t3: Giá trị không thay đổi.
- pc: Tăng lên.

Sau lệnh *bne t0, zero, else*

- t0: Giá trị không thay đổi (vẫn là 1).
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.

- t3: Giá trị không thay đổi.
- pc: Thay đổi, trở đến lệnh tại nhãn else (vì t0 khác 0).

Sau lệnh *addi t2, t2, -1* (trong nhánh else)

t2	7	0xffffffff
----	---	------------

- t0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: $-1(y = y - 1, y=0)$.
- t3: Giá trị không thay đổi.
- pc: Tăng lên.

Sau lệnh *add t3, t3, t3* (trong nhánh else)

t3	28	0x00000000
----	----	------------

- t0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- t3: 0 (vì $0 * 2 = 0$).
- pc: Tăng lên (trở đến endif:).

Sau *endif*:

Kết thúc chương trình

Sự thay đổi của bộ nhớ: Bộ nhớ không thay đổi trong quá trình chạy chương trình, do chỉ thao tác với các thanh ghi và đã nạp dữ liệu từ bộ nhớ của biến i, j vào

thanh ghi s1, s2; ở cuối chương trình cũng không có các thao tác lưu giá trị biến vào bộ nhớ => bộ nhớ chỉ thay đổi ở bước khởi tạo biến, sau đó giữ nguyên đến cuối.

Assignment 2

Tạo project để thực hiện đoạn mã trong Home Assignment 2. Khởi tạo các biến cần thiết và mảng A. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy. Thay bộ giá trị khác để kiểm tra sự đúng đắn của chương trình.

Nhập chương trình:

```
1  # Laboratory 3, Home Assignment 2
2  .data
3      A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
4  .text
5      # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
6      li s1, 0          # i = 0
7      la s2, A          # load address of array A in s2
8      li s3, 9          # s3 = 9 (number of element in array A)
9      li s4, 1          # s4 = 1 (step = 1)
10     li s5, 0          # sum = 0
11 loop:
12     bge s1, s3, endloop # if i >= n then end loop
13     add t1, s1, s1      # t1 = 2 * s1
14     add t1, t1, t1      # t1 = 4 * s1 => t1 = 4*i
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              # go to loop
20 endloop:
```

```
# Laboratory 3, Home Assignment 2
```

```
.data
```

```
    A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
```

```
.text
```

```
    # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
```

```
    li s1, 0          # i = 0
```

```
    la s2, A          # load address of array A in s2
```

```
    li s3, 9          # s3 = 9 (number of element in array A)
```

```
    li s4, 1          # s4 = 1 (step = 1)
```

```

        li s5, 0      # sum = 0
loop:
        bge s1, s3, endloop # if i >= n then end loop
        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)        # load value of A[i] in t0
        add s5, s5, t0      # sum = sum + A[i]
        add s1, s1, s4      # i = i + step
        j loop              # go to loop
endloop:

```

Sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy

- Giá trị ban đầu:

Thanh ghi:

- s1 (i): Giá trị không xác định.
- s2 (base address of A): Giá trị không xác định.
- s3 (n): Giá trị không xác định.
- s4 (step): Giá trị không xác định.
- s5 (sum): Giá trị không xác định.
- t0: Giá trị không xác định.
- t1: Giá trị không xác định.
- pc: Giá trị không xác định (trở đến lệnh la s2, A).

Bộ nhớ:

- A[0]: 1
- A[1]: 3
- A[2]: 2
- A[3]: 5
- A[4]: 4

- A[5]: 7
- A[6]: 8
- A[7]: 9
- A[8]: 6
- Sau lệnh *li s1, 0*

s1	9	0x00000000
----	---	------------

Thanh ghi:

- s1: 0.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

- Sau lệnh *la s2, A*

s2	18	0x10010000
----	----	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Địa chỉ của mảng A (0x10010000).
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.

- t1: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

- Sau lệnh *li s3, 9*

s3	19	0x00000009
----	----	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: 9.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

- Sau lệnh *li s4, 1*

s4	20	0x00000001
----	----	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: 1.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.

- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

- Sau lệnh *li s5, 0*

s5	21	0x00000000
----	----	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: 0.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- pc: Tăng lên (trở đến nhãn loop).

Bộ nhớ: Không thay đổi.

- **Bắt đầu vòng lặp loop (s1 = 0):**

Sau lệnh *bge s1, s3, endloop* (lần đầu tiên):

Thanh ghi:

- s1: Giá trị không thay đổi (vẫn là 0).
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- pc: Không nhảy (vì $0 < 9$), trở đến lệnh *add t1, s1, s1*.

Bộ nhớ: Không thay đổi.

Sau lệnh *add t1, s1, s1* (lần 1):

t1	6	0x00000000
----	---	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: 0 (vì $0 * 2 = 0$).
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *add t1, t1, t1* (lần 1):

t1	6	0x00000000
----	---	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: 0 (vì $0 * 2 = 0$).
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *add t1, t1, s2* (lần 1):

t1	6	0x10010000
----	---	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Địa chỉ của A[0] (0x10010000).
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lw t0, 0(t1) (lần 1):

t0	5	0x00000001
----	---	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: 1 (giá trị của A[0]).
- t1: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau add s5, s5, t0 (lần 1):

s5	21	0x00000001
----	----	------------

Thanh ghi:

- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: 1 (vì $0 + 1 = 1$).
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau add s1, s1, s4 (lần 1):

s1	9	0x00000001
----	---	------------

Thanh ghi:

- s1: 1 (vì $0 + 1 = 1$).
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- s4: Giá trị không thay đổi.
- s5: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- PC: Tăng lên.
- **Bộ nhớ:** Không thay đổi.

