BÁO CÁO THỰC HÀNH

IT3280 – 156789 – THỰC HÀNH KIẾN TRÚC MÁY TÍNH

NỘI DUNG

Lab 02: Tập lệnh, các lệnh cơ bản, các chỉ thị biên dịch

Họ và tên	Nguyễn Minh Quân
Mã số sinh viên	20235815

Assignment 1

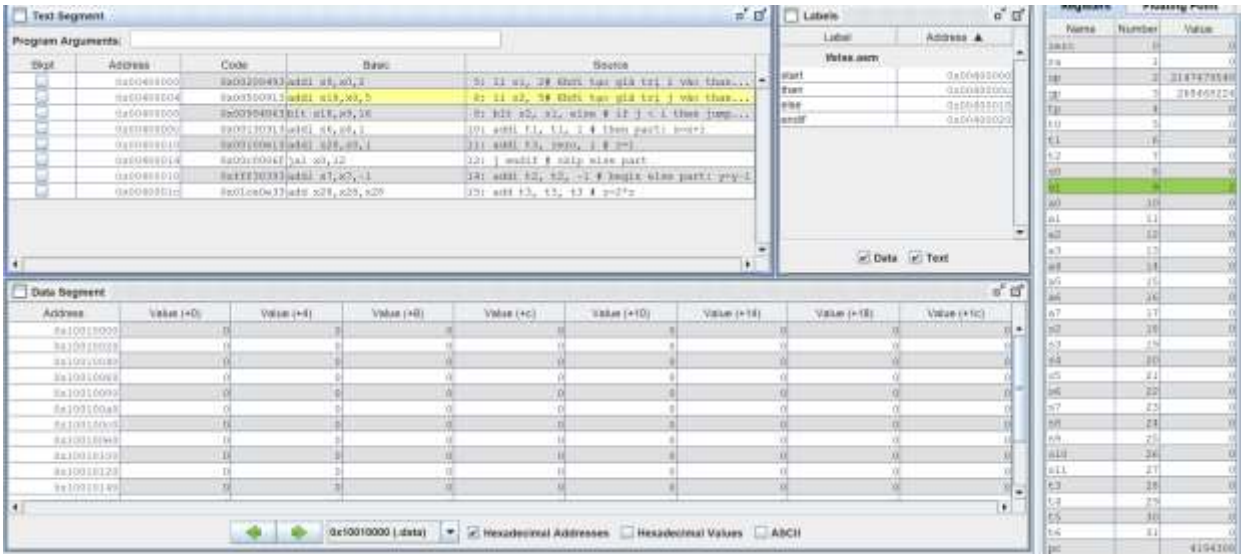
Tạo project để thực hiện đoạn mã trong Home Assignment 1. Khởi tạo các biến cần thiết. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, kiểm tra sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy.

Chương trình thực hiện:

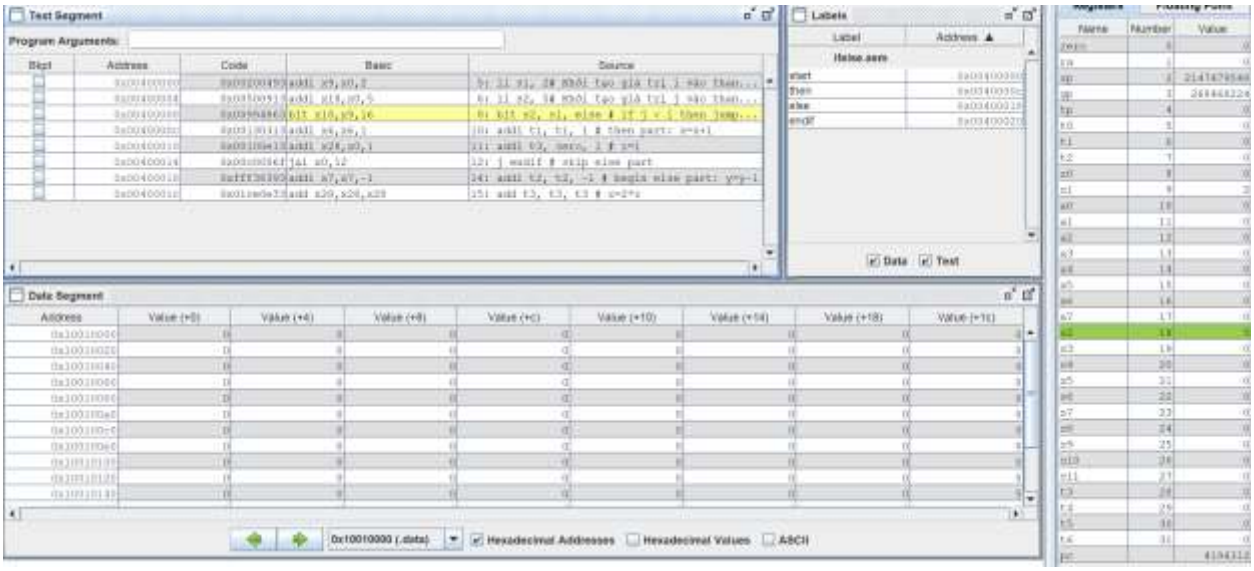
```
1  # Laboratory Exercise 3, Home Assignment 1
2  .text
3  start:
4  # TODO:
5  li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
6  li s2, 5# Khởi tạo giá trị j vào thanh ghi s2
7  # Cách 1:
8  blt s2, s1, else # if j < i then jump else
9  then:
10 addi t1, t1, 1 # then part: x=x+1
11 addi t3, zero, 1 # z=1
12 j endif # skip else part
13 else:
14 addi t2, t2, -1 # begin else part: y=y-1
15 add t3, t3, t3 # z=2*z
16 endif:
```

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.

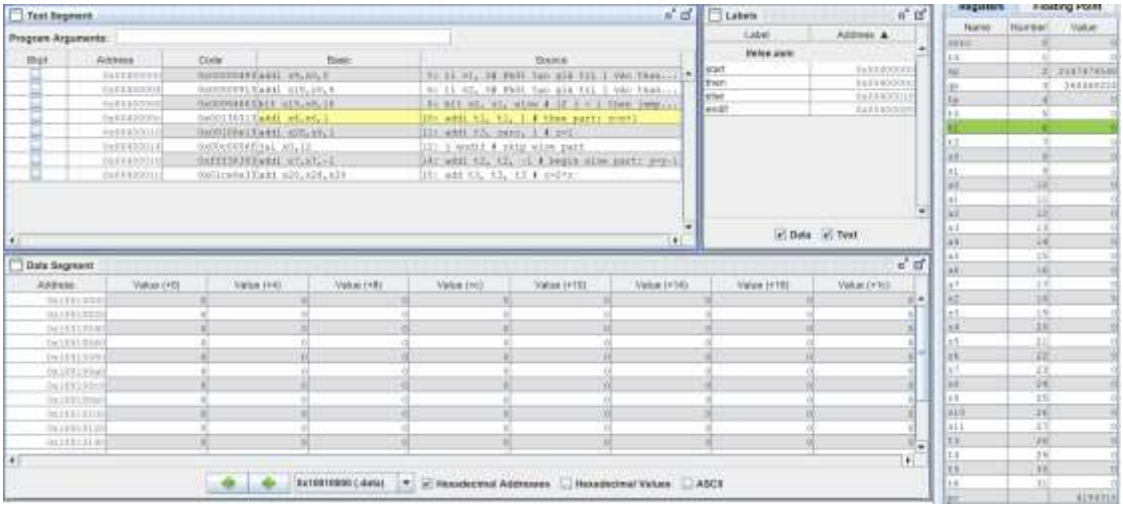
- Lệnh 1: li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
  S1 = 2



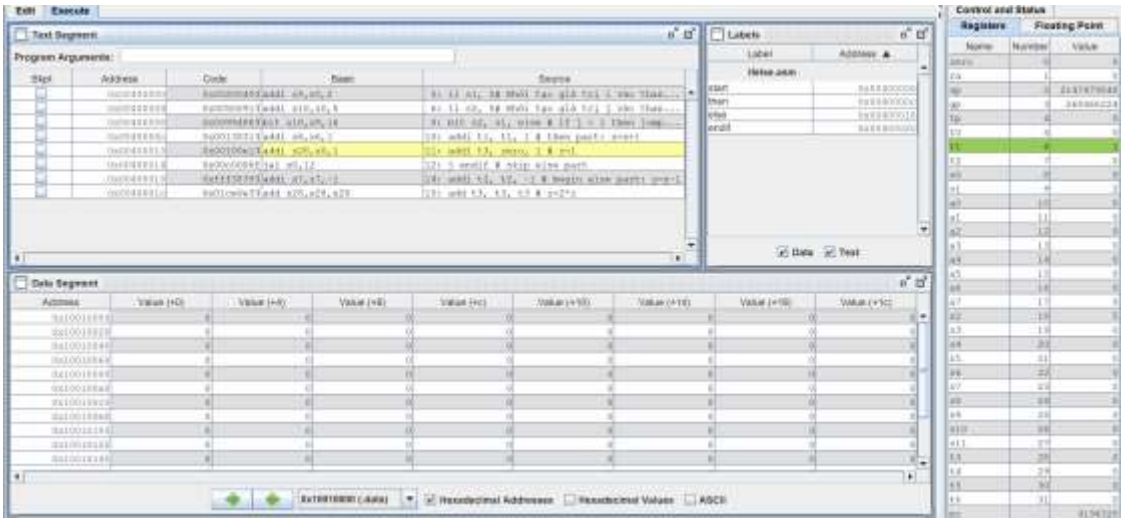
- Lệnh 2: li s2, 5# Khởi tạo giá trị j vào thanh ghi s2
  S2 = 5



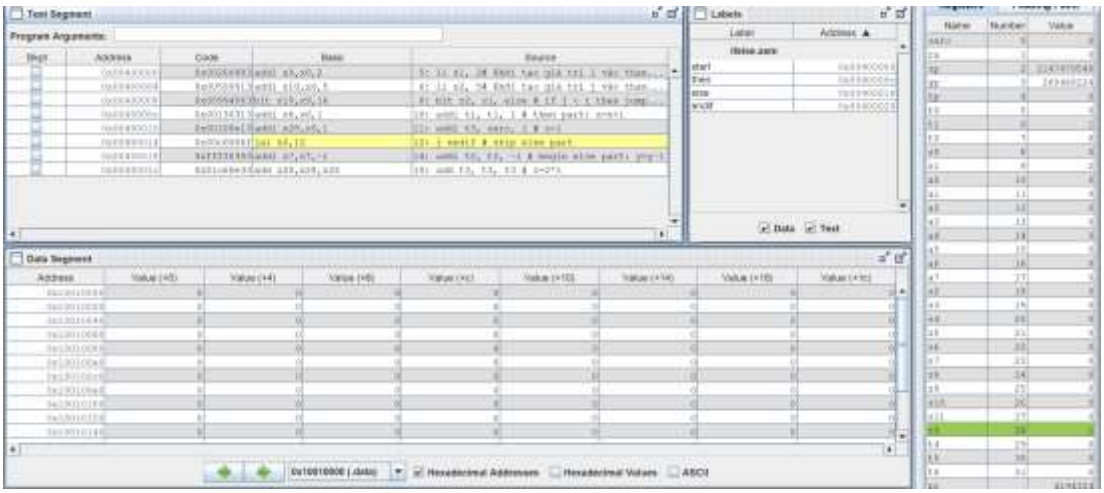
- Lệnh 3: blt s2, s1, else # if j < i thì nhảy tới phần else
  Vì j = 5 > i = 2 → nhảy tới label then



