

# 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

The screenshot shows a debugger window with the following components:

- Text Segment:** A table of instructions. The instruction at address 0x00400004 is highlighted:
 

Dispt	Address	Code	Basic	Source
0x00400000	0x00400000	addi s1, s0, 2		li s1, 2, 2# Khởi tạo giá trị i vào thanh...
0x00400004	0x00400005	addi s1, s0, 5		li s1, 5, 5# Khởi tạo giá trị j vào thanh...
- Labels:** A table showing labels and their addresses:
 

Label	Address
start	0x00400000
then	0x00400005
else	0x00400010
endif	0x00400020
- Register Window:** Shows the state of registers. Register s1 contains the value 2.
 

Name	Number	Value
s0	0	0
s1	2	2
s2	3	2147483648
s3	4	2147483648
s4	5	0
s5	6	0
s6	7	0
s7	8	0
s8	9	0
s9	10	0
s10	11	0
s11	12	0
s12	13	0
s13	14	0
s14	15	0
s15	16	0
s16	17	0
s17	18	0
s18	19	0
s19	20	0
s20	21	0
s21	22	0
s22	23	0
s23	24	0
s24	25	0
s25	26	0
s26	27	0
s27	28	0
s28	29	0
s29	30	0
s30	31	0
pc		4194300
- Data Segment:** A table showing memory addresses and their values. The address 0x10010000 is highlighted.
 

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
0x10010000	0	0	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000C	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001C	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002C	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003C	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0

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

The screenshot shows a debugger window with the following components:

- Text Segment:** A table of instructions. The instruction at address 0x00400005 is highlighted:
 

Dispt	Address	Code	Basic	Source
0x00400000	0x00400000	addi s1, s0, 2		li s1, 2, 2# Khởi tạo giá trị i vào thanh...
0x00400004	0x00400005	addi s1, s0, 5		li s1, 5, 5# Khởi tạo giá trị j vào thanh...
0x00400008	0x00400009	blt s1, s2, 5# if i < j then jump...		li s1, s2, s1, else # if i < j then jump...
- Labels:** A table showing labels and their addresses:
 

Label	Address
start	0x00400000
then	0x00400005
else	0x00400010
endif	0x00400020
- Register Window:** Shows the state of registers. Register s2 contains the value 5.
 

Name	Number	Value
s0	0	0
s1	2	2
s2	3	5
s3	4	2147483648
s4	5	2147483648
s5	6	0
s6	7	0
s7	8	0
s8	9	0
s9	10	0
s10	11	0
s11	12	0
s12	13	0
s13	14	0
s14	15	0
s15	16	0
s16	17	0
s17	18	0
s18	19	0
s19	20	0
s20	21	0
s21	22	0
s22	23	0
s23	24	0
s24	25	0
s25	26	0
s26	27	0
s27	28	0
s28	29	0
s29	30	0
s30	31	0
pc		4194312
- Data Segment:** A table showing memory addresses and their values. The address 0x10010000 is highlighted.
 

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
0x10010000	0	0	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000C	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001C	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002C	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003C	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0

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

**Test Segment**

Step	Address	Code	Mean	Source
0	00000000	00000000	00,00,0	0: 00,00,00
1	00000001	00000001	01,01,0	0: 01,00,00
2	00000002	00000002	02,02,0	0: 02,00,00
3	00000003	00000003	03,03,0	0: 03,00,00
4	00000004	00000004	04,04,0	0: 04,00,00
5	00000005	00000005	05,05,0	0: 05,00,00
6	00000006	00000006	06,06,0	0: 06,00,00
7	00000007	00000007	07,07,0	0: 07,00,00
8	00000008	00000008	08,08,0	0: 08,00,00
9	00000009	00000009	09,09,0	0: 09,00,00
10	0000000A	0000000A	0A,0A,0	0: 0A,00,00
11	0000000B	0000000B	0B,0B,0	0: 0B,00,00
12	0000000C	0000000C	0C,0C,0	0: 0C,00,00
13	0000000D	0000000D	0D,0D,0	0: 0D,00,00
14	0000000E	0000000E	0E,0E,0	0: 0E,00,00
15	0000000F	0000000F	0F,0F,0	0: 0F,00,00
16	00000010	00000010	10,10,0	0: 10,00,00
17	00000011	00000011	11,11,0	0: 11,00,00
18	00000012	00000012	12,12,0	0: 12,00,00
19	00000013	00000013	13,13,0	0: 13,00,00
20	00000014	00000014	14,14,0	0: 14,00,00
21	00000015	00000015	15,15,0	0: 15,00,00
22	00000016	00000016	16,16,0	0: 16,00,00
23	00000017	00000017	17,17,0	0: 17,00,00
24	00000018	00000018	18,18,0	0: 18,00,00
25	00000019	00000019	19,19,0	0: 19,00,00
26	0000001A	0000001A	1A,1A,0	0: 1A,00,00
27	0000001B	0000001B	1B,1B,0	0: 1B,00,00
28	0000001C	0000001C	1C,1C,0	0: 1C,00,00
29	0000001D	0000001D	1D,1D,0	0: 1D,00,00
30	0000001E	0000001E	1E,1E,0	0: 1E,00,00
31	0000001F	0000001F	1F,1F,0	0: 1F,00,00

**Labels**

Label	Address
start	00000000
end	0000001F

**Registers**

Name	Number	Value
R0	0	0
R1	1	0
R2	2	0
R3	3	0
R4	4	0
R5	5	0
R6	6	0
R7	7	0
R8	8	0
R9	9	0
R10	10	0
R11	11	0
R12	12	0
R13	13	0
R14	14	0
R15	15	0
R16	16	0
R17	17	0
R18	18	0
R19	19	0
R20	20	0
R21	21	0
R22	22	0
R23	23	0
R24	24	0
R25	25	0
R26	26	0
R27	27	0
R28	28	0
R29	29	0
R30	30	0
R31	31	0

**Data Segment**

Address	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)
00000000	0	0	0	0	0	0	0	0	0
00000001	0	0	0	0	0	0	0	0	0
00000002	0	0	0	0	0	0	0	0	0
00000003	0	0	0	0	0	0	0	0	0
00000004	0	0	0	0	0	0	0	0	0
00000005	0	0	0	0	0	0	0	0	0
00000006	0	0	0	0	0	0	0	0	0
00000007	0	0	0	0	0	0	0	0	0
00000008	0	0	0	0	0	0	0	0	0
00000009	0	0	0	0	0	0	0	0	0
0000000A	0	0	0	0	0	0	0	0	0
0000000B	0	0	0	0	0	0	0	0	0
0000000C	0	0	0	0	0	0	0	0	0
0000000D	0	0	0	0	0	0	0	0	0
0000000E	0	0	0	0	0	0	0	0	0
0000000F	0	0	0	0	0	0	0	0	0
00000010	0	0	0	0	0	0	0	0	0
00000011	0	0	0	0	0	0	0	0	0
00000012	0	0	0	0	0	0	0	0	0
00000013	0	0	0	0	0	0	0	0	0
00000014	0	0	0	0	0	0	0	0	0
00000015	0	0	0	0	0	0	0	0	0
00000016	0	0	0	0	0	0	0	0	0
00000017	0	0	0	0	0	0	0	0	0
00000018	0	0	0	0	0	0	0	0	0
00000019	0	0	0	0	0	0	0	0	0
0000001A	0	0	0	0	0	0	0	0	0
0000001B	0	0	0	0	0	0	0	0	0
0000001C	0	0	0	0	0	0	0	0	0
0000001D	0	0	0	0	0	0	0	0	0
0000001E	0	0	0	0	0	0	0	0	0
0000001F	0	0	0	0	0	0	0	0	0

EXT10000 (data)    ☒ Hexadecimal Addresses    ☐ Hexadecimal Values    ☐ ASCII

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

**Test Segment**

Step	Address	Code	Mean	Source
0	00000000	00000000	00,00,0	0: 00,00,00
1	00000001	00000001	01,01,0	0: 01,00,00
2	00000002	00000002	02,02,0	0: 02,00,00
3	00000003	00000003	03,03,0	0: 03,00,00
4	00000004	00000004	04,04,0	0: 04,00,00
5	00000005	00000005	05,05,0	0: 05,00,00
6	00000006	00000006	06,06,0	0: 06,00,00
7	00000007	00000007	07,07,0	0: 07,00,00
8	00000008	00000008	08,08,0	0: 08,00,00
9	00000009	00000009	09,09,0	0: 09,00,00
10	0000000A	0000000A	0A,0A,0	0: 0A,00,00
11	0000000B	0000000B	0B,0B,0	0: 0B,00,00
12	0000000C	0000000C	0C,0C,0	0: 0C,00,00
13	0000000D	0000000D	0D,0D,0	0: 0D,00,00
14	0000000E	0000000E	0E,0E,0	0: 0E,00,00
15	0000000F	0000000F	0F,0F,0	0: 0F,00,00
16	00000010	00000010	10,10,0	0: 10,00,00
17	00000011	00000011	11,11,0	0: 11,00,00
18	00000012	00000012	12,12,0	0: 12,00,00
19	00000013	00000013	13,13,0	0: 13,00,00
20	00000014	00000014	14,14,0	0: 14,00,00
21	00000015	00000015	15,15,0	0: 15,00,00
22	00000016	00000016	16,16,0	0: 16,00,00
23	00000017	00000017	17,17,0	0: 17,00,00
24	00000018	00000018	18,18,0	0: 18,00,00
25	00000019	00000019	19,19,0	0: 19,00,00
26	0000001A	0000001A	1A,1A,0	0: 1A,00,00
27	0000001B	0000001B	1B,1B,0	0: 1B,00,00
28	0000001C	0000001C	1C,1C,0	0: 1C,00,00
29	0000001D	0000001D	1D,1D,0	0: 1D,00,00
30	0000001E	0000001E	1E,1E,0	0: 1E,00,00
31	0000001F	0000001F	1F,1F,0	0: 1F,00,00