Sau j loop (lần 1):

Thanh ghi: Không có thanh ghi nào thay đổi giá trị *trực tiếp*.

Bộ nhớ: Không thay đổi.

- pc: Thay đổi, trở trở lại nhãn loop.
- Vòng lặp tiếp tục lần 2

Vòng lặp sẽ tiếp tục thực hiện, với s1 (i) tăng dần lên 1, s5 (sum) cộng dồn giá trị của các phần tử trong mảng A, và t1 lần lượt chứa địa chỉ của A[1], A[2], ... cho đến khi s1 (i) bằng s3 (n = 9).

Vòng lặp cuối cùng (khi s1 = 8):

Sau add s5, s5, t0 (lần cuối, s1 = 8):

- s5 sẽ chứa tổng của các phần tử từ A[0] đến A[7] cộng thêm A[8]
- Tổng từ A[0] đến A[7] = 30, s5 = 30 + A[8] = 30 + 6 = 36

Sau add s1, s1, s4 (lần cuối, s1 = 8):

- s1: 9 (vì 8+1=9).

Sau j loop (lần cuối):

- pc: Trở đến loop.

Tại loop (lần tiếp theo, s1 = 9)

Sau bge s1, s3, endloop (khi s1 = 9):

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000006
t1	6	0x10010020
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000009
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	0x00000009
s4	20	0x00000001
s5	21	0x0000002d
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

Thanh ghi:

- s1: 9.
- s2: Địa chỉ của mảng A.
- s3: 9.
- s4: 1.
- s5: 45 (Tổng của tất cả các phần tử trong mảng A).
- t0: Giá trị của A[8] (từ vòng lặp trước).
- t1: Địa chỉ của A[8] (từ vòng lặp trước).
- pc: Nhảy đến endloop (vì $9 \geq 9$).

Bộ nhớ: Không thay đổi.

Sau khi đến endloop:

- Các thanh ghi, bộ nhớ giữ nguyên giá trị (Bộ nhớ giữ nguyên giá trị do chỉ thao tác với thanh ghi, không có các lệnh tác động vào bộ nhớ trực tiếp)

Như vậy, sau khi vòng lặp kết thúc, thanh ghi s5 sẽ chứa tổng của tất cả các phần tử trong mảng A ($1 + 3 + 2 + 5 + 4 + 7 + 8 + 9 + 6 = 45$).

s5	21	0x0000002d
----	----	------------

Thay bộ giá trị khác để kiểm tra sự đúng đắn của chương trình

```
1  # Laboratory 3, Home Assignment 2
2  .data
3      A: .word 5, 7, 8, 9, 10, 15, 14, 16, 8
4  .text
5      # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
6      li s1, 0          # i = 0
7      la s2, A          # load address of array A in s2
8      li s3, 9          # s3 = 9 (number of element in array A)
9      li s4, 1          # s4 = 1 (step = 1)
10     li s5, 0          # sum = 0
11 loop:
12     bge s1, s3, endloop # if i >= n then end loop
13     add t1, s1, s1      # t1 = 2 * s1
14     add t1, t1, t1      # t1 = 4 * s1 => t1 = 4*i
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              # go to loop
20 endloop:
```


Sau khi chạy chương trình ta được tổng $s5 = 5c_{16} = 92_{10}$ (kết quả đúng)

s5	21	0x0000005c
----	----	------------

Assignment 3

Tạo project để thực hiện đoạn mã trong Home Assignment 3. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở từng bước chạy. Thay đổi bộ giá trị và chạy lại chương trình một vài lần để kiểm tra tất cả các trường hợp

Nhập chương trình:

```
1 # Laboratory Exercise 3, Home Assignment 3
2 .data:
3     test: .word 1 # Khai báo biến test với giá trị 1
4 .text
5     la s0, test    # Nạp địa chỉ biến test vào s0
6     lw s1, 0(s0)   # Nạp giá trị của biến test vào s1
7     li s2, 10      # a = 10
8     li s3, 5       # b = 5
9     li t0, 0       # Nạp giá trị cần kiểm tra
10    li t1, 1       # Nạp giá trị cần kiểm tra
11    li t2, 2       # Nạp giá trị cần kiểm tra
12    beq s1, t0, case_0
13    beq s1, t1, case_1
14    beq s1, t2, case_2
15    j default
16 case_0:
17     addi s2, s2, 1 # a = a + 1
18     j continue
19 case_1:
20     sub s2, s2, t1 # a = a - 1
21     j continue
22 case_2:
23     add s3, s3, s3 # b = 2 * b
24     j continue
25 default:
26 continue:
```

```
# Laboratory Exercise 3, Home Assignment 3
.data:
    test: .word 1 # Khai báo biến test với giá trị 1
.text
    la s0, test    # Nạp địa chỉ biến test vào s0
    lw s1, 0(s0)   # Nạp giá trị của biến test vào s1
    li s2, 10      # a = 10
```

```

        li s3, 5      # b = 5
        li t0, 0      # Nạp giá trị cần kiểm tra
        li t1, 1      # Nạp giá trị cần kiểm tra
        li t2, 2      # Nạp giá trị cần kiểm tra
        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:

```

Sự thay đổi của bộ nhớ và nội dung các thanh ghi ở từng bước chạy

Giá trị ban đầu:

Thanh ghi:

- s0: Giá trị không xác định.
- s1: Giá trị không xác định.
- s2: Giá trị không xác định.
- s3: Giá trị không xác định.
- t0: Giá trị không xác định.
- t1: Giá trị không xác định.
- t2: Giá trị không xác định.
- pc: Giá trị không xác định (trở đến lệnh la s0, test).

Bộ nhớ:

- Địa chỉ test: 1 (giá trị khởi tạo).

Sau lệnh *la s0, test*

s0	8	0x10010000
----	---	------------

Thanh ghi:

- s0: Địa chỉ của biến test (0x10010000).
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *lw s1, 0(s0)*:

s1	9	0x00000001
----	---	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: 1 (giá trị của biến test).
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *li s2, 10*

s2	18	0x0000000a
----	----	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: 10 (giá trị khởi tạo của a).
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *li s3, 5*

s3	19	0x00000005
----	----	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: 5 (giá trị khởi tạo của b).
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *li t0, 0*

t0	5	0x00000000
----	---	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: 0.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *li t1, 1*

t1	6	0x00000001
----	---	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: 1.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau lệnh *li t2, 2*

t2	7	0x00000002
----	---	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: 2.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau beq s1, t0, case_0: Không thay đổi bất kì thứ gì

Sau beq s1, t1, case_1:

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: Giá trị không thay đổi.
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Nhảy đến case_1 (bỏ qua case_0).

Bộ nhớ: Không thay đổi.

Sau sub s2, s2, t1 (trong case_1):

s2	18	0x00000009
----	----	------------

Thanh ghi:

- s0: Giá trị không thay đổi.
- s1: Giá trị không thay đổi.
- s2: 9 (vì $10 - 1 = 9$).
- s3: Giá trị không thay đổi.
- t0: Giá trị không thay đổi.
- t1: Giá trị không thay đổi.
- t2: Giá trị không thay đổi.
- pc: Tăng lên.

Bộ nhớ: Không thay đổi.

Sau j continue (trong case_1):

Thanh ghi: Giá trị các thanh ghi không thay đổi

Bộ nhớ: Không thay đổi.

- pc: Nhảy đến continue.

Sau continue:

Thanh ghi: Các giá trị giữ nguyên

Bộ nhớ: Không thay đổi.

- Chương trình kết thúc.

Như vậy, sau khi thực thi, thanh ghi s2 (biến a) sẽ có giá trị 9, và các thanh ghi/biến khác giữ nguyên giá trị như đã mô tả.

Thay đổi bộ giá trị và chạy lại chương trình một vài lần để kiểm tra tất cả các trường hợp.

EditExecute

Assignment1.asmAssignment2.asmAssignment3.asm

```
1 # Laboratory Exercise 3, Home Assignment 3
2 .data:
3     testr: word 1 # Khai báo biến test với giá trị 1
4     .text
5     la s0, test # Map địa chỉ biến test vào s0
6     lw s1, 0(s0) # Map giá trị của biến test vào s1
7     li s2, 20 # a = 20
8     li s3, 15 # b = 15
9     li t0, 0 # Map giá trị của biến tra
10    li t1, 1 # Map giá trị của biến tra
11    li t2, 2 # Map giá trị của biến tra
12    beq s1, t0, case_0
13    beq s1, t1, case_1
14    beq s1, t2, case_2
15    } default
16
17    addi s2, s2, 1 # a = a + 1
18    } continue
19
20    case_1:
21    sub s2, s2, t1 # a = a - 1
22    } continue
23
24    case_2:
25    add s3, s3, s3 # b = 2 * b
26    } continue
27
28    default:
29    continue:
30
```

RegistersFloating PointControl and Status

Name	Number	Value
\$zero	0	0x00000000
\$ra	1	0x00000000
\$fp	2	0x7ffffefc
\$tp	3	0x10000000
\$t0	4	0x00000000
\$t1	5	0x00000000
\$t2	6	0x00000001
\$t3	7	0x00000002
\$t4	8	0x00000000
\$t5	9	0x00000001
\$t6	10	0x00000000
\$t7	11	0x00000000
\$t8	12	0x00000000
\$t9	13	0x00000000
\$s0	14	0x00000000
\$s1	15	0x00000000
\$s2	16	0x00000013
\$s3	17	0x0000000f
\$a0	18	0x00000000
\$a1	19	0x00000000
\$a2	20	0x00000000
\$a3	21	0x00000000
\$a4	22	0x00000000
\$a5	23	0x00000000
\$a6	24	0x00000000
\$a7	25	0x00000000
\$a8	26	0x00000000
\$a9	27	0x00000000
\$t0	28	0x00000000
\$t1	29	0x00000000
\$t2	30	0x00000000
\$t3	31	0x00000000
\$PC		0x00400040