- Lệnh 4: addi t1, t1, 1 # then part: x=x+1
  Tăng giá trị t1 lên 1 đơn vị



- Lệnh 5: addi t3, zero, 1 # Gán z =1

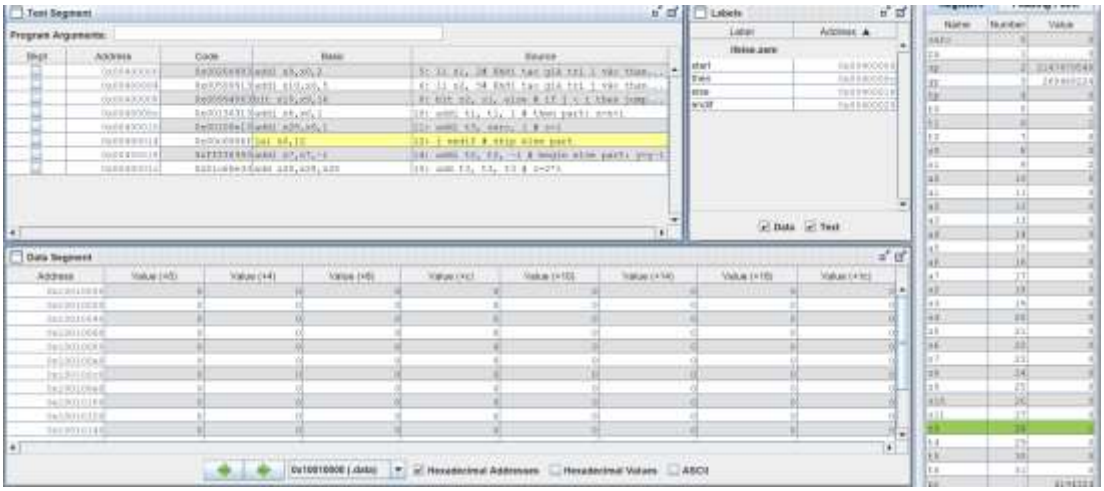


- Lệnh 6: j endif # Bỏ qua phần else và kết thúc chương trình

Với bộ trường hợp trên: i = 2, j =5. Vì i<j nên kết quả sẽ là

  - X = x + 1 = 1
  - Z=1

Kết quả thu được giống với lý thuyết



Assignment 2

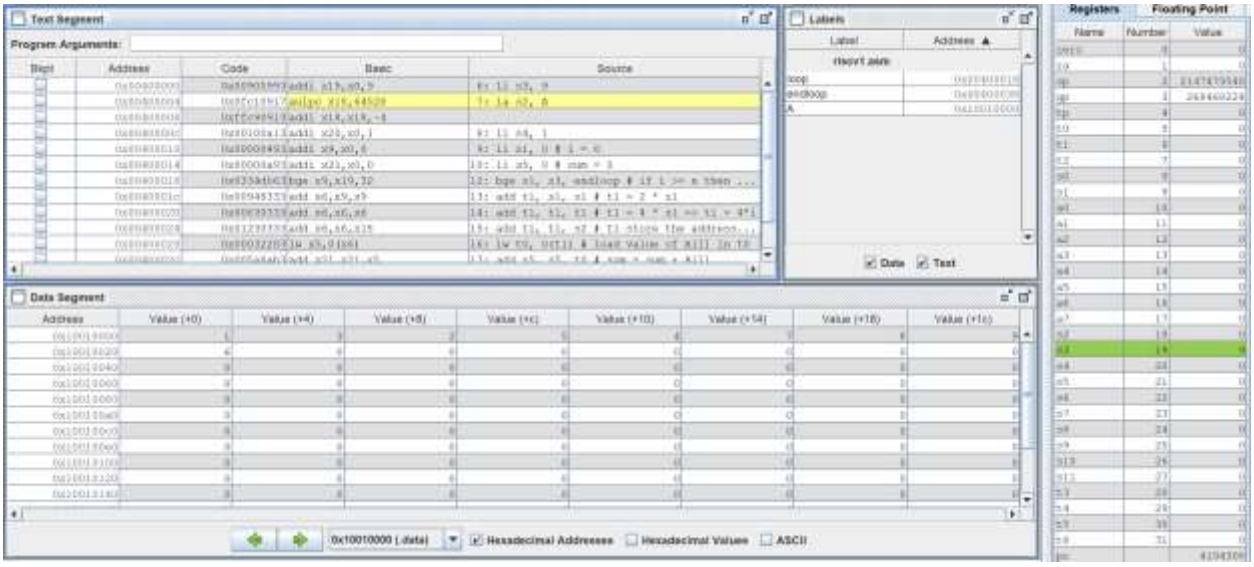
Tạo project để thực hiện đoạn mã trong Home Assignment 2. Khởi tạo các biến cần thiết và mảng A. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy. Thay bộ giá trị khác để kiểm tra sự đúng đắn của chương trình.

Chương trình sau khi khởi tạo các biến và mảng A:

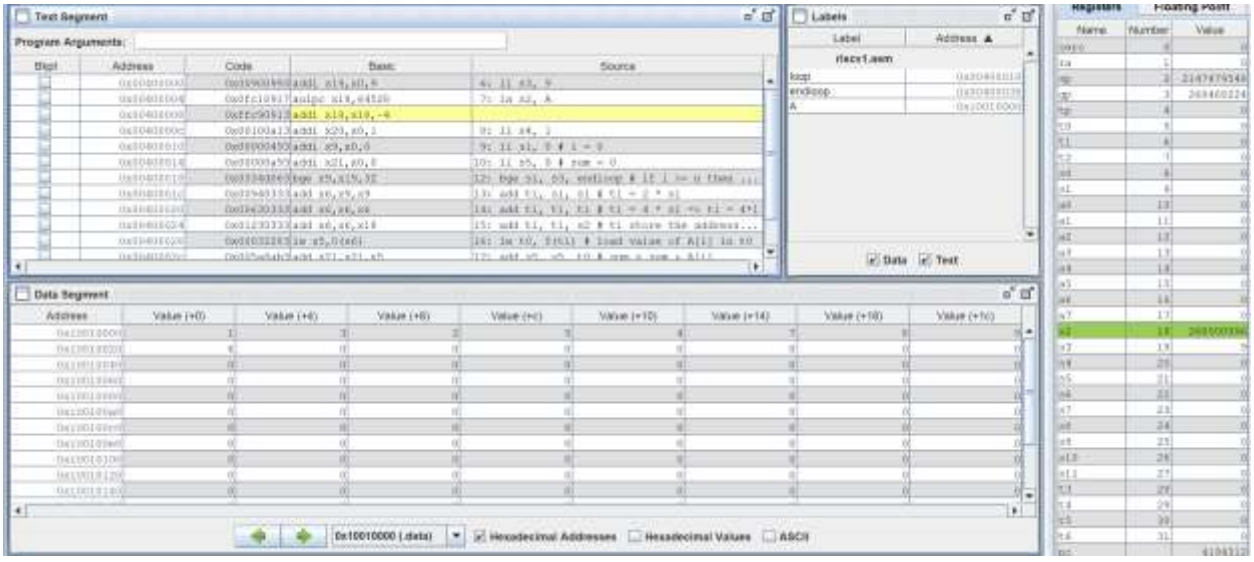
```
# Laboratory 3, Home Assigment 2
.data
A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
.text
# TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
li s3, 9
la s2, A
li s4, 1
li s1, 0 # i = 0
li s5, 0 # sum = 0
loop:
bge s1, s3, endloop # if i >= n then end loop
add t1, s1, s1 # t1 = 2 * s1
add t1, t1, t1 # t1 = 4 * 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:
```

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

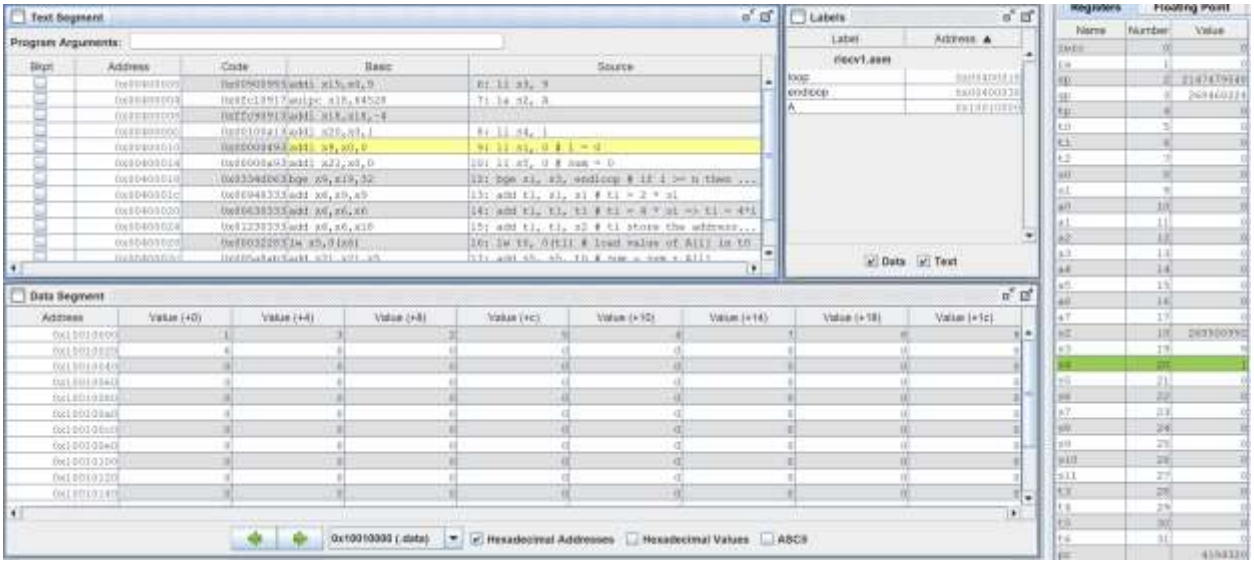
- Lệnh 1: li s3, 9# Gán s3 (số phần tử trong mảng) = 9