**Labels**

Label	Address
start	00000000
end	0000001F

**Registers**

Name	Number	Value
R0	0	0
R1	1	0
R2	2	0
R3	3	0
R4	4	0
R5	5	0
R6	6	0
R7	7	0
R8	8	0
R9	9	0
R10	10	0
R11	11	0
R12	12	0
R13	13	0
R14	14	0
R15	15	0
R16	16	0
R17	17	0
R18	18	0
R19	19	0
R20	20	0
R21	21	0
R22	22	0
R23	23	0
R24	24	0
R25	25	0
R26	26	0
R27	27	0
R28	28	0
R29	29	0
R30	30	0
R31	31	0

**Data Segment**

Address	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)
00000000	0	0	0	0	0	0	0	0	0
00000001	0	0	0	0	0	0	0	0	0
00000002	0	0	0	0	0	0	0	0	0
00000003	0	0	0	0	0	0	0	0	0
00000004	0	0	0	0	0	0	0	0	0
00000005	0	0	0	0	0	0	0	0	0
00000006	0	0	0	0	0	0	0	0	0
00000007	0	0	0	0	0	0	0	0	0
00000008	0	0	0	0	0	0	0	0	0
00000009	0	0	0	0	0	0	0	0	0
0000000A	0	0	0	0	0	0	0	0	0
0000000B	0	0	0	0	0	0	0	0	0
0000000C	0	0	0	0	0	0	0	0	0
0000000D	0	0	0	0	0	0	0	0	0
0000000E	0	0	0	0	0	0	0	0	0
0000000F	0	0	0	0	0	0	0	0	0
00000010	0	0	0	0	0	0	0	0	0
00000011	0	0	0	0	0	0	0	0	0
00000012	0	0	0	0	0	0	0	0	0
00000013	0	0	0	0	0	0	0	0	0
00000014	0	0	0	0	0	0	0	0	0
00000015	0	0	0	0	0	0	0	0	0
00000016	0	0	0	0	0	0	0	0	0
00000017	0	0	0	0	0	0	0	0	0
00000018	0	0	0	0	0	0	0	0	0
00000019	0	0	0	0	0	0	0	0	0
0000001A	0	0	0	0	0	0	0	0	0
0000001B	0	0	0	0	0	0	0	0	0
0000001C	0	0	0	0	0	0	0	0	0
0000001D	0	0	0	0	0	0	0	0	0
0000001E	0	0	0	0	0	0	0	0	0
0000001F	0	0	0	0	0	0	0	0	0

EXT10000 (data)    ☒ Hexadecimal Addresses    ☐ Hexadecimal Values    ☐ ASCII

- Lệnh 5: addi t3, zero, 1 # Gán z=1

**Test Segment**

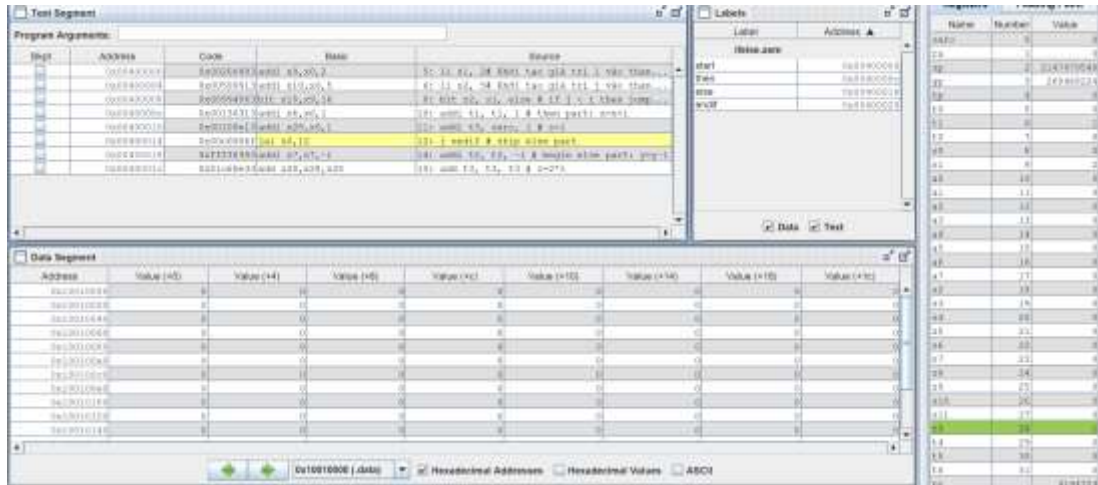
Step	Address	Code	Mean	Source
0	00000000	00000000	00,00,0	0: 00,00,00

- 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:

```
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 s3, 9
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:
```



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

The screenshot shows the Immortal debugger interface. The **Text Segment** window displays the following assembly code:

Dispt	Address	Code	Basic	Source
0x10010000	0x10010000	li s3, 9	li s3, 9	
0x10010001	0x10010001	li s2, 0	li s2, 0	
0x10010002	0x10010002	li s1, 0	li s1, 0	
0x10010003	0x10010003	li s0, 0	li s0, 0	
0x10010004	0x10010004	li s3, 9	li s3, 9	
0x10010005	0x10010005	li s2, 0	li s2, 0	
0x10010006	0x10010006	li s1, 0	li s1, 0	
0x10010007	0x10010007	li s0, 0	li s0, 0	
0x10010008	0x10010008	li s3, 9	li s3, 9	
0x10010009	0x10010009	li s2, 0	li s2, 0	
0x1001000A	0x1001000A	li s1, 0	li s1, 0	
0x1001000B	0x1001000B	li s0, 0	li s0, 0	
0x1001000C	0x1001000C	li s3, 9	li s3, 9	
0x1001000D	0x1001000D	li s2, 0	li s2, 0	
0x1001000E	0x1001000E	li s1, 0	li s1, 0	
0x1001000F	0x1001000F	li s0, 0	li s0, 0	
0x10010010	0x10010010	li s3, 9	li s3, 9	
0x10010011	0x10010011	li s2, 0	li s2, 0	
0x10010012	0x10010012	li s1, 0	li s1, 0	
0x10010013	0x10010013	li s0, 0	li s0, 0	
0x10010014	0x10010014	li s3, 9	li s3, 9	
0x10010015	0x10010015	li s2, 0	li s2, 0	
0x10010016	0x10010016	li s1, 0	li s1, 0	
0x10010017	0x10010017	li s0, 0	li s0, 0	
0x10010018	0x10010018	li s3, 9	li s3, 9	
0x10010019	0x10010019	li s2, 0	li s2, 0	
0x1001001A	0x1001001A	li s1, 0	li s1, 0	
0x1001001B	0x1001001B	li s0, 0	li s0, 0	
0x1001001C	0x1001001C	li s3, 9	li s3, 9	
0x1001001D	0x1001001D	li s2, 0	li s2, 0	
0x1001001E	0x1001001E	li s1, 0	li s1, 0	
0x1001001F	0x1001001F	li s0, 0	li s0, 0	

The **Data Segment** window shows memory addresses from 0x10010000 to 0x10010100, all containing the value 0. The **Registers** window on the right shows the state of various registers, with s3 highlighted and containing the value 9.