Edit		Execute		Registers		Floating Point	Control and Status
		Assignment1.asm	Assignment2.asm	Assignment3.asm	Name	Number	Value
1	# Laboratory Exercise 3, Home Assignment 3				\$zero	0	0x00000000
2	.data:				\$ra	1	0x00000000
3	test: word 4 # Khai báo biến test với giá trị 4				\$fp	2	0x7ffffefc
4	.text				\$tp	3	0x10000000
5	la s0, test # Map địa chỉ biến test vào s0				\$t0	4	0x00000000
6	lw s1, 0(s0) # Map giá trị của biến test vào s1				\$t1	5	0x00000000
7	li s2, 21 # a = 21				\$t2	6	0x00000001
8	li s3, 2 # b = 2				\$t3	7	0x00000004
9	li t0, 0 # Map giá trị của biến tra				\$t4	8	0x10010000
10	li t1, 1 # Map giá trị của biến tra				\$t5	9	0x00000004
11	li t2, 4 # Map giá trị của biến tra				\$t6	10	0x00000000
12	beq s1, t0, case_0				\$t7	11	0x00000000
13	beq s1, t1, case_1				\$t8	12	0x00000000
14	beq s1, t2, case_2				\$t9	13	0x00000000
15	default				\$a0	14	0x00000000
16					\$a1	15	0x00000000
17	addi s2, s2, 1 # a = a + 1				\$a2	16	0x00000000
18	continue				\$a3	17	0x00000000
19	case_1:				\$a4	18	0x00000015
20	sub s2, s2, t1 # a = a - 1				\$a5	19	0x00000004
21	continue				\$a6	20	0x00000000
22	case_2:				\$a7	21	0x00000000
23	add s3, s3, s3 # b = 2 * b				\$a8	22	0x00000000
24	continue				\$a9	23	0x00000000
25	default:				\$s0	24	0x00000000
26	continue				\$s1	25	0x00000000
					\$s2	26	0x00000000
					\$s3	27	0x00000000
					\$t0	28	0x00000000
					\$t1	29	0x00000000
					\$t2	30	0x00000000
					\$t3	31	0x00000000
					\$PC		0x00400040

Ta thấy kết quả đều đúng trong những lần thay đổi bộ giá trị

Assignment 4

Lần lượt thay thế điều kiện rẽ nhánh trong Home Assignment 1 bằng các điều kiện sau đây:

a. $i < j$

Nhập chương trình:


```

1  # Laboratory Exercise 3, Assignment 4a
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 5      # i = 5
6      li s2, 5      # j = 5
7      li t1, 1      # t1 = 1
8      li t2, 2      # t2 = 2
9      li t3, 3      # t3 = 3
10     # Điều kiện rẽ nhánh i < j
11     bge s1, s2, else
12 then:
13     addi t1, t1, 1    # then part: x = x + 1
14     addi t3, zero, 1  # z = 1
15     j     endif      # skip else part
16 else:
17     addi t2, t2, -1    # begin else part: y = y - 1
18     add  t3, t3, t3    # z = 2 * z
19 endif:

```

Laboratory Exercise 3, Assignment 4a

.text

start:

 # Khởi tạo i, j, x, y, z

 li s1, 5 # i = 5

 li s2, 5 # j = 5

 li t1, 1 # t1 = 1

 li t2, 2 # t2 = 2

 li t3, 3 # t3 = 3

 # Điều kiện rẽ nhánh i < j

 bge s1, s2, else

then:

 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:

Với $i = j = 5$

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000001
t2	7	0x00000001
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	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	0x00000006
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

Ta có kết quả sau:

$$y = y - 1 = 2 - 1 = 1$$

$$z = 2 * z = 2 * 3 = 6$$

Với $i = 4, j = 5$ ($i < j$)

```

1  # Laboratory Exercise 3, Assignment 4a
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 4      # i = 4
6      li s2, 5      # j = 5
7      li t1, 1      # x = 1
8      li t2, 2      # y = 2
9      li t3, 3      # z = 3
10     # Điều kiện rẽ nhánh i < j
11     bge s1, s2, else
12 then:
13     addi t1, t1, 1    # then part: x = x + 1
14     addi t3, zero, 1  # z = 1
15     j     endif      # skip else part
16 else:
17     addi t2, t2, -1    # begin else part: y = y - 1
18     add  t3, t3, t3    # z = 2 * z
19 endif:

```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
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	0x00000001
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

Ta có kết quả sau:

$x = x + 1 = 2$

$z = 1$

Với $i > j$ (tương tự với khi $i = j$)

b. $i \geq j$

Nhập chương trình:

```
1  # Laboratory Exercise 3, Assignment 4b
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 10      # i = 10
6      li s2, 5       # j = 5
7      li t1, 1       # x = 1
8      li t2, 2       # y = 2
9      li t3, 3       # z = 3
10     # Điều kiện rẽ nhánh i >= j
11     blt s1, s2, else
12 then:
13     addi t1, t1, 1   # then part: x = x + 1
14     addi t3, zero, 1 # z = 1
15     j     endif     # skip else part
16 else:
17     addi t2, t2, -1  # begin else part: y = y - 1
18     add  t3, t3, t3  # z = 2 * z
19 endif:
20
```