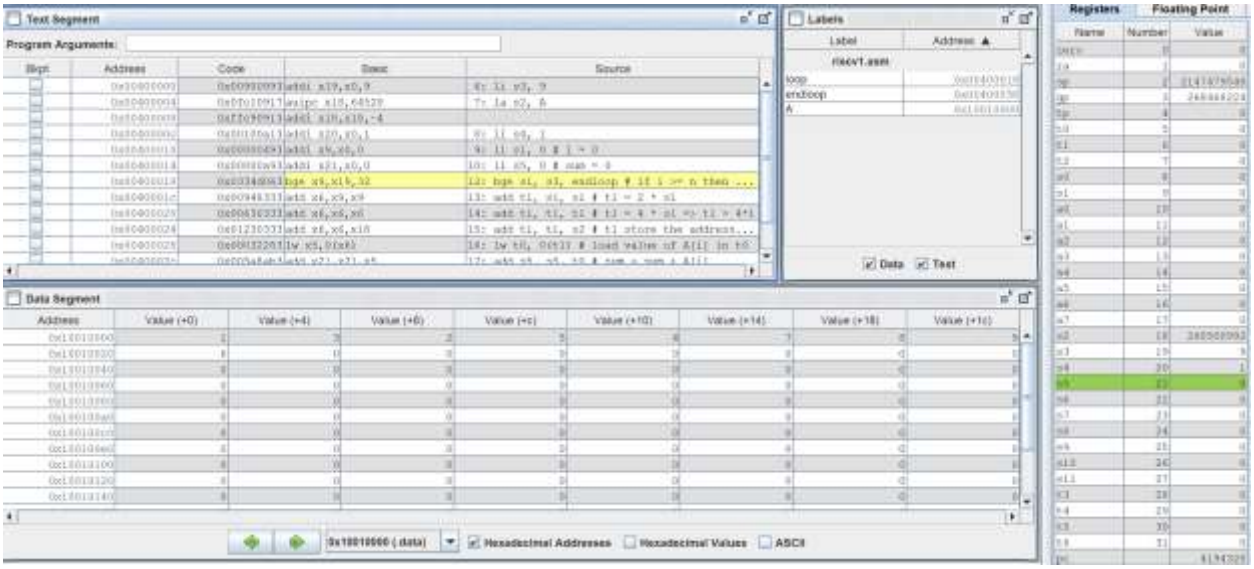
- Lệnh 2: la s2, A #Lưu địa chỉ của mảng vào s2



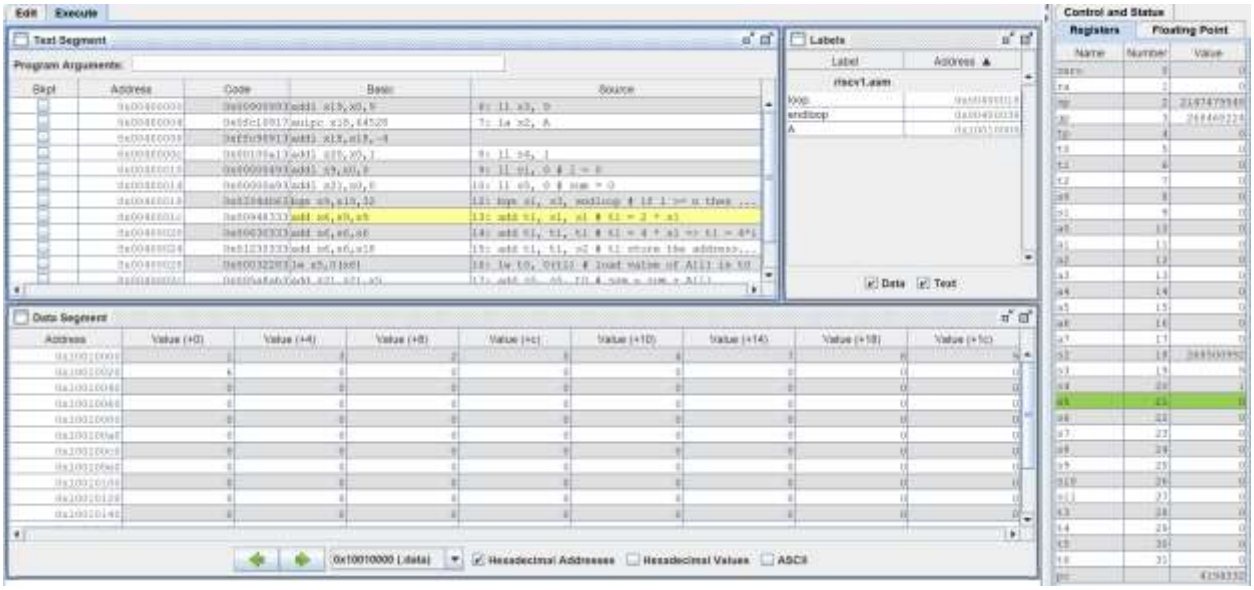
- Lệnh 3: li s4, 1 #s4 = step (bước nhảy, duyệt từng phần tử)



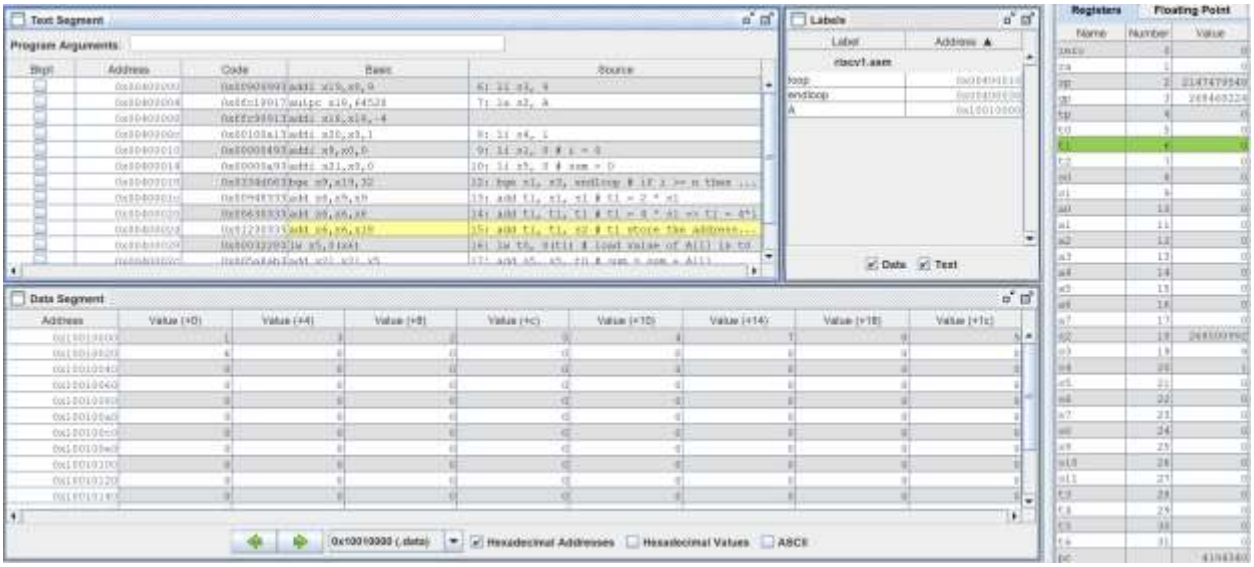
- Lệnh 4 và 5 : li s1, 0 và li s5, 0 # Gán i = 0 và sum=0



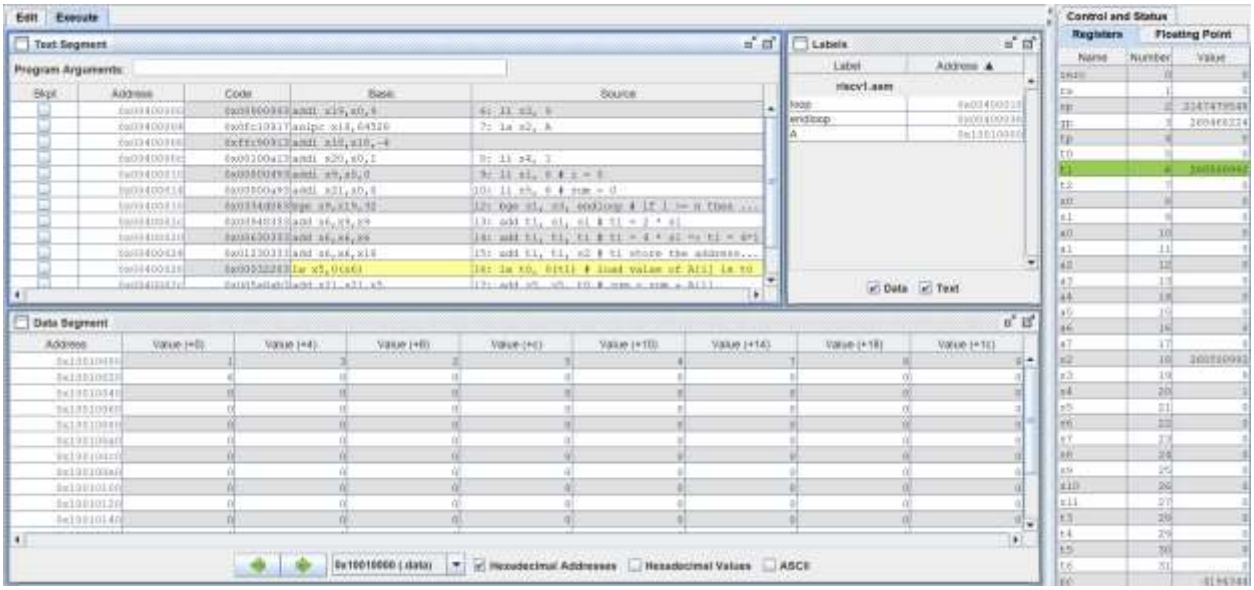
- Lệnh 6: bge s1, s3, endloop # if i >= n then thoát vòng lặp



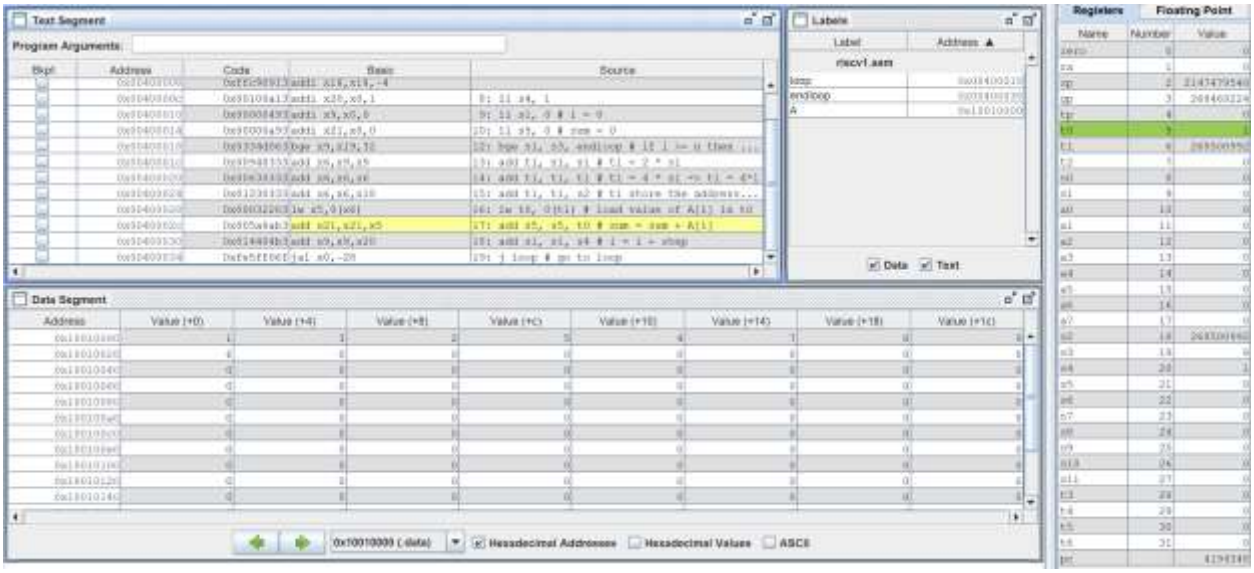
- Lệnh 7 và 8 : add t1, s1, s1 # t1 = 2 * s1
  add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i