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

The screenshot shows the Immortal debugger interface. The **Text Segment** window displays the following assembly code:

Dispt	Address	Code	Basic	Source
0x10010000	0x10010000	li s3, 9	li s3, 9	
0x10010001	0x10010001	la s2, A	la s2, A	
0x10010002	0x10010002	li s1, 0	li s1, 0	
0x10010003	0x10010003	li s0, 0	li s0, 0	
0x10010004	0x10010004	li s3, 9	li s3, 9	
0x10010005	0x10010005	li s2, 0	li s2, 0	
0x10010006	0x10010006	li s1, 0	li s1, 0	
0x10010007	0x10010007	li s0, 0	li s0, 0	
0x10010008	0x10010008	li s3, 9	li s3, 9	
0x10010009	0x10010009	li s2, 0	li s2, 0	
0x1001000A	0x1001000A	li s1, 0	li s1, 0	
0x1001000B	0x1001000B	li s0, 0	li s0, 0	
0x1001000C	0x1001000C	li s3, 9	li s3, 9	
0x1001000D	0x1001000D	li s2, 0	li s2, 0	
0x1001000E	0x1001000E	li s1, 0	li s1, 0	
0x1001000F	0x1001000F	li s0, 0	li s0, 0	
0x10010010	0x10010010	li s3, 9	li s3, 9	
0x10010011	0x10010011	li s2, 0	li s2, 0	
0x10010012	0x10010012	li s1, 0	li s1, 0	
0x10010013	0x10010013	li s0, 0	li s0, 0	
0x10010014	0x10010014	li s3, 9	li s3, 9	
0x10010015	0x10010015	li s2, 0	li s2, 0	
0x10010016	0x10010016	li s1, 0	li s1, 0	
0x10010017	0x10010017	li s0, 0	li s0, 0	
0x10010018	0x10010018	li s3, 9	li s3, 9	
0x10010019	0x10010019	li s2, 0	li s2, 0	
0x1001001A	0x1001001A	li s1, 0	li s1, 0	
0x1001001B	0x1001001B	li s0, 0	li s0, 0	
0x1001001C	0x1001001C	li s3, 9	li s3, 9	
0x1001001D	0x1001001D	li s2, 0	li s2, 0	
0x1001001E	0x1001001E	li s1, 0	li s1, 0	
0x1001001F	0x1001001F	li s0, 0	li s0, 0	

The **Data Segment** window shows memory addresses from 0x10010000 to 0x10010100, all containing the value 0. The **Registers** window on the right shows the state of various registers, with s2 highlighted and containing the value 0.

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

**Text Segment**

Addr	Address	Code	Basic	Source
0x10010000	0x10010000	li s4, 1	li s4, 1	
0x10010001	0x10010001	li s5, 0	li s5, 0	
0x10010002	0x10010002	li s6, 0	li s6, 0	
0x10010003	0x10010003	li s7, 0	li s7, 0	
0x10010004	0x10010004	li s8, 0	li s8, 0	
0x10010005	0x10010005	li s9, 0	li s9, 0	
0x10010006	0x10010006	li s10, 0	li s10, 0	
0x10010007	0x10010007	li s11, 0	li s11, 0	
0x10010008	0x10010008	li s12, 0	li s12, 0	
0x10010009	0x10010009	li s13, 0	li s13, 0	
0x1001000A	0x1001000A	li s14, 0	li s14, 0	
0x1001000B	0x1001000B	li s15, 0	li s15, 0	
0x1001000C	0x1001000C	li s16, 0	li s16, 0	
0x1001000D	0x1001000D	li s17, 0	li s17, 0	
0x1001000E	0x1001000E	li s18, 0	li s18, 0	
0x1001000F	0x1001000F	li s19, 0	li s19, 0	
0x10010010	0x10010010	li s20, 0	li s20, 0	
0x10010011	0x10010011	li s21, 0	li s21, 0	
0x10010012	0x10010012	li s22, 0	li s22, 0	
0x10010013	0x10010013	li s23, 0	li s23, 0	
0x10010014	0x10010014	li s24, 0	li s24, 0	
0x10010015	0x10010015	li s25, 0	li s25, 0	
0x10010016	0x10010016	li s26, 0	li s26, 0	
0x10010017	0x10010017	li s27, 0	li s27, 0	
0x10010018	0x10010018	li s28, 0	li s28, 0	
0x10010019	0x10010019	li s29, 0	li s29, 0	
0x1001001A	0x1001001A	li s30, 0	li s30, 0	
0x1001001B	0x1001001B	li s31, 0	li s31, 0	
0x1001001C	0x1001001C	li s32, 0	li s32, 0	
0x1001001D	0x1001001D	li s33, 0	li s33, 0	
0x1001001E	0x1001001E	li s34, 0	li s34, 0	
0x1001001F	0x1001001F	li s35, 0	li s35, 0	
0x10010020	0x10010020	li s36, 0	li s36, 0	
0x10010021	0x10010021	li s37, 0	li s37, 0	
0x10010022	0x10010022	li s38, 0	li s38, 0	
0x10010023	0x10010023	li s39, 0	li s39, 0	
0x10010024	0x10010024	li s40, 0	li s40, 0	
0x10010025	0x10010025	li s41, 0	li s41, 0	
0x10010026	0x10010026	li s42, 0	li s42, 0	
0x10010027	0x10010027	li s43, 0	li s43, 0	
0x10010028	0x10010028	li s44, 0	li s44, 0	
0x10010029	0x10010029	li s45, 0	li s45, 0	
0x1001002A	0x1001002A	li s46, 0	li s46, 0	
0x1001002B	0x1001002B	li s47, 0	li s47, 0	
0x1001002C	0x1001002C	li s48, 0	li s48, 0	
0x1001002D	0x1001002D	li s49, 0	li s49, 0	
0x1001002E	0x1001002E	li s50, 0	li s50, 0	
0x1001002F	0x1001002F	li s51, 0	li s51, 0	
0x10010030	0x10010030	li s52, 0	li s52, 0	
0x10010031	0x10010031	li s53, 0	li s53, 0	
0x10010032	0x10010032	li s54, 0	li s54, 0	
0x10010033	0x10010033	li s55, 0	li s55, 0	
0x10010034	0x10010034	li s56, 0	li s56, 0	
0x10010035	0x10010035	li s57, 0	li s57, 0	
0x10010036	0x10010036	li s58, 0	li s58, 0	
0x10010037	0x10010037	li s59, 0	li s59, 0	
0x10010038	0x10010038	li s60, 0	li s60, 0	
0x10010039	0x10010039	li s61, 0	li s61, 0	
0x1001003A	0x1001003A	li s62, 0	li s62, 0	
0x1001003B	0x1001003B	li s63, 0	li s63, 0	
0x1001003C	0x1001003C	li s64, 0	li s64, 0	
0x1001003D	0x1001003D	li s65, 0	li s65, 0	
0x1001003E	0x1001003E	li s66, 0	li s66, 0	
0x1001003F	0x1001003F	li s67, 0	li s67, 0	
0x10010040	0x10010040	li s68, 0	li s68, 0	
0x10010041	0x10010041	li s69, 0	li s69, 0	
0x10010042	0x10010042	li s70, 0	li s70, 0	
0x10010043	0x10010043	li s71, 0	li s71, 0	
0x10010044	0x10010044	li s72, 0	li s72, 0	
0x10010045	0x10010045	li s73, 0	li s73, 0	
0x10010046	0x10010046	li s74, 0	li s74, 0	
0x10010047	0x10010047	li s75, 0	li s75, 0	
0x10010048	0x10010048	li s76, 0	li s76, 0	
0x10010049	0x10010049	li s77, 0	li s77, 0	
0x1001004A	0x1001004A	li s78, 0	li s78, 0	
0x1001004B	0x1001004B	li s79, 0	li s79, 0	
0x1001004C	0x1001004C	li s80, 0	li s80, 0	
0x1001004D	0x1001004D	li s81, 0	li s81, 0	
0x1001004E	0x1001004E	li s82, 0	li s82, 0	
0x1001004F	0x1001004F	li s83, 0	li s83, 0	
0x10010050	0x10010050	li s84, 0	li s84, 0	
0x10010051	0x10010051	li s85, 0	li s85, 0	
0x10010052	0x10010052	li s86, 0	li s86, 0	
0x10010053	0x10010053	li s87, 0	li s87, 0	
0x10010054	0x10010054	li s88, 0	li s88, 0	
0x10010055	0x10010055	li s89, 0	li s89, 0	
0x10010056	0x10010056	li s90, 0	li s90, 0	
0x10010057	0x10010057	li s91, 0	li s91, 0	
0x10010058	0x10010058	li s92, 0	li s92, 0	
0x10010059	0x10010059	li s93, 0	li s93, 0	
0x1001005A	0x1001005A	li s94, 0	li s94, 0	
0x1001005B	0x1001005B	li s95, 0	li s95, 0	
0x1001005C	0x1001005C	li s96, 0	li s96, 0	
0x1001005D	0x1001005D	li s97, 0	li s97, 0	
0x1001005E	0x1001005E	li s98, 0	li s98, 0	
0x1001005F	0x1001005F	li s99, 0	li s99, 0	
0x10010060	0x10010060	li s100, 0	li s100, 0	
0x10010061	0x10010061	li s101, 0	li s101, 0	
0x10010062	0x10010062	li s102, 0	li s102, 0	
0x10010063	0x10010063	li s103, 0	li s103, 0	
0x10010064	0x10010064	li s104, 0	li s104, 0	
0x10010065	0x10010065	li s105, 0	li s105, 0	
0x10010066	0x10010066	li s106, 0	li s106, 0	
0x10010067	0x10010067	li s107, 0	li s107, 0	
0x10010068	0x10010068	li s108, 0	li s108, 0	
0x10010069	0x10010069	li s109, 0	li s109, 0	
0x1001006A	0x1001006A	li s110, 0	li s110, 0	
0x1001006B	0x1001006B	li s111, 0	li s111, 0	
0x1001006C	0x1001006C	li s112, 0	li s112, 0	
0x1001006D	0x1001006D	li s113, 0	li s113, 0	
0x1001006E	0x1001006E	li s114, 0	li s114, 0	
0x1001006F	0x1001006F	li s115, 0	li s115, 0	
0x10010070	0x10010070	li s116, 0	li s116, 0	
0x10010071	0x10010071	li s117, 0	li s117, 0	
0x10010072	0x10010072	li s118, 0	li s118, 0	
0x10010073	0x10010073	li s119, 0	li s119, 0	
0x10010074	0x10010074	li s120, 0	li s120, 0	
0x10010075	0x10010075	li s121, 0	li s121, 0	
0x10010076	0x10010076	li s122, 0	li s122, 0	
0x10010077	0x10010077	li s123, 0	li s123, 0	
0x10010078	0x10010078	li s124, 0	li s124, 0	
0x10010079	0x10010079	li s125, 0	li s125, 0	
0x1001007A	0x1001007A	li s126, 0	li s126, 0	
0x1001007B	0x1001007B	li s127, 0	li s127, 0	
0x1001007C	0x1001007C	li s128, 0	li s128, 0	
0x1001007D	0x1001007D	li s129, 0	li s129, 0	
0x1001007E	0x1001007E	li s130, 0	li s130, 0	
0x1001007F	0x1001007F	li s131, 0	li s131, 0	
0x10010080	0x10010080	li s132, 0	li s132, 0	
0x10010081	0x10010081	li s133, 0	li s133, 0	
0x10010082	0x10010082	li s134, 0	li s134, 0	
0x10010083	0x10010083	li s135, 0	li s135, 0	
0x10010084	0x10010084	li s136, 0	li s136, 0	
0x10010085	0x10010085	li s137, 0	li s137, 0	
0x10010086	0x10010086	li s138, 0	li s138, 0	
0x10010087	0x10010087	li s139, 0	li s139, 0	
0x10010088	0x10010088	li s140, 0	li s140, 0	
0x10010089	0x10010089	li s141, 0	li s141, 0	
0x1001008A	0x1001008A	li s142, 0	li s142, 0	
0x1001008B	0x1001008B	li s143, 0	li s143, 0	
0x1001008C	0x1001008C	li s144, 0	li s144, 0	
0x1001008D	0x1001008D	li s145, 0	li s145, 0	
0x1001008E	0x1001008E	li s146, 0	li s146, 0	
0x1001008F	0x1001008F	li s147, 0	li s147, 0	
0x10010090	0x10010090	li s148, 0	li s148, 0	
0x10010091	0x10010091	li s149, 0	li s149, 0	
0x10010092	0x10010092	li s150, 0	li s150, 0	
0x10010093	0x10010093	li s151, 0	li s151, 0	
0x10010094	0x10010094	li s152, 0	li s152, 0	
0x10010095	0x10010095	li s153, 0	li s153, 0	
0x10010096	0x10010096	li s154, 0	li s154, 0	
0x10010097	0x10010097	li s155, 0	li s155, 0	
0x10010098	0x10010098	li s156, 0	li s156, 0	
0x10010099	0x10010099	li s157, 0	li s157, 0	
0x1001009A	0x1001009A	li s158, 0	li s158, 0	
0x1001009B	0x1001009B	li s159, 0	li s159, 0	
0x1001009C	0x1001009C	li s160, 0	li s160, 0	
0x1001009D	0x1001009D	li s161, 0	li s161, 0	
0x1001009E	0x1001009E	li s162, 0	li s162, 0	
0x1001009F	0x1001009F	li s163, 0	li s163, 0	
0x100100A0	0x100100A0	li s164, 0	li s164, 0	
0x100100A1	0x100100A1	li s165, 0	li s165, 0	
0x100100A2	0x100100A2	li s166, 0	li s166, 0	
0x100100A3	0x100100A3	li s167, 0	li s167, 0	
0x100100A4	0x100100A4	li s168, 0	li s168, 0	
0x100100A5	0x100100A5	li s169, 0	li s169, 0	
0x100100A6	0x100100A6	li s170, 0	li s170, 0	
0x100100A7	0x100100A7	li s171, 0	li s171, 0	
0x100100A8	0x100100A8	li s172, 0	li s172, 0	
0x100100A9	0x100100A9	li s173, 0	li s173, 0	
0x100100AA	0x100100AA	li s174, 0	li s174, 0	
0x100100AB	0x100100AB	li s175, 0	li s175	