Laboratory Exercise 3, Assignment 4b

.text

start:

 # Khởi tạo i, j, x, y, z

 li s1, 10 # i = 10

 li s2, 5 # j = 5

 li t1, 1 # x = 1

 li t2, 2 # y = 2

 li t3, 3 # z = 3

 # Điều kiện rẽ nhánh i >= j

 blt s1, s2, else

then:

 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:

```

Với $i \geq j$

```

1  # Laboratory Exercise 3, Assignment 4b
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 10          # i = 10
6      li s2, 5           # j = 5
7      li t1, 1           # x = 1
8      li t2, 2           # y = 2
9      li t3, 3           # z = 3
10     # Điều kiện rẽ nhánh i >= j
11     blt s1, s2, else
12 then:
13     addi t1, t1, 1      # then part: x = x + 1
14     addi t3, zero, 1    # z = 1
15     j      endif      # skip else part
16 else:
17     addi t2, t2, -1     # begin else part: y = y - 1
18     add  t3, t3, t3     # z = 2 * z
19 endif:

```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000002
s0	8	0x00000000
s1	9	0x0000000a
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	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

Ta có kết quả sau:

$$x = x + 1 = 2$$

$$z = 1$$

Với $i < j$

```

1  # Laboratory Exercise 3, Assignment 4b
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 1          # i = 1
6      li s2, 5          # j = 5
7      li t1, 1          # x = 1
8      li t2, 2          # y = 2
9      li t3, 3          # z = 3
10     # Điều kiện rẽ nhánh i >= j
11     blt s1, s2, else
12 then:
13     addi    t1, t1, 1    # then part: x = x + 1
14     addi    t3, zero, 1  # z = 1
15     j      endif        # skip else part
16 else:
17     addi    t2, t2, -1   # begin else part: y = y - 1
18     add     t3, t3, t3    # z = 2 * z
19 endif:

```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000001
t2	7	0x00000001
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	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	0x00000006
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

Ta có kết quả sau:

$$y = y - 1 = 2 - 1 = 1$$

$$z = 2 * z = 3 * 2 = 6$$

$$c. \ i + j \leq 0$$

Nhập chương trình:


```

1  # Laboratory Exercise 3, Assignment 4c
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 10      # i = 10
6      li s2, 5       # j = 5
7      li t1, 1       # x = 1
8      li t2, 2       # y = 2
9      li t3, 3       # z = 3
10     # Điều kiện rẽ nhánh i + j <= 0
11     add t0, s1, s2  # sum = i + j
12     bgtz t0, else   # if i + j <= 0, nhảy đến else
13 then:
14     addi t1, t1, 1   # then part: x = x + 1
15     addi t3, zero, 1 # z = 1
16     j     endif      # skip else part
17 else:
18     addi t2, t2, -1   # begin else part: y = y - 1
19     add  t3, t3, t3    # z = 2 * z
20 endif:

```

Laboratory Exercise 3, Assignment 4c

.text

start:

 # Khởi tạo i, j, x, y, z

 li s1, 10 # i = 10

 li s2, 5 # j = 5

 li t1, 1 # x = 1

 li t2, 2 # y = 2

 li t3, 3 # z = 3

 # Điều kiện rẽ nhánh i + j <= 0

 add t0, s1, s2 # sum = i + j

 bgtz t0, else # if i + j <= 0, nhảy đến else

then:

 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:

Với $i + j > 0$

```

1  # Laboratory Exercise 3, Assignment 4c
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, 10      # i = 10
6      li s2, 5       # j = 5
7      li t1, 1       # x = 1
8      li t2, 2       # y = 2
9      li t3, 3       # z = 3
10     # Điều kiện rẽ nhánh i + j <= 0
11     add t0, s1, s2  # sum = i + j
12     bgtz t0, else   # if i + j <= 0, nhảy đến else
13 then:
14     addi    t1, t1, 1      # then part: x = x + 1
15     addi    t3, zero, 1    # z = 1
16     j      endif         # skip else part
17 else:
18     addi    t2, t2, -1     # begin else part: y = y - 1
19     add     t3, t3, t3     # z = 2 * z
20 endif:

```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x0000000f
t1	6	0x00000001
t2	7	0x00000001
s0	8	0x00000000
s1	9	0x0000000a
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	0x00000006
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400034

$y = y - 1 = 1$

$z = 2 * z = 6$

Với $i + j \leq 0$

```

1  # Laboratory Exercise 3, Assignment 4c
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z
5      li s1, -5      # i = -5
6      li s2, 5       # j = 5
7      li t1, 1       # x = 1
8      li t2, 2       # y = 2
9      li t3, 3       # z = 3
10     # Điều kiện rẽ nhánh i + j <= 0
11     add t0, s1, s2  # sum = i + j
12     bgtz t0, else  # if i + j <= 0, nhảy đến else
13 then:
14     addi t1, t1, 1   # then part: x = x + 1
15     addi t3, zero, 1 # z = 1
16     j     endif     # skip else part
17 else:
18     addi t2, t2, -1  # begin else part: y = y - 1
19     add t3, t3, t3   # z = 2 * z
20 endif:

```

zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000002
t2	7	0x00000002
s0	8	0x00000000
s1	9	0xffffffffb
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	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400034

$x = x + 1 = 2$

$z = 1$

d. $i + j > m + n$

Nhập chương trình:

```

1  # Laboratory Exercise 3, Assignment 4d
2  .text
3  start:
4      # Khởi tạo i, j, x, y, z, m, n
5      li s1, 10      # i = 10
6      li s2, 5       # j = 5
7      li t1, 1       # x = 1
8      li t2, 2       # y = 2
9      li t3, 3       # z = 3
10     li s4, 7       # m = 7
11     li s5, 3       # n = 3
12     # Điều kiện rẽ nhánh i + j > m + n
13     add t0, s1, s2  # sum = i + j
14     add t4, s4, s5  # sum2 = m + n
15     bge t4, t0, else    # if i + j <= m + n, nhảy đến else
16 then:
17     addi    t1, t1, 1    # then part: x = x + 1
18     addi    t3, zero, 1  # z = 1
19     j      endif        # skip else part
20 else:
21     addi    t2, t2, -1    # begin else part: y = y - 1
22     add     t3, t3, t3    # z = 2 * z
23 endif:

```

Laboratory Exercise 3, Assignment 4d

.text

start:

 # Khởi tạo i, j, x, y, z, m, n

 li s1, 10 # i = 10

 li s2, 5 # j = 5

 li t1, 1 # x = 1

 li t2, 2 # y = 2

 li t3, 3 # z = 3

 li s4, 7 # m = 7

 li s5, 3 # n = 3

 # Điều kiện rẽ nhánh i + j > m + n

 add t0, s1, s2 # sum = i + j

 add t4, s4, s5 # sum2 = m + n

 bge t4, t0, else # if i + j <= m + n, nhảy đến else

then:

 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:

```

Với $i + j > m + n$

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x0000000f
t1	6	0x00000002
t2	7	0x00000002
s0	8	0x00000000
s1	9	0x0000000a
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	0x00000007
s5	21	0x00000003
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x0000000a
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

$x = x + 1 = 2$

$z = 1$

Với $i + j \leq m + n$

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000007
t1	6	0x00000001
t2	7	0x00000001
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	0x00000005
s3	19	0x00000000
s4	20	0x00000007
s5	21	0x00000003
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000006
t4	29	0x0000000a
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

$y = y - 1 = 1$

$z = 2 * z = 6$

Assignment 5

Lần lượt thay thế điều kiện nhảy (thoát khỏi vòng lặp) trong Home Assignment 2 bằng các điều kiện sau đây: (Cần thiết lập giá trị các phần tử của mảng để điều kiện có thể được thỏa mãn.)

a. $i > n$

Nhập chương trình:


```

1  # Laboratory 3, Assignment 5a
2  .data
3      A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
4  .text
5      # TODO: Khởi tạo giá trị các thanh ghi
6      li s1, 0          # i = 0
7      la s2, A           # load address of array A in s2
8      li s3, 9           # s3 = 9 (number of element in array A)
9      li s4, 1           # s4 = 1 (step = 1)
10     li s5, 0           # sum = 0
11 loop:
12     blt s3, s1, endloop # if i > n then end loop
13     add t1, s1, s1       # t1 = 2 * s1
14     add t1, t1, t1       # t1 = 4 * s1 => t1 = 4*i
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              # go to loop
20 endloop:

```

Laboratory 3, Assignment 5a

.data

A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6

.text

TODO: Khởi tạo giá trị các thanh ghi

li s1, 0 # i = 0

la s2, A # load address of array A in s2

li s3, 9 # s3 = 9 (number of element in array A)

li s4, 1 # s4 = 1 (step = 1)

li s5, 0 # sum = 0

loop:

blt s3, s1, endloop # if i > n then end loop

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) # load value of A[i] in t0

add s5, s5, t0 # sum = sum + A[i]

add s1, s1, s4 # i = i + step

j loop # go to loop

endloop:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x10010024
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x0000000a
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	0x00000009
s4	20	0x00000001
s5	21	0x0000002d
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

Giải thích kết quả chạy:

- Thay lệnh bge ban đầu ở bài 2 thành blt s3, s1, endloop.
 - Nếu $n < i$, kết thúc vòng lặp; ngược lại thực hiện tiếp chương trình.
- b. $sum < 0$

Nhập chương trình:

```

1  # Laboratory 3, Assignment 5b
2  .data
3      A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
4  .text
5      # TODO: Khởi tạo giá trị các thanh ghi
6      li s1, 0          # i = 0
7      la s2, A          # load address of array A in s2
8      li s3, 9          # s3 = 9 (number of element in array A)
9      li s4, 1          # s4 = 1 (step = 1)
10     li s5, 0          # sum = 0
11 loop:
12     blt s5, zero, endloop # if sum < 0 then end loop
13     add t1, s1, s1        # t1 = 2 * s1
14     add t1, t1, t1        # t1 = 4 * s1 => t1 = 4*i
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               # go to loop
20 endloop:

```

Laboratory 3, Assignment 5b

.data

A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6

.text

TODO: Khởi tạo giá trị các thanh ghi

li s1, 0 # i = 0

la s2, A # load address of array A in s2

li s3, 9 # s3 = 9 (number of element in array A)

li s4, 1 # s4 = 1 (step = 1)

li s5, 0 # sum = 0

loop:

blt s5, zero, endloop # if sum < 0 then end loop

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) # load value of A[i] in t0

add s5, s5, t0 # sum = sum + A[i]

add s1, s1, s4 # i = i + step

j loop # go to loop

endloop:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
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	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		0x00400000

Giải thích kết quả chạy:

- Thay lệnh bge ban đầu ở bài 2 thành blt s5, zero, endloop.
- Nếu s5 (sum) < 0, kết thúc vòng lặp; ngược lại thực hiện tiếp chương trình.

c. $A[i] == 0$

Nhập chương trình:

```

1  # Laboratory 3, Assignment 5c
2  .data
3      A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
4  .text
5      # TODO: Khởi tạo giá trị các thanh ghi
6      li s1, 0          # i = 0
7      la s2, A          # load address of array A in s2
8      li s3, 9          # s3 = 9 (number of element in array A)
9      li s4, 1          # s4 = 1 (step = 1)
10     li s5, 0          # sum = 0
11 loop:
12     beq t0, zero, endloop # if A[i] == 0 then end loop
13     add t1, s1, s1        # t1 = 2 * s1
14     add t1, t1, t1        # t1 = 4 * s1 => t1 = 4*i
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               # go to loop
20 endloop:

```

Laboratory 3, Assignment 5c

.data

A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6

.text

TODO: Khởi tạo giá trị các thanh ghi

li s1, 0 # i = 0

la s2, A # load address of array A in s2

li s3, 9 # s3 = 9 (number of element in array A)

li s4, 1 # s4 = 1 (step = 1)

li s5, 0 # sum = 0

loop:

beq t0, zero, endloop # if A[i] == 0 then end loop

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) # load value of A[i] in t0

add s5, s5, t0 # sum = sum + A[i]

add s1, s1, s4 # i = i + step

j loop # go to loop

endloop:

zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffc
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	0x00000009
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		0x0040003c

Giải thích kết quả chạy:

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) # load value of A[i] in t0

- 4 câu lệnh để load giá trị của A[i] vào t0; hai câu lệnh add ban đầu là để cập nhật địa chỉ của A[i] sau mỗi lần lặp, vì 1 word có giá trị (4 byte).
- Nếu A[i] = 0, nhảy đến nhãn endloop, ngược lại thực hiện tiếp chương trình.

Assignment 6

Tạo project để thực hiện chương trình sau:

Tìm phần tử có giá trị tuyệt đối lớn nhất từ một danh sách các số nguyên 32-bit.
Giả sử danh sách số nguyên được lưu trong một mảng biết trước số phần tử.

```
1  # Laboratory 3, Assignment 6
2  .data
3      array: .word 1, -3, 2, -5, 4, 7, -8, -9, -21
4  .text
5  start:
6      la t0, array          # t0: &A
7      li t1, 9              # t1: n (number element of Array)
8      li t2, 0              # t2: i = 0
9      lw t3, 0(t0)          # t3: max_abs = A[0]
10     li t6, -2147483648     # t6: -2^31
11     mv a1, t3              # a1: max_val = A[0] (giữ nguyên dấu)
12     bltz t3, neg_change   # Nếu A[0] < 0, đảo dấu
13     j loop
14 neg_change:
15     sub t3, zero, t3      # max_abs = -A[0]
16 loop:
17     bge t2, t1, endloop   # Thoát nếu i >= n
18     add t4, t2, t2        # t4 = i * 2
19     add t4, t4, t4        # t4 = i * 4
20     add t4, t4, t0        # t4 = &A[i]
21     lw t5, 0(t4)          # t5 = A[i]
22
23     beq t5, t6, set_max   # Nếu A[i] == -2^31, dừng ngay lập tức
24     bltz t5, neg_val     # Nếu A[i] < 0, đảo dấu
25     j compare
26 neg_val:
27     sub t5, zero, t5      # t5 = 0 - t5
28 compare:
29     blt t5, t3, increment # Nếu max_abs >= abs(A[i]), bỏ qua
30     addi t3, t5, 0        # max_abs = abs(A[i])
31     mv a1, t5             # max_val = A[i] (giữ nguyên dấu)
32 increment:
33     addi t2, t2, 1        # i++
34     j loop
35 set_max:
36     mv a1, t6             # max_val = -2^31
37     j endloop
38 endloop:
39     # a1 chứa phần tử có giá trị tuyệt đối lớn nhất, giữ nguyên dấu
```