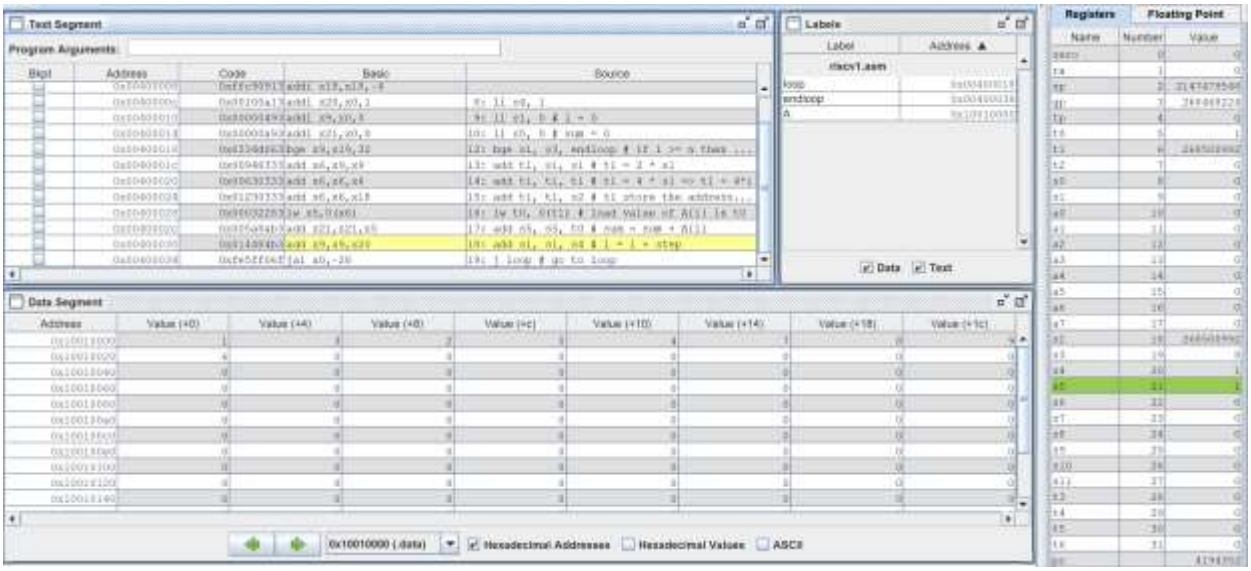
- Lệnh 9 : add t1, t1, s2 # t1=A + 4*i → t1 lưu trữ địa chỉ của A[i]



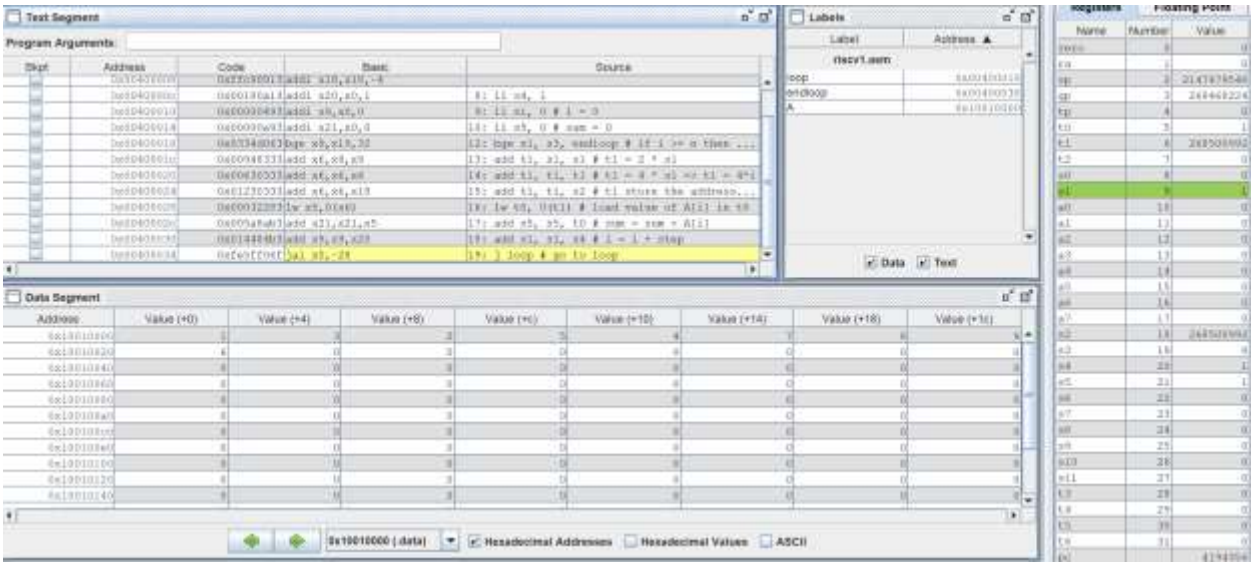
- Lệnh 10: lw t0, 0(t1) # load giá trị A[i] vào t0



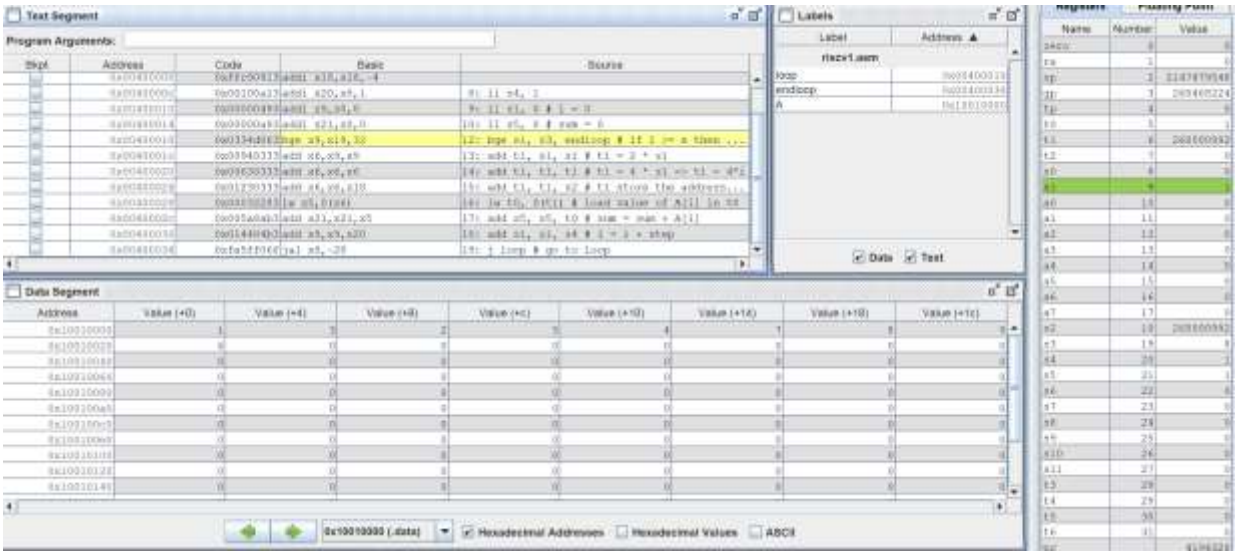
- Lệnh 11: add s5, s5, t0 # sum = sum + A[i]