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

The screenshot shows a debugger interface with three main panels: Program Arguments, Assembly, and Registers.

**Program Arguments:**

Op1	Address	Code	Base	Source
	0x00000000	0x00000000 addl r19, r0, 0		r1: r1, r0, 0
	0x00000001	0x00000001 mulpl r19, r0, 0		r1: r1, r0, 0
	0x00000002	0x00000002 addl r19, r0, 0		r1: r1, r0, 0
	0x00000003	0x00000003 addl r19, r0, 0		r1: r1, r0, 0
	0x00000004	0x00000004 addl r19, r0, 0		r1: r1, r0, 0
	0x00000005	0x00000005 addl r19, r0, 0		r1: r1, r0, 0
	0x00000006	0x00000006 addl r19, r0, 0		r1: r1, r0, 0
	0x00000007	0x00000007 addl r19, r0, 0		r1: r1, r0, 0
	0x00000008	0x00000008 addl r19, r0, 0		r1: r1, r0, 0
	0x00000009	0x00000009 addl r19, r0, 0		r1: r1, r0, 0
	0x0000000A	0x0000000A addl r19, r0, 0		r1: r1, r0, 0
	0x0000000B	0x0000000B addl r19, r0, 0		r1: r1, r0, 0
	0x0000000C	0x0000000C addl r19, r0, 0		r1: r1, r0, 0
	0x0000000D	0x0000000D addl r19, r0, 0		r1: r1, r0, 0
	0x0000000E	0x0000000E addl r19, r0, 0		r1: r1, r0, 0
	0x0000000F	0x0000000F addl r19, r0, 0		r1: r1, r0, 0
	0x00000010	0x00000010 addl r19, r0, 0		r1: r1, r0, 0
	0x00000011	0x00000011 addl r19, r0, 0		r1: r1, r0, 0
	0x00000012	0x00000012 addl r19, r0, 0		r1: r1, r0, 0
	0x00000013	0x00000013 addl r19, r0, 0		r1: r1, r0, 0
	0x00000014	0x00000014 addl r19, r0, 0		r1: r1, r0, 0
	0x00000015	0x00000015 addl r19, r0, 0		r1: r1, r0, 0
	0x00000016	0x00000016 addl r19, r0, 0		r1: r1, r0, 0
	0x00000017	0x00000017 addl r19, r0, 0		r1: r1, r0, 0
	0x00000018	0x00000018 addl r19, r0, 0		r1: r1, r0, 0
	0x00000019	0x00000019 addl r19, r0, 0		r1: r1, r0, 0
	0x0000001A	0x0000001A addl r19, r0, 0		r1: r1, r0, 0
	0x0000001B	0x0000001B addl r19, r0, 0		r1: r1, r0, 0
	0x0000001C	0x0000001C addl r19, r0, 0		r1: r1, r0, 0
	0x0000001D	0x0000001D addl r19, r0, 0		r1: r1, r0, 0
	0x0000001E	0x0000001E addl r19, r0, 0		r1: r1, r0, 0
	0x0000001F	0x0000001F addl r19, r0, 0		r1: r1, r0, 0

**Registers:**

Name	Number	Value
r0	0	0
r1	1	0
r2	2	2157479340
r3	3	269468224
r4	4	0
r5	5	0
r6	6	0
r7	7	0
r8	8	0
r9	9	0
r10	10	0
r11	11	0
r12	12	0
r13	13	0
r14	14	0
r15	15	0
r16	16	0
r17	17	0
r18	18	269468224
r19	19	0
r20	20	0
r21	21	0
r22	22	0
r23	23	0
r24	24	0
r25	25	0
r26	26	0
r27	27	0
r28	28	0
r29	29	0
r30	30	0
r31	31	0
PC		4194340

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

The screenshot shows a debugger interface with three main panels: Program Arguments, Assembly, and Registers.