Laboratory 3, Assignment 6

.data

```

array: .word 1, -3, 2, -5, 4, 7, -8, -9, -21
.text
start:
    la t0, array      # t0: &A
    li t1, 9           # t1: n (number element of Array)
    li t2, 0           # t2: i = 0
    lw t3, 0(t0)       # t3: max_abs = A[0]
    li t6, -2147483648 # t6: -2^31
    mv a1, t3          # a1: max_val = A[0] (giữ nguyên dấu)
    bltz t3, neg_change # Nếu A[0] < 0, đảo dấu
    j loop
neg_change:
    sub t3, zero, t3   # max_abs = -A[0]
loop:
    bge t2, t1, endloop # Thoát nếu i >= n
    add t4, t2, t2
    add t4, t4, t4      # t4 = i * 4
    add t4, t4, t0      # t4 = &A[i]
    lw t5, 0(t4)        # t5 = A[i]

    beq t5, t6, set_max # Nếu A[i] == -2^31, dừng ngay lập tức
    bltz t5, neg_val    # Nếu A[i] < 0, đảo dấu
    j compare
neg_val:
    sub t5, zero, t5    # t5 = 0 - t5
compare:
    blt t5, t3, increment # Nếu max_abs >= abs(A[i]), bỏ qua
    addi t3, t3, 0      # max_abs = abs(A[i])
    mv a1, t5          # max_val = A[i] (giữ nguyên dấu)
increment:
    addi t2, t2, 1      # i++
    j loop
set_max:
    mv a1, t6          # max_val = -2^31
    j endloop
endloop:
    # a1 chứa phần tử có giá trị tuyệt đối lớn nhất, giữ nguyên dấu

```

Quan sát kết quả chạy:

Với trường hợp mảng số không có phần tử -2^{31}

zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x10010000
t1	6	0x00000009
t2	7	0x00000009
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000015
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	0x00000015
t4	29	0x10010020
t5	30	0x00000015
t6	31	0x80000000
pc		0x00400070

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0xffffffffd	0x00000002	0xffffffffb	0x00000004	0x00000007	0xffffffff8	0xffffffff7
0x10010020	0xffffffffeb	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

Với trường hợp mảng có chứa phần tử -2^{31}

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x10010000
t1	6	0x00000009
t2	7	0x00000007
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x80000000
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	0x00000008
t4	29	0x1001001c
t5	30	0x80000000
t6	31	0x80000000
pc		0x00400070

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0xffffffff	0x00000002	0xffffffff	0x00000004	0x00000007	0xffffffff	0x80000000
0x10010020	0xffffffff	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

Giải thích kết quả chạy:

- t3 chứa giá trị tuyệt đối lớn nhất ban đầu (tạm thời là A[0]).
- a1 chứa **giá trị của phần tử có giá trị tuyệt đối lớn nhất** theo yêu cầu của bài toán. (mv a1, t3# a1: max_val = A[0] (giữ nguyên dấu))

- Nếu $A[0]$ âm (bit t_3), thì $t_3 = |A[0]|$.
- Loop: Duyệt mảng bằng cách tính địa chỉ $A[i]$ và tải giá trị vào t_5 .
- Nếu $A[i](t_5) == -2^{31}$, ta chọn luôn làm phần tử có giá trị tuyệt đối lớn nhất.
- Nếu $A[i](t_5)$ âm, đổi dấu để lấy giá trị tuyệt đối.
- Nếu $|A[i]| > \text{max_abs}$, cập nhật $\text{max_abs}(t_3)$ và gán $a1 = A[i]$.
- Tăng chỉ số i ($i++$)
- Kết thúc vòng lặp

KẾT LUẬN:

Lệnh Branch (Rẽ nhánh):

- **Chức năng:** Thay đổi luồng thực thi *có điều kiện*, dựa trên kết quả so sánh. Nếu điều kiện *không* đúng, chương trình tiếp tục lệnh kế tiếp.
- **Địa chỉ đích:** Tính *tương đối* so với PC hiện tại (dùng offset).
- **Khuôn dạng:** B-type/SB-type (sử dụng offset 12-bit).
- **Mục đích:** Dùng cho các cấu trúc điều khiển trong phạm vi gần (vòng lặp, if-else).

Lệnh Jump (Nhảy):

- **Chức năng:** Thay đổi luồng thực thi *vô điều kiện*. Luôn luôn nhảy đến địa chỉ mới.
- **Địa chỉ đích:** Có thể *tương đối* (offset so với PC) hoặc *tuyệt đối* (tính từ giá trị thanh ghi + offset).
- **Khuôn dạng:** J-type (jal), I-type (jalr), hoặc pseudo-instruction (j).
- **Mục đích:** Dùng cho lời gọi hàm, returns, và nhảy đến các vị trí xa.

Tại sao khác khuôn dạng?

- **Chức năng khác:** Branch *có điều kiện*, jump *vô điều kiện*.

- **Địa chỉ đích khác:** Branch thường dùng *tương đối*, jump có thể dùng cả *tương đối* và *tuyệt đối*.
- **Tối ưu hóa:** Khuôn dạng khác nhau giúp RISC-V:
 - Dùng ít bit cho branch (offset nhỏ).
 - Hỗ trợ cả địa chỉ tương đối và tuyệt đối.
 - Thực hiện branch và jump hiệu quả.

Tóm lại: Branch rẽ nhánh *có điều kiện* và thường dùng địa chỉ *tương đối*.
Jump nhảy *vô điều kiện* và có thể dùng cả địa chỉ *tương đối* và *tuyệt đối*.
Khuôn dạng khác nhau để tối ưu hóa cho các mục đích sử dụng khác nhau.