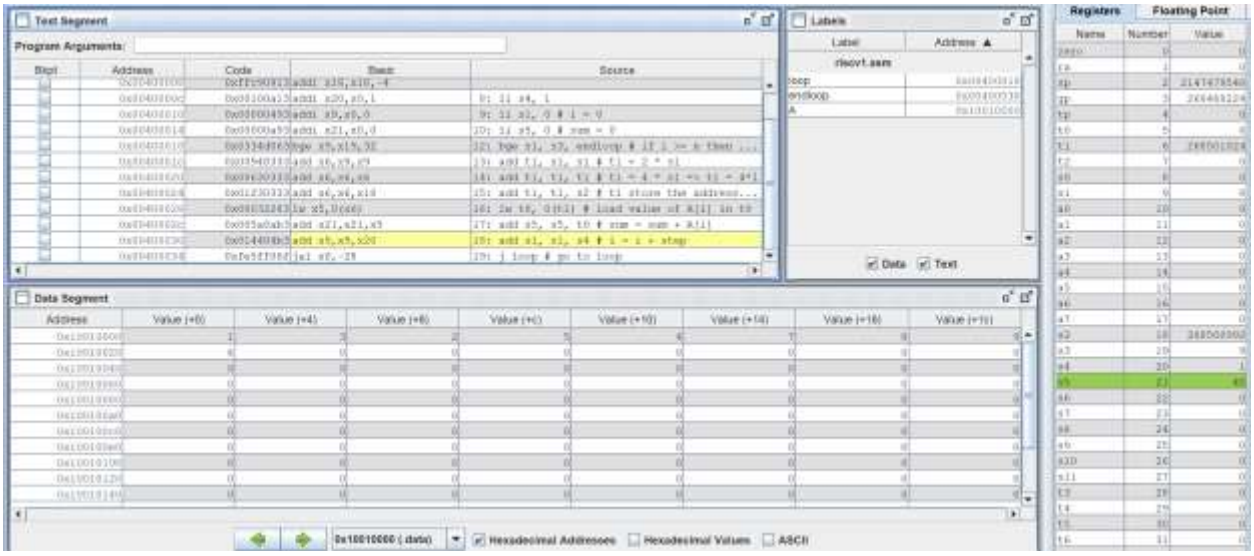
- Lệnh 12: add s1, s1, s4 # i = i + 1



- Lệnh 13: j loop # quay lại đầu vòng lặp



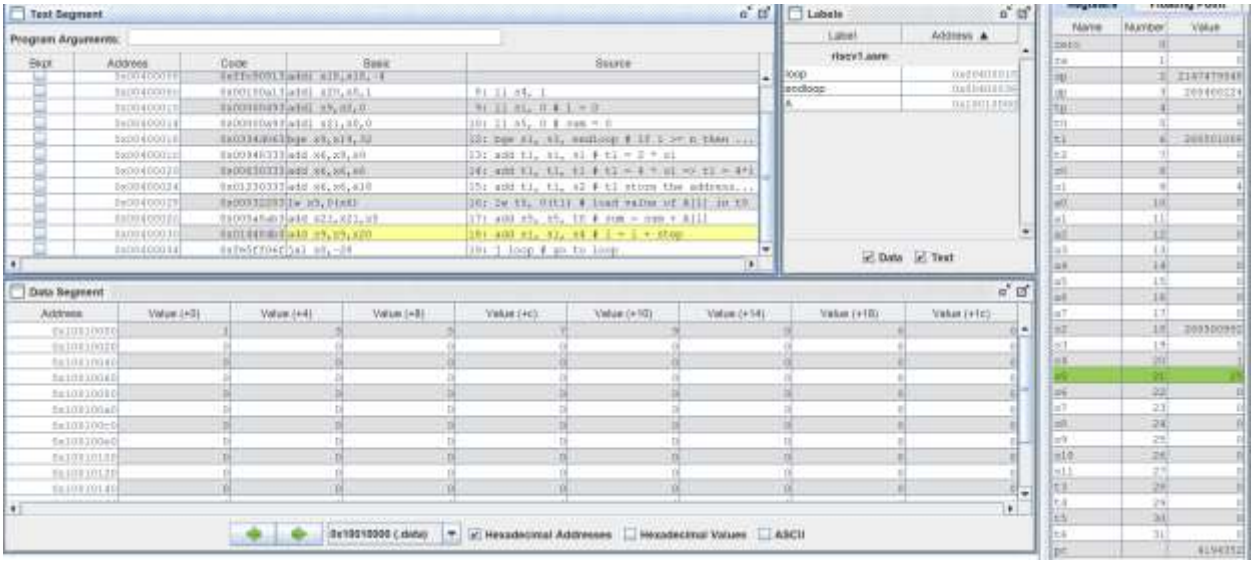
- Lặp lại các bước như trên sẽ cho ta kết quả cuối cùng s5(sum) = 45