**Program Arguments:**

Op1	Address	Code	Base	Source
	0x00000000	0x00000000 addl r19, r0, 0		r1: r1, r0, 0
	0x00000001	0x00000001 mulpl r19, r0, 0		r1: r1, r0, 0
	0x00000002	0x00000002 addl r19, r0, 0		r1: r1, r0, 0
	0x00000003	0x00000003 addl r19, r0, 0		r1: r1, r0, 0
	0x00000004	0x00000004 addl r19, r0, 0		r1: r1, r0, 0
	0x00000005	0x00000005 addl r19, r0, 0		r1: r1, r0, 0
	0x00000006	0x00000006 addl r19, r0, 0		r1: r1, r0, 0
	0x00000007	0x00000007 addl r19, r0, 0		r1: r1, r0, 0
	0x00000008	0x00000008 addl r19, r0, 0		r1: r1, r0, 0
	0x00000009	0x00000009 addl r19, r0, 0		r1: r1, r0, 0
	0x0000000A	0x0000000A addl r19, r0, 0		r1: r1, r0, 0
	0x0000000B	0x0000000B addl r19, r0, 0		r1: r1, r0, 0
	0x0000000C	0x0000000C addl r19, r0, 0		r1: r1, r0, 0
	0x0000000D	0x0000000D addl r19, r0, 0		r1: r1, r0, 0
	0x0000000E	0x0000000E addl r19, r0, 0		r1: r1, r0, 0
	0x0000000F	0x0000000F addl r19, r0, 0		r1: r1, r0, 0
	0x00000010	0x00000010 addl r19, r0, 0		r1: r1, r0, 0
	0x00000011	0x00000011 addl r19, r0, 0		r1: r1, r0, 0
	0x00000012	0x00000012 addl r19, r0, 0		r1: r1, r0, 0
	0x00000013	0x00000013 addl r19, r0, 0		r1: r1, r0, 0
	0x00000014	0x00000014 addl r19, r0, 0		r1: r1, r0, 0
	0x00000015	0x00000015 addl r19, r0, 0		r1: r1, r0, 0
	0x00000016	0x00000016 addl r19, r0, 0		r1: r1, r0, 0
	0x00000017	0x00000017 addl r19, r0, 0		r1: r1, r0, 0
	0x00000018	0x00000018 addl r19, r0, 0		r1: r1, r0, 0
	0x00000019	0x00000019 addl r19, r0, 0		r1: r1, r0, 0
	0x0000001A	0x0000001A addl r19, r0, 0		r1: r1, r0, 0
	0x0000001B	0x0000001B addl r19, r0, 0		r1: r1, r0, 0
	0x0000001C	0x0000001C addl r19, r0, 0		r1: r1, r0, 0
	0x0000001D	0x0000001D addl r19, r0, 0		r1: r1, r0, 0
	0x0000001E	0x0000001E addl r19, r0, 0		r1: r1, r0, 0
	0x0000001F	0x0000001F addl r19, r0, 0		r1: r1, r0, 0

**Registers:**

Name	Number	Value
r0	0	0
r1	1	0
r2	2	2157479340
r3	3	269468224
r4	4	0
r5	5	0
r6	6	0
r7	7	0
r8	8	0
r9	9	0
r10	10	0
r11	11	0
r12	12	0
r13	13	0
r14	14	0
r15	15	0
r16	16	0
r17	17	0
r18	18	269468224
r19	19	0
r20	20	0
r21	21	0
r22	22	0
r23	23	0
r24	24	0
r25	25	0
r26	26	0
r27	27	0
r28	28	0
r29	29	0
r30	30	0
r31	31	0
PC		4194340

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

The screenshot displays a MIPS assembler interface with three main panels:

- Text Segment:** Shows the assembly code. Instruction 9, `add t1, t1, s2`, is highlighted in yellow. The source code is:
 

```

      00000000: addi s2, s0, 4
      00000001: lwf s1, 0x17anipr s1, 64326
      00000002: lwf s2, 0x17anipr s2, 64326
      00000003: lwf s3, 0x17anipr s3, 64326
      00000004: lwf s4, 0x17anipr s4, 64326
      00000005: lwf s5, 0x17anipr s5, 64326
      00000006: lwf s6, 0x17anipr s6, 64326
      00000007: lwf s7, 0x17anipr s7, 64326
      00000008: lwf s8, 0x17anipr s8, 64326
      00000009: lwf s9, 0x17anipr s9, 64326
      0000000A: lwf s10, 0x17anipr s10, 64326
      0000000B: lwf s11, 0x17anipr s11, 64326
      0000000C: lwf s12, 0x17anipr s12, 64326
      0000000D: lwf s13, 0x17anipr s13, 64326
      0000000E: lwf s14, 0x17anipr s14, 64326
      0000000F: lwf s15, 0x17anipr s15, 64326
      00000010: lwf s16, 0x17anipr s16, 64326
      00000011: lwf s17, 0x17anipr s17, 64326
      00000012: lwf s18, 0x17anipr s18, 64326
      00000013: lwf s19, 0x17anipr s19, 64326
      00000014: lwf s20, 0x17anipr s20, 64326
      00000015: lwf s21, 0x17anipr s21, 64326
      00000016: lwf s22, 0x17anipr s22, 64326
      00000017: lwf s23, 0x17anipr s23, 64326
      00000018: lwf s24, 0x17anipr s24, 64326
      00000019: lwf s25, 0x17anipr s25, 64326
      0000001A: lwf s26, 0x17anipr s26, 64326
      0000001B: lwf s27, 0x17anipr s27, 64326
      0000001C: lwf s28, 0x17anipr s28, 64326
      0000001D: lwf s29, 0x17anipr s29, 64326
      0000001E: lwf s30, 0x17anipr s30, 64326
      0000001F: lwf s31, 0x17anipr s31, 64326
      
```
- Data Segment:** Shows memory addresses from 0x1000000 to 0x1000040. The values are mostly 0, except for 0x1000000 which is 1, 0x1000001 which is 4, and 0x1000002 which is 16.
- Control and Status:** Shows register values. The `t1` register is highlighted in green and contains the value 0x1000000.

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

The screenshot displays a MIPS assembler interface with three main panels:

- Text Segment:** Shows the assembly code. Instruction 10, `lw t0, 0(t1)`, is highlighted in yellow. The source code is:
 

```

      00000000: addi s2, s0, 4
      00000001: lwf s1, 0x17anipr s1, 64326
      00000002: lwf s2, 0x17anipr s2, 64326
      00000003: lwf s3, 0x17anipr s3, 64326
      00000004: lwf s4, 0x17anipr s4, 64326
      00000005: lwf s5, 0x17anipr s5, 64326
      00000006: lwf s6, 0x17anipr s6, 64326
      00000007: lwf s7, 0x17anipr s7, 64326
      00000008: lwf s8, 0x17anipr s8, 64326
      00000009: lwf s9, 0x17anipr s9, 64326
      0000000A: lwf s10, 0x17anipr s10, 64326
      0000000B: lwf s11, 0x17anipr s11, 64326
      0000000C: lwf s12, 0x17anipr s12, 64326
      0000000D: lwf s13, 0x17anipr s13, 64326
      0000000E: lwf s14, 0x17anipr s14, 64326
      0000000F: lwf s15, 0x17anipr s15, 64326
      00000010: lwf s16, 0x17anipr s16, 64326
      00000011: lwf s17, 0x17anipr s17, 64326
      00000012: lwf s18, 0x17anipr s18, 64326
      00000013: lwf s19, 0x17anipr s19, 64326
      00000014: lwf s20, 0x17anipr s20, 64326
      00000015: lwf s21, 0x17anipr s21, 64326
      00000016: lwf s22, 0x17anipr s22, 64326
      00000017: lwf s23, 0x17anipr s23, 64326
      00000018: lwf s24, 0x17anipr s24, 64326
      00000019: lwf s25, 0x17anipr s25, 64326
      0000001A: lwf s26, 0x17anipr s26, 64326
      0000001B: lwf s27, 0x17anipr s27, 64326
      0000001C: lwf s28, 0x17anipr s28, 64326
      0000001D: lwf s29, 0x17anipr s29, 64326
      0000001E: lwf s30, 0x17anipr s30, 64326
      0000001F: lwf s31, 0x17anipr s31, 64326
      
```
- Data Segment:** Shows memory addresses from 0x1000000 to 0x1000040. The values are mostly 0, except for 0x1000000 which is 1, 0x1000001 which is 4, and 0x1000002 which is 16.
- Control and Status:** Shows register values. The `t0` register is highlighted in green and contains the value 0x1000000.

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





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

The screenshot shows a debugger interface with three main panels:

- Text Segment:** Displays assembly code. The highlighted instruction is `j loop # quay lại đầu vòng lặp` at address `0x00000013`. The code is in ARM assembly, with comments in Vietnamese.
- Data Segment:** A table showing memory values at various addresses. The values are mostly 0, with some non-zero values at higher addresses.
- Registers:** A table showing the state of various registers. The `r10` register is highlighted, showing a value of `0x00000013`.

- Lập lại các bước như trên sẽ cho ta kết quả cuối cùng  $s5(\text{sum}) = 45$

The screenshot shows a debugger interface with three main panels:

- Text Segment:** Displays assembly code. The highlighted instruction is `j loop # quay lại đầu vòng lặp` at address `0x00000013`. The code is in ARM assembly, with comments in Vietnamese.
- Data Segment:** A table showing memory values at various addresses. The values are mostly 0, with some non-zero values at higher addresses.
- Registers:** A table showing the state of various registers. The `r10` register is highlighted, showing a value of `0x00000013`.

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

```

1  # Laboratory 3, Home Assignment 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  $s5(\text{sum}) = 25$

The screenshot shows the RARS MIPS simulator interface. The top window displays the assembly code, and the bottom window shows the state of registers and memory.

**Assembly Code Window:**

Address	Code	Base	Source
0x00000000	li s3, 5		
0x00000001	la s2, A		
0x00000002	li s4, 1		
0x00000003	li s1, 0		
0x00000004	li s5, 0		
0x00000005	loop:		
0x00000006	bge s1, s3, endloop		
0x00000007	add t1, s1, s1		
0x00000008	add t1, t1, t1		
0x00000009	add t1, t1, s2		
0x0000000A	lw t0, 0(t1)		
0x0000000B	add s5, s5, t0		
0x0000000C	add s1, s1, s4		
0x0000000D	j loop		
0x0000000E	endloop:		

**Registers Window:**

Name	Register	Value
\$zero	\$0	0
\$at	\$1	0
\$v0	\$2	2147483648
\$v1	\$3	2004000000
\$a0	\$4	0
\$a1	\$5	0
\$a2	\$6	2004000000
\$a3	\$7	0
\$t0	\$8	0
\$t1	\$9	4
\$t2	\$10	0
\$t3	\$11	0
\$t4	\$12	0
\$t5	\$13	0
\$t6	\$14	0
\$t7	\$15	0
\$s0	\$16	0
\$s1	\$17	0
\$s2	\$18	2004000000
\$s3	\$19	5
\$s4	\$20	1
\$s5	\$21	25
\$s6	\$22	0
\$s7	\$23	0
\$s8	\$24	0
\$s9	\$25	0
\$s10	\$26	0
\$s11	\$27	0
\$s12	\$28	0
\$s13	\$29	0
\$s14	\$30	0
\$s15	\$31	0

**Memory Window:**

Address	Value (Hex)	Value (Dec)	Value (Hex)	Value (Dec)	Value (Hex)	Value (Dec)	Value (Hex)	Value (Dec)
0x00000000	00000001	1	00000003	3	00000005	5	00000007	7
0x00000004	00000009	9	00000000	0	00000000	0	00000000	0
0x00000008	00000000	0	00000000	0	00000000	0	00000000	0
0x0000000C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000010	00000000	0	00000000	0	00000000	0	00000000	0
0x00000014	00000000	0	00000000	0	00000000	0	00000000	0
0x00000018	00000000	0	00000000	0	00000000	0	00000000	0
0x0000001C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000020	00000000	0	00000000	0	00000000	0	00000000	0
0x00000024	00000000	0	00000000	0	00000000	0	00000000	0
0x00000028	00000000	0	00000000	0	00000000	0	00000000	0
0x0000002C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000030	00000000	0	00000000	0	00000000	0	00000000	0
0x00000034	00000000	0	00000000	0	00000000	0	00000000	0
0x00000038	00000000	0	00000000	0	00000000	0	00000000	0
0x0000003C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000040	00000000	0	00000000	0	00000000	0	00000000	0
0x00000044	00000000	0	00000000	0	00000000	0	00000000	0
0x00000048	00000000	0	00000000	0	00000000	0	00000000	0
0x0000004C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000050	00000000	0	00000000	0	00000000	0	00000000	0
0x00000054	00000000	0	00000000	0	00000000	0	00000000	0
0x00000058	00000000	0	00000000	0	00000000	0	00000000	0
0x0000005C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000060	00000000	0	00000000	0	00000000	0	00000000	0
0x00000064	00000000	0	00000000	0	00000000	0	00000000	0
0x00000068	00000000	0	00000000	0	00000000	0	00000000	0
0x0000006C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000070	00000000	0	00000000	0	00000000	0	00000000	0
0x00000074	00000000	0	00000000	0	00000000	0	00000000	0
0x00000078	00000000	0	00000000	0	00000000	0	00000000	0
0x0000007C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000080	00000000	0	00000000	0	00000000	0	00000000	0
0x00000084	00000000	0	00000000	0	00000000	0	00000000	0
0x00000088	00000000	0	00000000	0	00000000	0	00000000	0
0x0000008C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000090	00000000	0	00000000	0	00000000	0	00000000	0
0x00000094	00000000	0	00000000	0	00000000	0	00000000	0
0x00000098	00000000	0	00000000	0	00000000	0	00000000	0
0x0000009C	00000000	0	00000000	0	00000000	0	00000000	0
0x000000A0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000A4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000A8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000AC	00000000	0	00000000	0	00000000	0	00000000	0
0x000000B0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000B4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000B8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000BC	00000000	0	00000000	0	00000000	0	00000000	0
0x000000C0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000C4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000C8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000CC	00000000	0	00000000	0	00000000	0	00000000	0
0x000000D0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000D4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000D8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000DC	00000000	0	00000000	0	00000000	0	00000000	0
0x000000E0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000E4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000E8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000EC	00000000	0	00000000	0	00000000	0	00000000	0
0x000000F0	00000000	0	00000000	0	00000000	0	00000000	0
0x000000F4	00000000	0	00000000	0	00000000	0	00000000	0
0x000000F8	00000000	0	00000000	0	00000000	0	00000000	0
0x000000FC	00000000	0	00000000	0	00000000	0	00000000	0
0x00000100	00000000	0	00000000	0	00000000	0	00000000	0
0x00000104	00000000	0	00000000	0	00000000	0	00000000	0
0x00000108	00000000	0	00000000	0	00000000	0	00000000	0
0x0000010C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000110	00000000	0	00000000	0	00000000	0	00000000	0
0x00000114	00000000	0	00000000	0	00000000	0	00000000	0
0x00000118	00000000	0	00000000	0	00000000	0	00000000	0
0x0000011C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000120	00000000	0	00000000	0	00000000	0	00000000	0
0x00000124	00000000	0	00000000	0	00000000	0	00000000	0
0x00000128	00000000	0	00000000	0	00000000	0	00000000	0
0x0000012C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000130	00000000	0	00000000	0	00000000	0	00000000	0
0x00000134	00000000	0	00000000	0	00000000	0	00000000	0
0x00000138	00000000	0	00000000	0	00000000	0	00000000	0
0x0000013C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000140	00000000	0	00000000	0	00000000	0	00000000	0
0x00000144	00000000	0	00000000	0	00000000	0	00000000	0
0x00000148	00000000	0	00000000	0	00000000	0	00000000	0
0x0000014C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000150	00000000	0	00000000	0	00000000	0	00000000	0
0x00000154	00000000	0	00000000	0	00000000	0	00000000	0
0x00000158	00000000	0	00000000	0	00000000	0	00000000	0
0x0000015C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000160	00000000	0	00000000	0	00000000	0	00000000	0
0x00000164	00000000	0	00000000	0	00000000	0	00000000	0
0x00000168	00000000	0	00000000	0	00000000	0	00000000	0
0x0000016C	00000000	0	00000000	0	00000000	0	00000000	0
0x00000170	00000000	0	00000000	0	00000000	0	00000000	0
0x00000174	00000000	0						

Chương trình thực hiện :

```

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

The screenshot displays the QtSpim MIPS simulator interface. The main window is divided into several panels:

- Program Arguments:** Shows the assembly code being executed, with the first instruction highlighted: `la s0, test # Nạp địa chỉ của biến test vào s0`.
- Labels:** A list of labels and their corresponding addresses, including `switchcase.asm`, `case_0`, `case_1`, `case_2`, `default`, `continue`, and `test`.
- Registers:** A table showing the current values of MIPS registers. The `s0` register is highlighted, showing its value as `00000000`.
- Data Segment:** A table showing the memory layout, with addresses and values for various data segments.

The bottom status bar indicates the current address is `0x10010000 (data)` and shows options for displaying hexadecimal addresses, hexadecimal values, and ASCII.



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

The screenshot shows a debugger interface with three main panels:

- Program Segment:** Displays assembly code. The instruction at address 0x00400310 is highlighted: `lw s1, 0(s0)`. The source code comment is "# Nạp giá trị của biến test vào s1".
- Data Segment:** Shows memory values. At address 0x10010000, the value is 0.
- Registers:** Shows the state of registers. Register s1 contains the value 0.

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

The screenshot shows a debugger interface with three main panels:

- Program Segment:** Displays assembly code. The instruction at address 0x00400310 is highlighted: `li t0, 0`. The source code comment is "# Nạp giá trị cần kiểm tra".
- Data Segment:** Shows memory values. At address 0x10010000, the value is 0.
- Registers:** Shows the state of registers. Register t0 contains the value 0.

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



Bộ trường hợp khác

```

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

The screenshot shows a MIPS simulator interface. The main window displays the assembly code from the first block. The register window on the right shows the state of the registers. Register \$s3 is highlighted in green, indicating its value. The 'Control and Status' window is also visible on the right side of the simulator.

Registers		Floating Point
Name	Number	Value
\$zero	0	0
\$ra	1	0
\$t0	2	2047470000
\$t1	3	250449724
\$t2	4	0
\$t3	5	0
\$t4	6	0
\$t5	7	0
\$t6	8	225550000
\$t7	9	0
\$t8	10	0
\$t9	11	0
\$s0	12	0
\$s1	13	0
\$s2	14	0
\$s3	15	5
\$s4	16	0
\$s5	17	0
\$s6	18	0
\$s7	19	0
\$s8	20	0
\$s9	21	0
\$s10	22	0
\$s11	23	0
\$s12	24	0
\$s13	25	0
\$s14	26	0
\$s15	27	0
\$s16	28	0
\$s17	29	0
\$s18	30	0
\$s19	31	0
\$PC		8144000



#### Assignment 4

Lần lượt thay thế điều kiện rẽ nhánh trong Home Assignment 1 bằng các điều kiện sau đây: a.  $i < j$  b.  $i \geq j$  c.  $i + j \leq 0$  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  $t3=1$

	name	number	value
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:		

b)  $i \geq j$

```

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  $t3=0$

The screenshot shows the MARS MIPS simulator interface. The 'Text Segment' window displays the assembly code with the 'endif' instruction highlighted. The 'Data Segment' window shows the memory layout. The 'Registers' window on the right shows the values of the registers, with t1 and t3 highlighted.