Thay bộ giá trị khác
Mảng A : [1, 3, 5, 7, 9]

```
1  # Laboratory 3, Home Assigment 2
2  .data
3  A: .word 1,3,5,7,9
4  .text
5  # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
6  li s3, 5
7  la s2, A
8  li s4, 1
9  li s1, 0 # i = 0
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 sẽ thu được kết quả s5(sum) = 25



Assignment 3

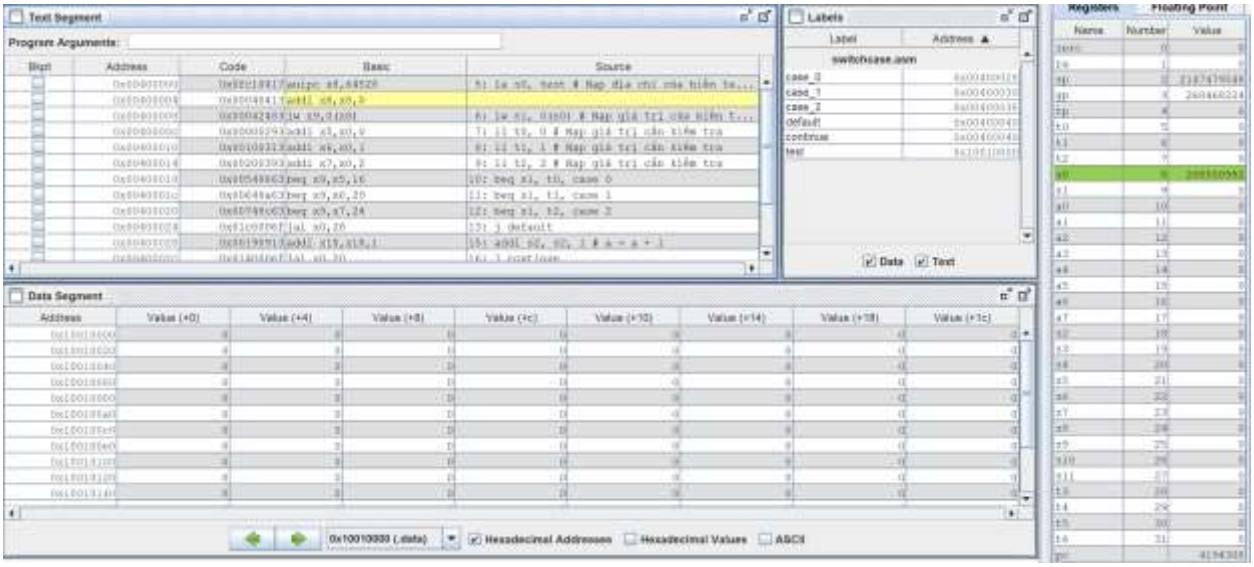
Tạo project để thực hiện đoạn mã trong Home Assignment 3. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở từng bước chạy. Thay đổi bộ giá trị và chạy lại chương trình một vài lần để kiểm tra tất cả các trường hợp.

Chương trình thực hiện :

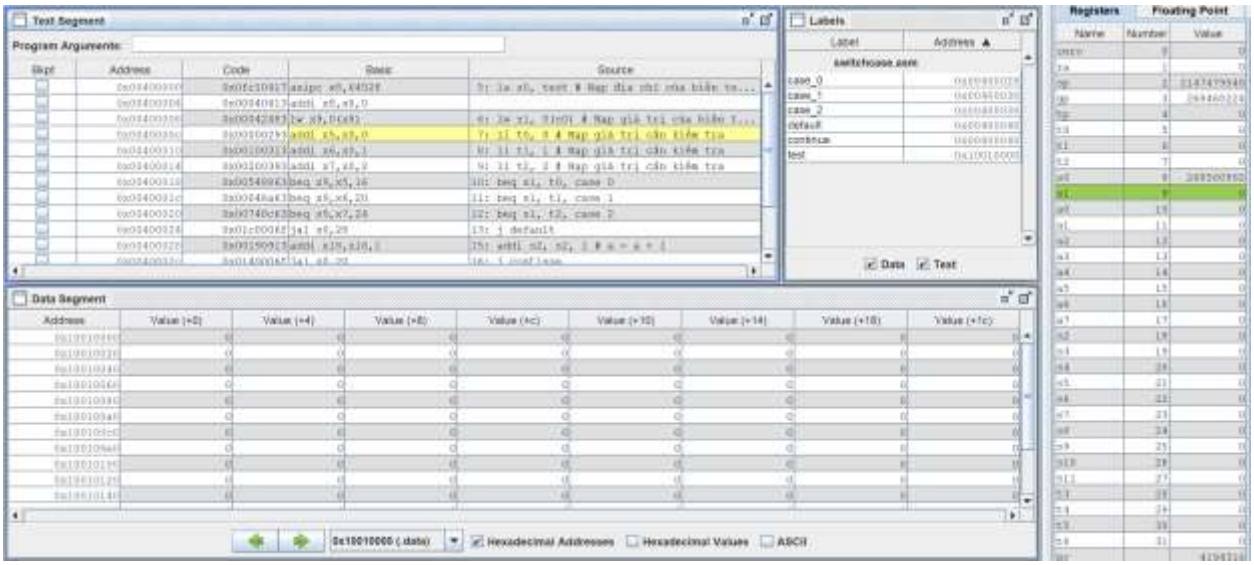
```
# Laboratory Exercise 3, Home Assignment 3
.data
test: .word 0
.text
la s0, test # Nạp địa chỉ của biến test vào s0
lw s1, 0(s0) # Nạp giá trị của biến test vào s1
li t0, 0 # Nạp giá trị cần kiểm tra
li t1, 1 # Nạp giá trị cần kiểm tra
li t2, 2 # Nạp giá trị cần kiểm tra
beq s1, t0, case_0
beq s1, 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:
```

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.

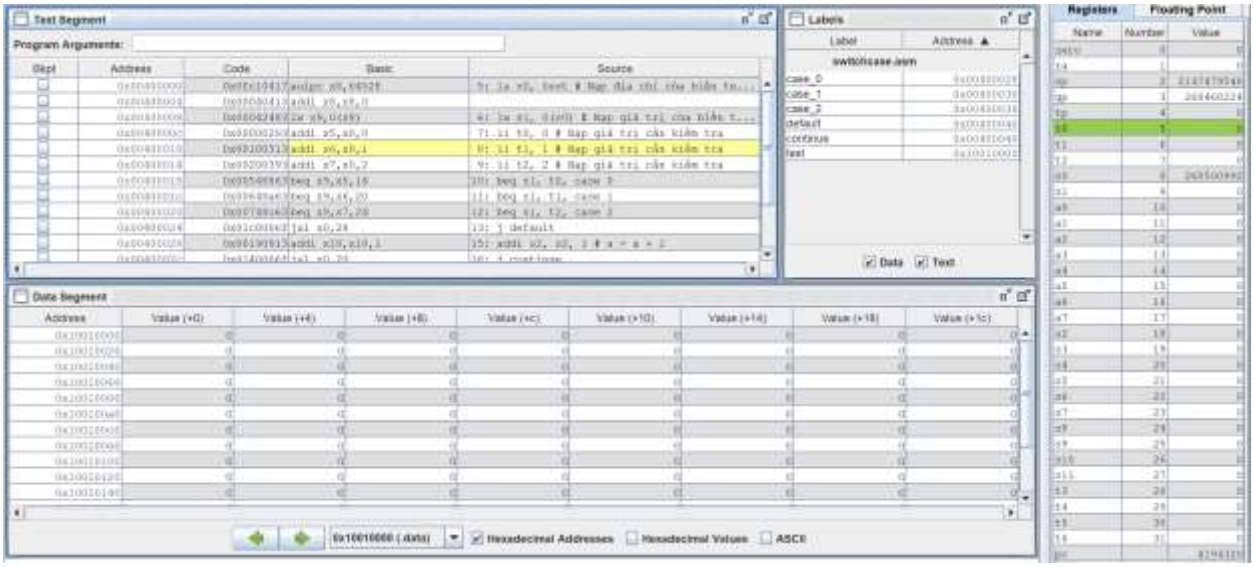
- Lệnh 1: la s0, test # Nạp địa chỉ của biến test vào s0



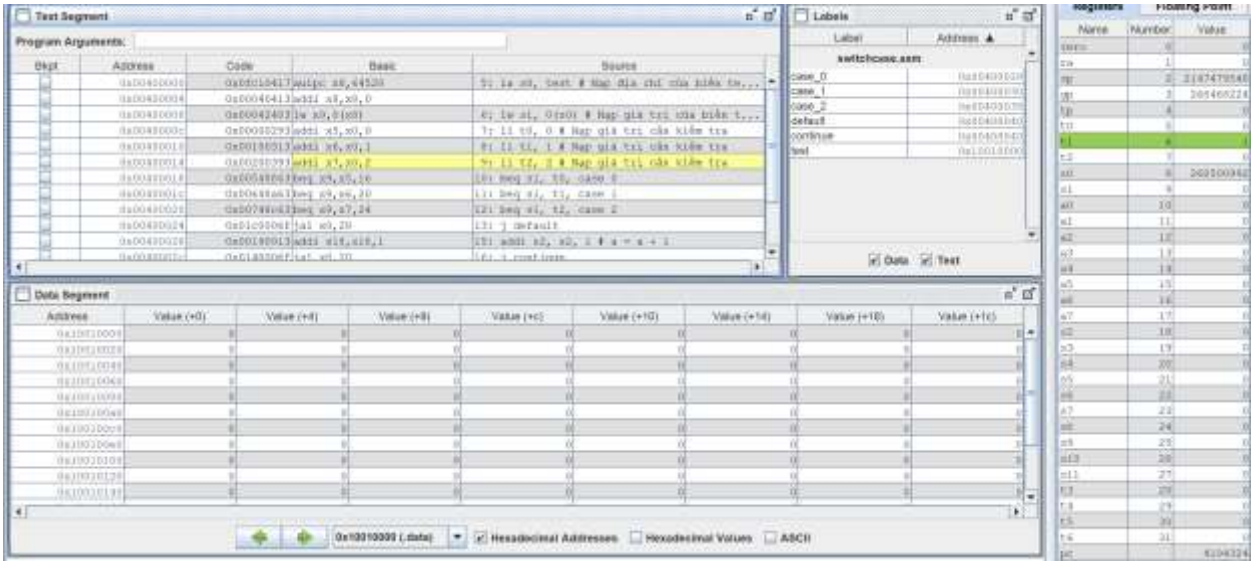
- Lệnh 2: lw s1, 0(s0) # Nạp giá trị của biến test vào s1



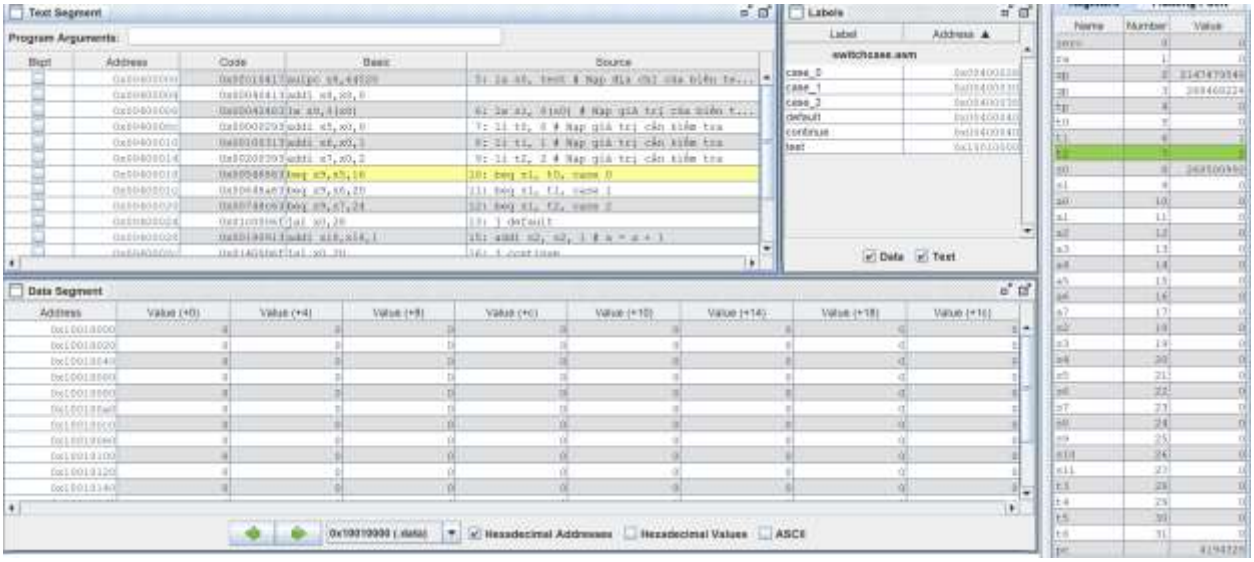
- Lệnh 3: li t0, 0 # Nạp giá trị cần kiểm tra



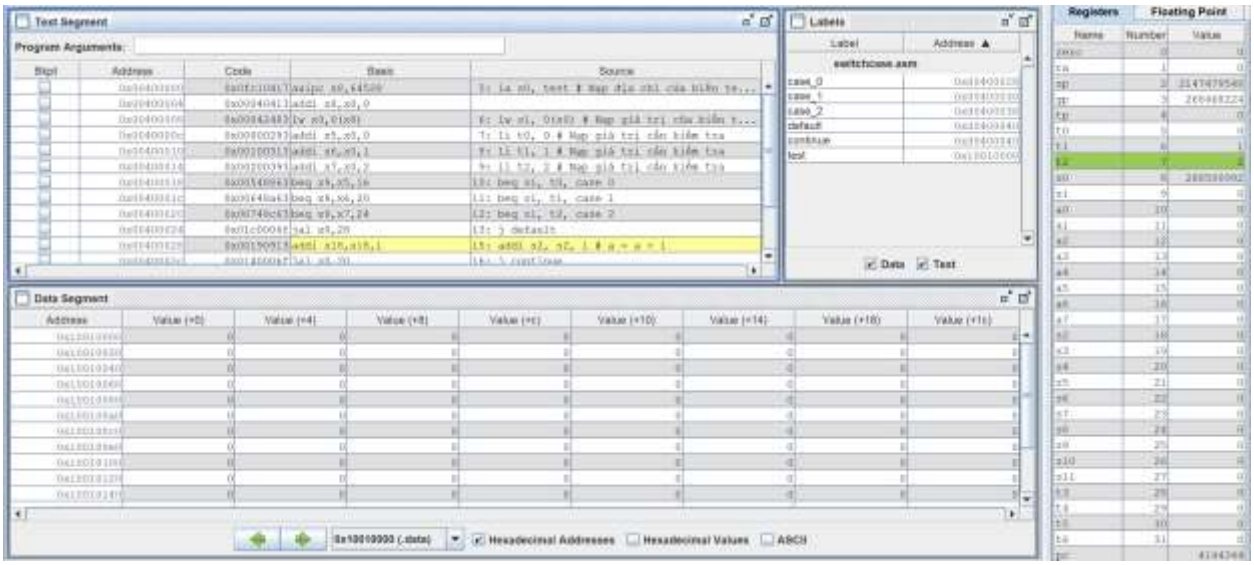
- Lệnh 4: li t1, 1 # Nạp giá trị cần kiểm tra



- Lệnh 5: li t2, 2 # Nạp giá trị cần kiểm tra



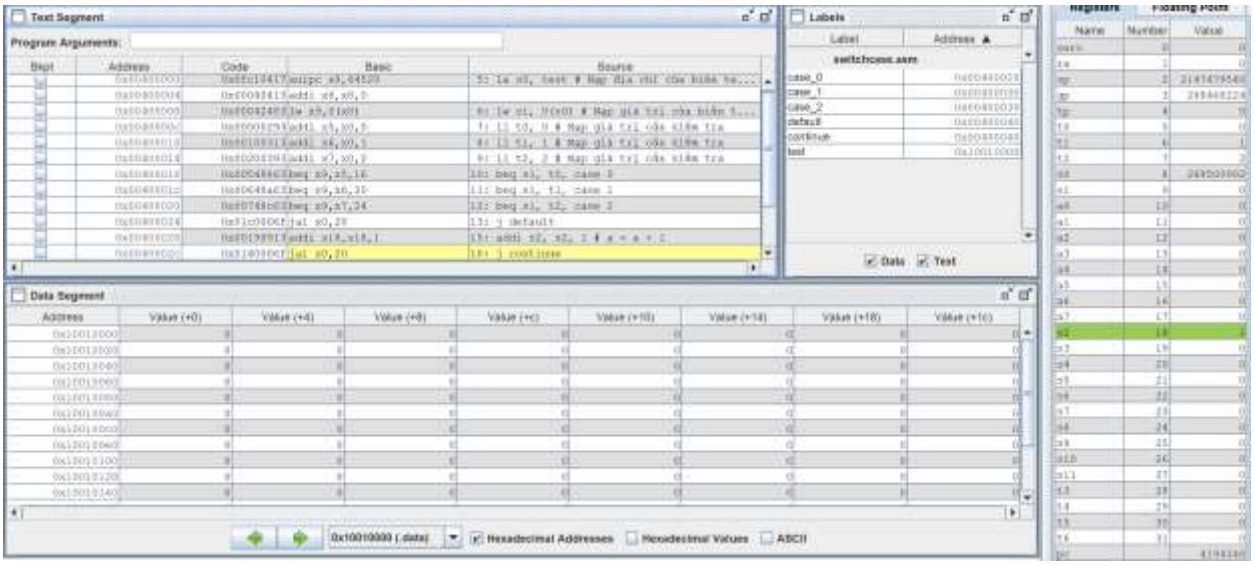
- Lệnh 6: beq s1, t0, case_0 # Nếu s1 == 0, nhảy tới case_0



Lệnh 7:beq s1, t1, case_1# Nếu s1 == 1, nhảy tới case_1

Lệnh 8:beq s1, t2, case_2# Nếu s1 == 2, nhảy tới case_2

Lệnh 9:addi s2, s2, 1 # a = a + 1



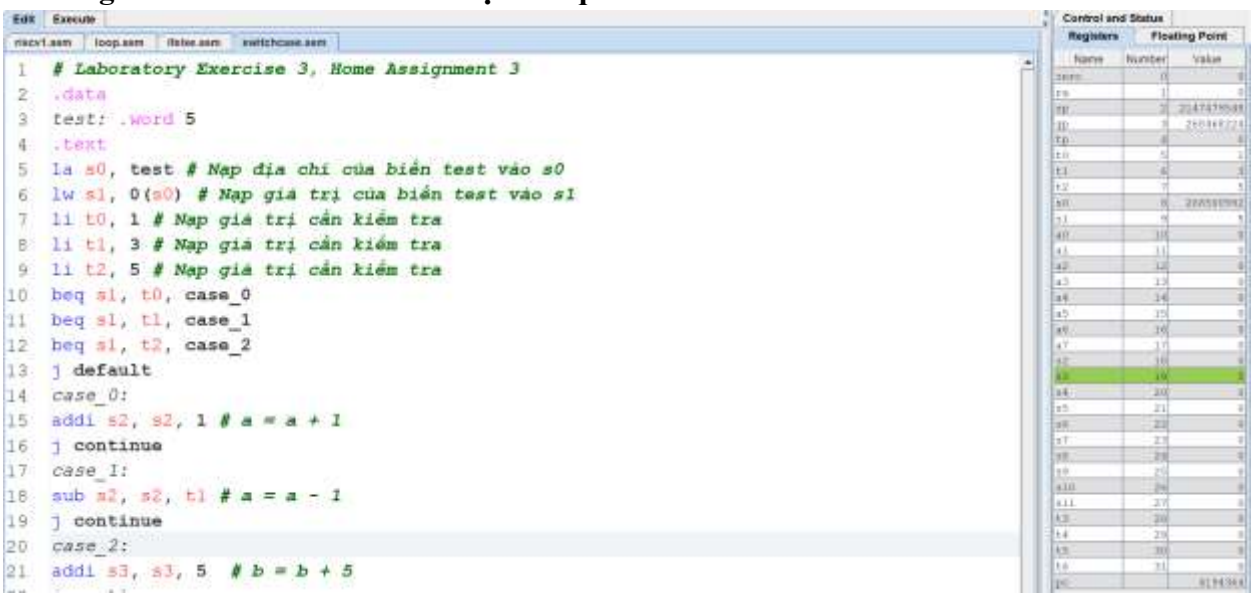
Lệnh 10:sub s2, s2, t1 # a = a – 1
Lệnh 11:add s3, s3, s3 # b = 2 * b

Bộ trường hợp khác

```
# 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, 1 # Nạp giá trị cần kiểm tra
li t1, 3 # Nạp giá trị cần kiểm tra
li t2, 5 # 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, 5  # b = b + 5
j continue
default:
continue:
```

Sau khi so sánh thấy trùng với t2 → nhảy tới case thực hiện lệnh s3 = s3 + 5

Chương trình kết thúc ta sẽ thu được kết quả s3 = 5



Assignment 4

Lần lượt thay thế điều kiện rẽ nhánh trong Home Assignment 1 bằng các điều kiện sau đây: a. i < j b. i >= j c. i + j <= 0 d. i + j > m + n (với m và n là các giá trị đã được lưu trong các thanh ghi)

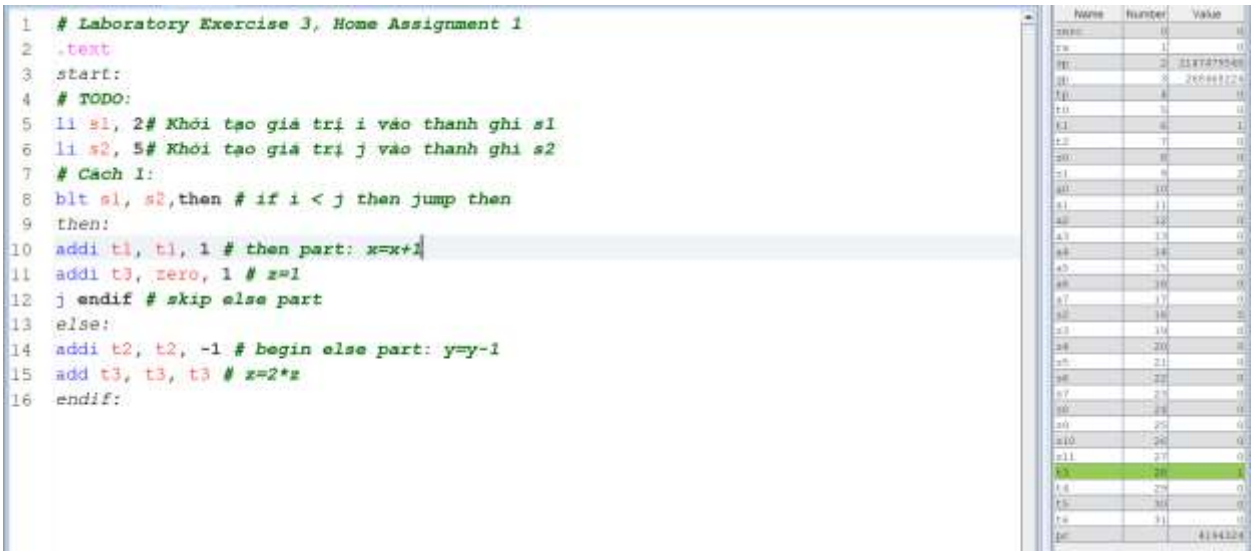
a) **i < j**

```
1  # Laboratory Exercise 3, Home Assignment 1
2  .text
3  start:
4  # TODO:
5  li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
6  li s2, 5# Khởi tạo giá trị j vào thanh ghi s2
7  # Cách 1:
8  blt s1, s2,then # if i < j then jump then
9  then:
10 addi t1, t1, 1 # then part: x=x+1
11 addi t3, zero, 1 # z=1
12 j endif # skip else part
13 else:
14 addi t2, t2, -1 # begin else part: y=y-1
15 add t3, t3, t3 # z=2*z
16 endif:
```

Khi chương trình kiểm tra thấy i=2 < j=5 thì lập tức jump tới label then thực hiện câu lệnh ở trong

Các bước còn lại thì tương tự như Assignment 1

Sau khi chương trình kết thúc, ta thu được kết quả t1=1 và t3=1



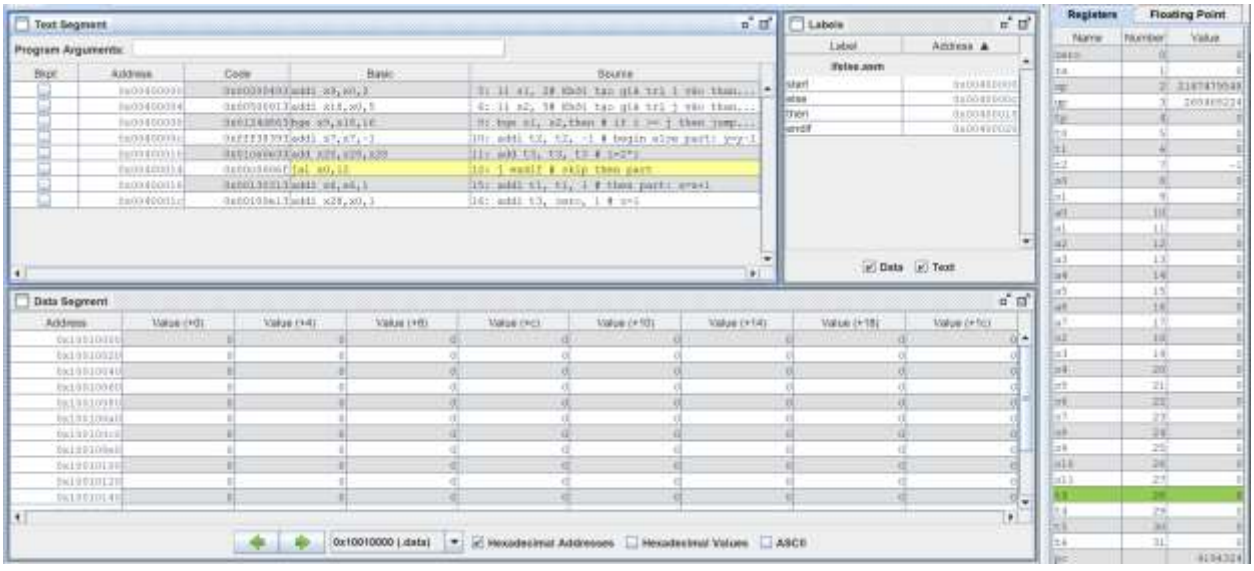
b) i >= j

```
1   # Laboratory Exercise 3, Home Assignment 1
2   .text
3   start:
4   # TODO:
5   li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
6   li s2, 5# Khởi tạo giá trị j vào thanh ghi s2
7   # Cách 1:
8   bge s1, s2,then # if i >= j then jump then
9   else:
10  addi t2, t2, -1 # begin else part: y=y-1
11  add t3, t3, t3 # z=2*z
12  j endif # skip then part
13
14  then:
15  addi t1, t1, 1 # then part: x=x+1
16  addi t3, zero, 1 # z=1
17  endif:
```

Chương trình kiểm tra thấy i=2 < j =5, thay vì nhảy tới then sẽ nhảy tới else thực hiện 2 câu lệnh trên

Các bước còn lại tương tự như assignment 1

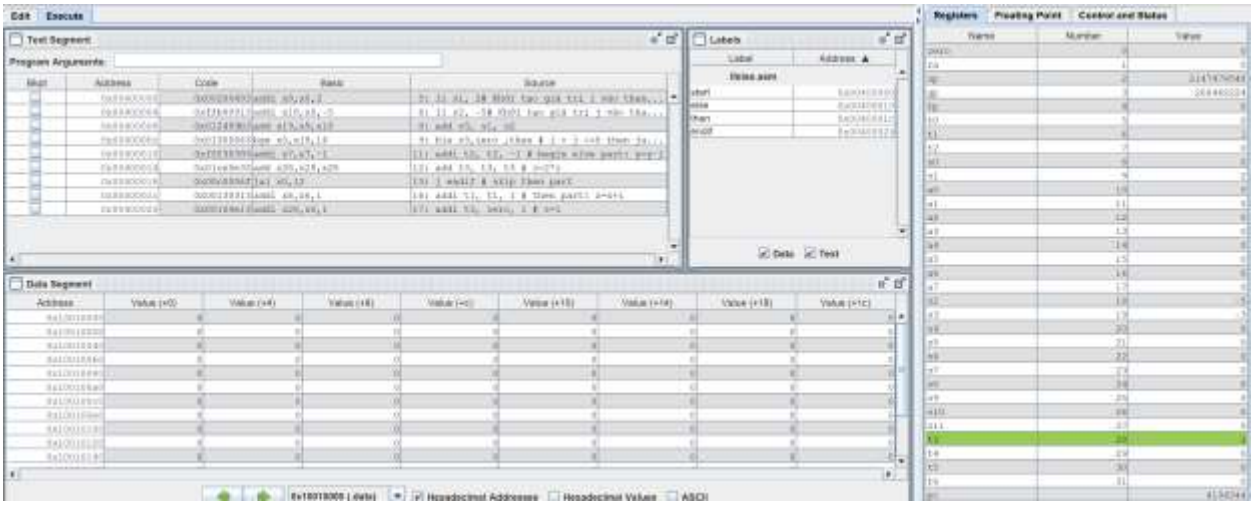
Sau khi chương trình kết thúc ta thu được kết quả t1=1 và t3 =0



c) i + j <= 0