Register	Value
t0	0
t1	1
t2	-1
t3	0
t4	0
t5	0
t6	0
t7	0
t8	0
t9	0
t10	0
t11	0
t12	0
t13	0
t14	0
t15	0
t16	0
t17	0
t18	0
t19	0
t20	0
t21	0
t22	0
t23	0
t24	0
t25	0
t26	0
t27	0
t28	0
t29	0
t30	0
t31	0

c)  $i + j \leq 0$

```

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  $t3=1$

The screenshot shows a debugger interface with two main panels. The top panel displays the assembly code being executed, and the bottom panel shows the state of the registers.

**Assembly Window:**

Addr	Address	Code	Hex	Source
00000000	00000000	li s1, 2	3f 33 31, 2	li s1, 2
00000001	00000001	li s2, -5	3f 33 32, -5	li s2, -5
00000002	00000002	add s3, s1, s2	8f 33 33, s1, s2	add s3, s1, s2
00000003	00000003	ble s3, zero, then	7f 33 34, zero, then	ble s3, zero, then
00000004	00000004	else:		
00000005	00000005	addi t2, t2, -1	8f 33 35, t2, -1	addi t2, t2, -1
00000006	00000006	add t3, t3, t3	8f 33 36, t3, t3	add t3, t3, t3
00000007	00000007	j endif	6f 33 37, endif	j endif
00000008	00000008	then:		
00000009	00000009	addi t1, t1, 1	8f 33 38, t1, 1	addi t1, t1, 1
0000000a	0000000a	addi t3, zero, 1	8f 33 39, zero, 1	addi t3, zero, 1
0000000b	0000000b	endif:		

**Registers Window:**

Register	Value
t0	0
t1	1
t2	-1
t3	1
t4	0
t5	0
t6	0
t7	0
t8	0
t9	0
t10	0
t11	0
t12	0
t13	0
t14	0
t15	0
t16	0
t17	0
t18	0
t19	0
t20	0
t21	0
t22	0
t23	0
t24	0
t25	0
t26	0
t27	0
t28	0
t29	0
t30	0
t31	0

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  $t2 = -4$ ,  $t3 = 14$

Test Segment

Program Arguments:

Inst	Address	Code	Binary	Source
li	00000000	0000000000000000	0000000000000000	li s1, 2
li	00000001	0000000000000000	0000000000000000	li s2, -5
li	00000002	0000000000000000	0000000000000000	li s3, 3
li	00000003	0000000000000000	0000000000000000	li s4, 4
add	00000004	0000000000000000	0000000000000000	add t2, s1, s2
add	00000005	0000000000000000	0000000000000000	add t3, s3, s4
bgt	00000006	0000000000000000	0000000000000000	bgt t2, t3, then
addi	00000007	0000000000000000	0000000000000000	addi t2, t2, -1
add	00000008	0000000000000000	0000000000000000	add t3, t3, t3
j	00000009	0000000000000000	0000000000000000	j endif
addi	0000000A	0000000000000000	0000000000000000	addi t1, t1, 1
addi	0000000B	0000000000000000	0000000000000000	addi t3, zero, 1
endif	0000000C	0000000000000000	0000000000000000	endif

Data Segment

Address	Value (HE)	Value (HE)	Value (HE)	Value (HE)	Value (HE)	Value (HE)	Value (HE)	Value (HE)
00000000	00	00	00	00	00	00	00	00
00000001	00	00	00	00	00	00	00	00
00000002	00	00	00	00	00	00	00	00
00000003	00	00	00	00	00	00	00	00
00000004	00	00	00	00	00	00	00	00
00000005	00	00	00	00	00	00	00	00
00000006	00	00	00	00	00	00	00	00
00000007	00	00	00	00	00	00	00	00
00000008	00	00	00	00	00	00	00	00
00000009	00	00	00	00	00	00	00	00
0000000A	00	00	00	00	00	00	00	00
0000000B	00	00	00	00	00	00	00	00
0000000C	00	00	00	00	00	00	00	00
0000000D	00	00	00	00	00	00	00	00
0000000E	00	00	00	00	00	00	00	00
0000000F	00	00	00	00	00	00	00	00
00000010	00	00	00	00	00	00	00	00
00000011	00	00	00	00	00	00	00	00
00000012	00	00	00	00	00	00	00	00
00000013	00	00	00	00	00	00	00	00
00000014	00	00	00	00	00	00	00	00
00000015	00	00	00	00	00	00	00	00
00000016	00	00	00	00	00	00	00	00
00000017	00	00	00	00	00	00	00	00
00000018	00	00	00	00	00	00	00	00
00000019	00	00	00	00	00	00	00	00
0000001A	00	00	00	00	00	00	00	00
0000001B	00	00	00	00	00	00	00	00
0000001C	00	00	00	00	00	00	00	00
0000001D	00	00	00	00	00	00	00	00
0000001E	00	00	00	00	00	00	00	00
0000001F	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00
00000021	00	00	00	00	00	00	00	00
00000022	00	00	00	00	00	00	00	00
00000023	00	00	00	00	00	00	00	00
00000024	00	00	00	00	00	00	00	00
00000025	00	00	00	00	00	00	00	00
00000026	00	00	00	00	00	00	00	00
00000027	00	00	00	00	00	00	00	00
00000028	00	00	00	00	00	00	00	00
00000029	00	00	00	00	00	00	00	00
0000002A	00	00	00	00	00	00	00	00
0000002B	00	00	00	00	00	00	00	00
0000002C	00	00	00	00	00	00	00	00
0000002D	00	00	00	00	00	00	00	00
0000002E	00	00	00	00	00	00	00	00
0000002F	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00
00000031	00	00	00	00	00	00	00	00
00000032	00	00	00	00	00	00	00	00
00000033	00	00	00	00	00	00	00	00
00000034	00	00	00	00	00	00	00	00
00000035	00	00	00	00	00	00	00	00
00000036	00	00	00	00	00	00	00	00
00000037	00	00	00	00	00	00	00	00
00000038	00	00	00	00	00	00	00	00
00000039	00	00	00	00	00	00	00	00
0000003A	00	00	00	00	00	00	00	00
0000003B	00	00	00	00	00	00	00	00
0000003C	00	00	00	00	00	00	00	00
0000003D	00	00	00	00	00	00	00	00
0000003E	00	00	00	00	00	00	00	00
0000003F	00	00	00	00	00	00	00	00
00000040	00	00	00	00	00	00	00	00
00000041	00	00	00	00	00	00	00	00
00000042	00	00	00	00	00	00	00	00
00000043	00	00	00	00	00	00	00	00
00000044	00	00	00	00	00	00	00	00
00000045	00	00	00	00	00	00	00	00
00000046	00	00	00	00	00	00	00	00
00000047	00	00	00	00	00	00	00	00
00000048	00	00	00	00	00	00	00	00
00000049	00	00	00	00	00	00	00	00
0000004A	00	00	00	00	00	00	00	00
0000004B	00	00	00	00	00	00	00	00
0000004C	00	00	00	00	00	00	00	00
0000004D	00	00	00	00	00	00	00	00
0000004E	00	00	00	00	00	00	00	00
0000004F	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00
00000051	00	00	00	00	00	00	00	00
00000052	00	00	00	00	00	00	00	00
00000053	00	00	00	00	00	00	00	00
00000054	00	00	00	00	00	00	00	00
00000055	00	00	00	00	00	00	00	00
00000056	00	00	00	00	00	00	00	00
00000057	00	00	00	00	00	00	00	00
00000058	00	00	00	00	00	00	00	00
00000059	00	00	00	00	00	00	00	00
0000005A	00	00	00	00	00	00	00	00
0000005B	00	00	00	00	00	00	00	00
0000005C	00	00	00	00	00	00	00	00
0000005D	00	00	00	00	00	00	00	00
0000005E	00	00	00	00	00	00	00	00
0000005F	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00
00000061	00	00	00	00	00	00	00	00
00000062	00	00	00	00	00	00	00	00
00000063	00	00	00	00	00	00	00	00
00000064	00	00	00	00	00	00	00	00
00000065	00	00	00	00	00	00	00	00
00000066	00	00	00	00	00	00	00	00
00000067	00	00	00	00	00	00	00	00
00000068	00	00	00	00	00	00	00	00
00000069	00	00	00	00	00	00	00	00
0000006A	00	00	00	00	00	00	00	00
0000006B	00	00	00	00	00	00	00	00
0000006C	00	00	00	00	00	00	00	00
0000006D	00	00	00	00	00	00	00	00
0000006E	00	00	00	00	00	00	00	00
0000006F	00	00	00	00	00	00	00	00
00000070	00	00	00	00	00	00	00	00
00000071	00	00	00	00	00	00	00	00
00000072	00	00	00	00	00	00	00	00
00000073	00	00	00	00	00	00	00	00
00000074	00	00	00	00	00	00	00	00
00000075	00	00	00	00	00	00	00	00
00000076	00	00	00	00	00	00	00	00
00000077	00	00	00	00	00	00	00	00
00000078	00	00	00	00	00	00	00	00
00000079	00	00	00	00	00	00	00	00
0000007A	00	00	00	00	00	00	00	00
0000007B	00	00	00	00	00	00	00	00
0000007C	00	00	00	00	00	00	00	00
0000007D	00	00	00	00	00	00	00	00
0000007E	00	00	00	00	00	00	00	00
0000007F	00	00	00	00	00	00	00	00

Labels

Label	Address
endif	00000009
then	00000006
then	00000006
until	0000000C

Registers

Name	Floating Point	Control and Status	Value
\$a0			2
\$a1			0
\$a2			0000000000000000
\$a3			0000000000000000
\$a4			0
\$a5			0
\$a6			0
\$a7			0
\$a8			0
\$a9			0
\$t0			0
\$t1			1
\$t2			0
\$t3			0
\$t4			0
\$t5			0
\$t6			0
\$t7			0
\$t8			0
\$t9			0
\$t10			0
\$t11			0
\$t12			0
\$t13			0
\$t14			0
\$t15			0
\$t16			0
\$t17			0
\$t18			0
\$t19			0
\$t20			0
\$t21			0
\$t22			0
\$t23			0
\$t24			0
\$t25			0
\$t26			0
\$t27			0
\$t28			0
\$t29			0
\$t30			0
\$t31			0
\$t32			0
\$t33			0
\$t34			0
\$t35			0
\$t36			0
\$t37			0
\$t38			0
\$t39			0
\$t40			0
\$t41			0
\$t42			0



## 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.  $\text{sum} < 0$       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  $s5 = 25$

Text Segment					Labels		Name			Number	Value
Offset	Address	Code	Name	Source	Label	Address					
00000000	00000000	addi s3, zero, 5	s3, 5	li s3, 5	loop	00000000				5	5
00000004	00000004	la s2, A			endloop	00000004				0	00000004
00000008	00000008	li s4, 1			A	00000008				1	1
0000000c	0000000c	li s1, 1	# i = 1								
00000010	00000010	li s5, 0	# sum = 0								
00000014	00000014	loop:									
00000018	00000018	bgt s1, s3, endloop	# if i > n then end loop								
0000001c	0000001c	addi t1, s1, -1									
00000020	00000020	add t1, t1, t1	# t1 = 2 * s1								
00000024	00000024	add t1, t1, t1	# t1 = 4 * s1 => t1 = 4*i								
00000028	00000028	add t1, t1, s2	# t1 store the address of A[i]								
0000002c	0000002c	lw t0, 0(t1)	# load value of A[i] in t0								
00000030	00000030	add s5, s5, t0	# sum = sum + A[i]								
00000034	00000034	add s1, s1, s4	# i = i + step								
00000038	00000038	j loop	# go to loop								
0000003c	0000003c	endloop:									

Data Segment										Name		Number	Value
Address	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)	Value (H)				
00000000	00	00	00	00	00	00	00	00	00				
00000004	00	00	00	00	00	00	00	00	00				
00000008	00	00	00	00	00	00	00	00	00				
0000000c	00	00	00	00	00	00	00	00	00				
00000010	00	00	00	00	00	00	00	00	00				
00000014	00	00	00	00	00	00	00	00	00				
00000018	00	00	00	00	00	00	00	00	00				
0000001c	00	00	00	00	00	00	00	00	00				
00000020	00	00	00	00	00	00	00	00	00				
00000024	00	00	00	00	00	00	00	00	00				
00000028	00	00	00	00	00	00	00	00	00				
0000002c	00	00	00	00	00	00	00	00	00				
00000030	00	00	00	00	00	00	00	00	00				
00000034	00	00	00	00	00	00	00	00	00				
00000038	00	00	00	00	00	00	00	00	00				
0000003c	00	00	00	00	00	00	00	00	00				

b)  $\text{sum} < 0$

Ta thêm câu lệnh `blt s5, zero, endloop` để kiểm tra xem  $\text{sum} < 0$  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 )

Test Segment					Labels		Registers		
Step	Address	Code	Data	Source	Label	Address	Value	Number	Value
1	0x00000000	li s3, 5	5	li s3, 5	loop_start	0x00000000	0	0	0
2	0x00000001	la s2, A	A	la s2, A	loop_start	0x00000001	0	0	0
3	0x00000002	li s4, 1	1	li s4, 1	loop_start	0x00000002	0	0	0
4	0x00000003	li s1, 1	1	li s1, 1	loop_start	0x00000003	0	0	0
5	0x00000004	li s5, 0	0	li s5, 0	loop_start	0x00000004	0	0	0
6	0x00000005	bgt s1, s3, endloop		bgt s1, s3, endloop	loop_start	0x00000005	0	0	0
7	0x00000006	addi t1, s1, -1		addi t1, s1, -1	loop_start	0x00000006	0	0	0
8	0x00000007	add t1, t1, t1		add t1, t1, t1	loop_start	0x00000007	0	0	0
9	0x00000008	add t1, t1, t1		add t1, t1, t1	loop_start	0x00000008	0	0	0
10	0x00000009	add t1, t1, s2		add t1, t1, s2	loop_start	0x00000009	0	0	0
11	0x0000000A	lw t0, 0(t1)		lw t0, 0(t1)	loop_start	0x0000000A	0	0	0
12	0x0000000B	add s5, s5, t0		add s5, s5, t0	loop_start	0x0000000B	0	0	0
13	0x0000000C	blt s5, zero, endloop		blt s5, zero, endloop	loop_start	0x0000000C	0	0	0
14	0x0000000D	add s1, s1, s4		add s1, s1, s4	loop_start	0x0000000D	0	0	0
15	0x0000000E	j loop		j loop	loop_start	0x0000000E	0	0	0
16	0x0000000F	endloop:		endloop:	endloop	0x0000000F	0	0	0

Address	Value (HE)	Value (DE)	Value (HE)	Value (DE)	Value (HE)	Value (DE)	Value (HE)	Value (DE)
0x00000000	0	0	0	0	0	0	0	0
0x00000001	0	0	0	0	0	0	0	0
0x00000002	0	0	0	0	0	0	0	0
0x00000003	0	0	0	0	0	0	0	0
0x00000004	0	0	0	0	0	0	0	0
0x00000005	0	0	0	0	0	0	0	0
0x00000006	0	0	0	0	0	0	0	0
0x00000007	0	0	0	0	0	0	0	0
0x00000008	0	0	0	0	0	0	0	0
0x00000009	0	0	0	0	0	0	0	0
0x0000000A	0	0	0	0	0	0	0	0
0x0000000B	0	0	0	0	0	0	0	0
0x0000000C	0	0	0	0	0	0	0	0
0x0000000D	0	0	0	0	0	0	0	0
0x0000000E	0	0	0	0	0	0	0	0
0x0000000F	0	0	0	0	0	0	0	0

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