```
1  # Laboratory Exercise 3, Home Assignment 1
2  .text
3  start:
4  # TODO:
5  li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
6  li s2, -5# Khởi tạo giá trị j vào thanh ghi s2
7
8  add s3, s1, s2
9  ble s3,zero ,then # i + j <=0 then jump then
10 else:
11 addi t2, t2, -1 # begin else part: y=y-1
12 add t3, t3, t3 # z=2*z
13 j endif # skip then part
14
15 then:
16 addi t1, t1, 1 # then part: x=x+1
17 addi t3, zero, 1 # z=1
18 endif:
```

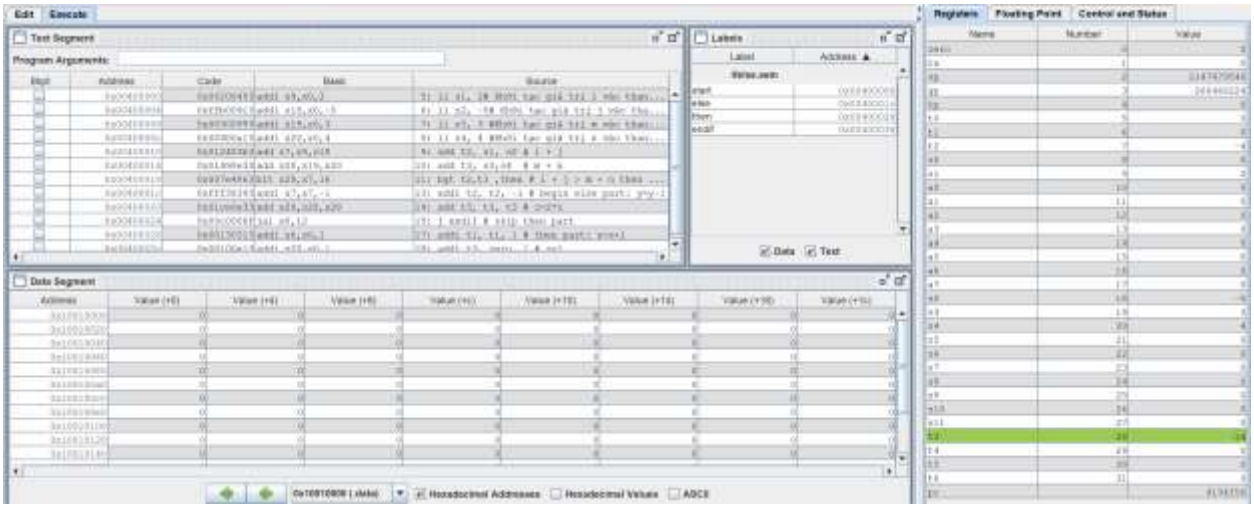
Sau khi chương trình kết thúc ta thu được kết quả là t1 = 1 và t3=1



d) i + j > m + n

```
1  # Laboratory Exercise 3, Home Assignment 1
2  .text
3  start:
4  # TODO:
5  li s1, 2# Khởi tạo giá trị i vào thanh ghi s1
6  li s2, -5# Khởi tạo giá trị j vào thanh ghi s2
7  li s3, 3 #Khởi tạo giá trị m vào thanh ghi s3
8  li s4, 4 #Khởi tạo giá trị n vào thanh ghi s4
9  add t2, s1, s2 # i + j
10 add t3, s3,s4  # m + n
11 bgt t2,t3 ,then # i + j > m + n then jump then
12 else:
13 addi t2, t2, -1 # begin else part: y=y-1
14 add t3, t3, t3 # z=2*z
15 j endif # skip then part
16 then:
17 addi t1, t1, 1 # then part: x=x+1
18 addi t3, zero, 1 # z=1
19 endif:
```

Sau khi chương trình kết thúc ta thu được kết quả t2 = -4, t3 = 14



Assignment 5

Lần lượt thay thế điều kiện nhảy (thoát khỏi vòng lặp) trong Home Assignment 2 bằng các điều kiện sau đây: a. i > n b. sum < 0 c. A[i] == 0

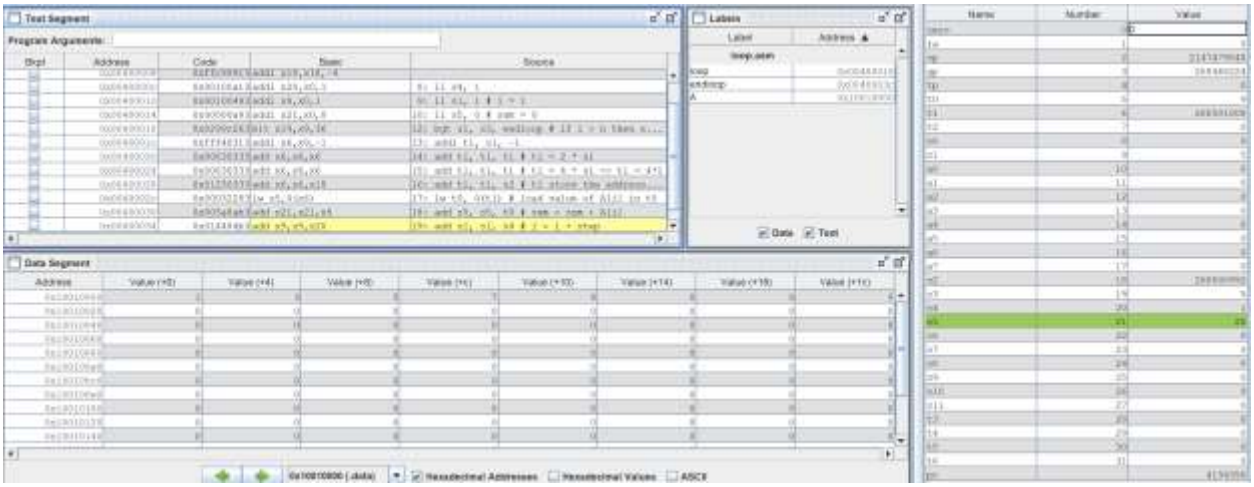
Cần thiết lập giá trị các phần tử của mảng để điều kiện có thể được thỏa mãn.

a) i > n

Với điều kiện nhảy là i > n, ta cho i chạy từ 1 – 5 đồng nghĩa với việc phải gán thanh ghi t1 = i – 1 bởi mảng A có tối đa 5 phần tử theo thứ tự [0,1,2,3,4]

```
A: .word 1,3,5,7,9
.text
# TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
li s3, 5
la s2, A
li s4, 1
li s1, 1 # i = 1
li s5, 0 # sum = 0
loop:
bgt s1, s3, endloop # if i > n then end loop
addi t1, s1, -1
add t1, t1, t1 # 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:
```

Sau khi chương trình kết thúc ta thu được kết quả s5 = 25



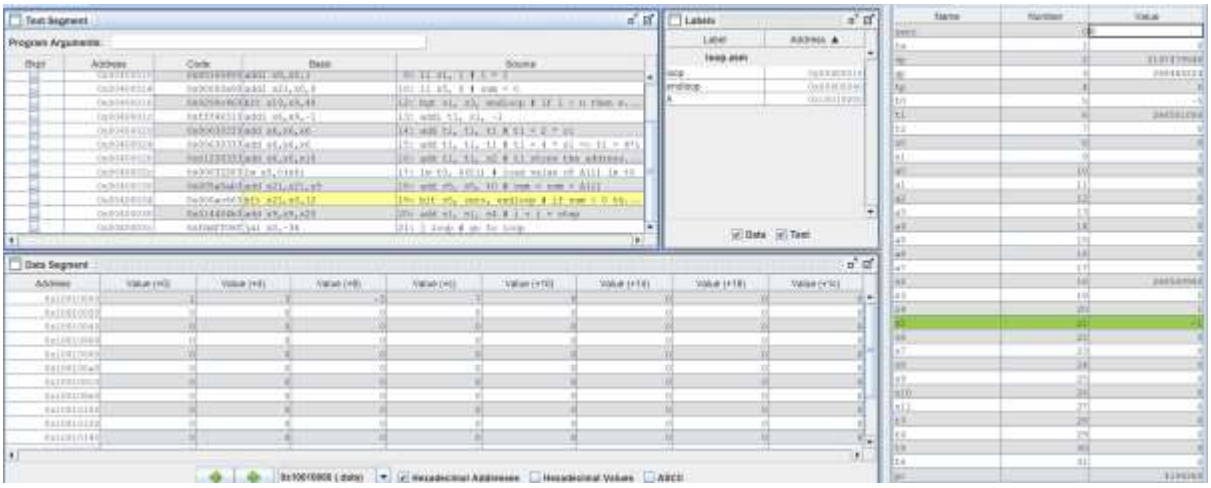
b) sum < 0

Ta thêm câu lệnh blt s5, zero, endloop để kiểm tra xem sum < 0 hay không, có thì endloop luôn

Phần còn lại thì như Assignment 3

```
3   A: .word 1,3,-5,7,9
4   .text
5   # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
6   li s3, 5
7   la s2, A
8   li s4, 1
9   li s1, 1 # i = 1
10  li s5, 0 # sum = 0
11  loop:
12  bgt s1, s3, endloop # if i > n then end loop
13  addi t1, s1, -1
14  add t1, t1, t1 # t1 = 2 * s1
15  add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
16  add t1, t1, s2 # t1 store the address of A[i]
17  lw t0, 0(t1) # load value of A[i] in t0
18  add s5, s5, t0 # sum = sum + A[i]
19  blt s5, zero, endloop # if sum < 0 then end loop
20  add s1, s1, s4 # i = i + step
21  j loop # go to loop
22  endloop:
```

Sau khi chương trình kết thúc, ta thu được kết quả s5 = -1 (Do s5 < 0 nên sẽ break luôn khỏi vòng lặp)



c) A[i] ==0

Ta thêm câu lệnh beq t0, zero, endloop để kiểm tra giá trị của A[i] với 0. Nếu bằng 0 thì break vòng lặp.

Phần còn lại thì tương tự như Assignment 3

```
# Laboratory 3, Home Assigment 2
.data
A: .word 1,3,0,7,9
.text
# TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4
li s3, 5
la s2, A
li s4, 1
li s1, 1 # i = 1
li s5, 0 # sum = 0
loop:
bgt s1, s3, endloop # if i > n then end loop
addi t1, s1, -1
add t1, t1, t1 # 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
beq t0, zero, endloop
add s5, s5, t0 # sum = sum + A[i]
add s1, s1, s4 # i = i + step
j loop # go to loop
endloop:
```

Sau khi kết thúc chương trình, ta thu được kết quả s5 = 4 (Do A[2] = 0 → break vòng lặp)

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.

Chương trình thực hiện:

```
.data
        A: .word 1, -3 , 5, -4, 8 , -10
.text
        li s3, 6 # so phan tu cua A
        la s4, A
        li s1, 0 # i =0
        li s5, 0 # max = 0
loop:
        bge s1, s3, endloop
        add t1, s1, s1  #i=2*s1
        add t1, t1, t1  #i = 4 * s1 = 4*i
        add t1, t1, s4  #luu tru dia chi cua A[i] vao t1
        lw t0, 0(t1) #luu tru gia tri cua A[i] vao t0
    #kiem tra gia tri tuyet doi
        bge t0, zero, checkmax #Neu >=0 thi giu nguyen, di vao check
        neg t0,t0 # Gan t0 = -t0 neu t0 la so am

checkmax:
        bgt t0, s5, then # if A[i] > max -> nhay vao then
        addi s1, s1, 1 #i++
        j loop

then:
        mv s5, t0 #max = A[i]
        addi s1,s1, 1 # i++
        j loop#quay lai loop
endloop:
```