```
1  # Laboratory 3, Home Assignment 2
2  .data
3  A: .word 1,3,0,7,9
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 beq t0, zero, endloop
19 add s5, s5, t0 # sum = sum + A[i]
20 add s1, s1, s4 # i = i + step
21 j loop # go to loop
22 endloop:
```

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

Register	Value
\$0	0
\$1	1
\$2	5
\$3	238787682
\$4	238787682
\$5	4
\$6	238787682
\$7	238787682
\$8	238787682
\$9	238787682
\$10	238787682
\$11	238787682
\$12	238787682
\$13	238787682
\$14	238787682
\$15	238787682
\$16	238787682
\$17	238787682
\$18	238787682
\$19	238787682
\$20	238787682
\$21	238787682
\$22	238787682
\$23	238787682
\$24	238787682
\$25	238787682
\$26	238787682
\$27	238787682
\$28	238787682
\$29	238787682
\$30	238787682
\$31	238787682

## 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:

```
1  .data
2      A: .word 1, -3, 5, -4, 8, -10
3  .text
4      li s3, 6 # so phan tu cua A
5      la s4, A
6      li s1, 0 # i = 0
7      li s5, 0 # max = 0
8  loop:
9      bge s1, s3, endloop
10     add t1, s1, s1 # i = 2 * s1
11     add t1, t1, t1 # i = 4 * s1 = 4 * i
12     add t1, t1, s4 # luu tru dia chi cua A[i] vao t1
13     lw t0, 0(t1) # luu tru gia tri cua A[i] vao t0
14     #kiem tra gia tri tuyet doi
15     bge t0, zero, checkmax #Neu >=0 thi giu nguyen, di vao check
16     neg t0, t0 # Gan t0 = -t0 neu t0 la so am
17
18  checkmax:
19     bgt t0, s5, then # if A[i] > max -> nhay vao then
20     addi s1, s1, 1 #i++
21     j loop
22
23  then:
24     mv s5, t0 #max = A[i]
25     addi s1, s1, 1 # i++
26     j loop # quay lai loop
27  endloop:
```