BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Lab 3 : Tập lệnh, các lệnh cơ bản, các chỉ thị biên dịch

Họ tên	MSSV
Phạm Minh Hiển	20235705

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.

```
# Laboratory Exercise 3, Assignment 1
.text
start:
        addi s1, zero, 5
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        blt s2, s1, else
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

Sự thay đổi của bộ nhớ các thanh ghi:

- Sau mỗi câu lệnh trong *start* thanh ghi pc có giá trị thay đổi lần lượt tăng thêm 4 byte và trỏ đến câu lệnh tiếp theo được thực hiện, đến câu lệnh **addi** ngay trước **blt** thì có giá trị 0x00400014. Khi thực hiện **blt** thì thanh ghi pc sẽ chỉ đến địa chỉ câu lệnh **addi** trong **else** do lúc này s2 < s1 và có giá trị 0x00400024.

- Thanh ghi s1 và s2 sau lệnh thứ nhất sẽ có giá trị lần lượt là 5 và 3 do phép **addi**.
- Sau các lệnh **addi** *ở start* thì t1, t2, t3 có giá trị lần lượt là 1, 2 ,3.

	t1	**t2**	**t3**
Sau lệnh **blt**	0x00000001	0x00000002	0x00000003
Sau lệnh **addi** (else)	0x00000001	0x00000001	0x00000006
Sau lệnh **add** (else)	0x00000001	0x00000001	0x00000006

Nhận xét:

Gán s1 = 5 và s2 = 3, t1 = 1, t2 = 2, t3 = 3 :

t1 = 1 (0x00000001) không thay đổi.

t2 = 1 (0x00000001) vì thực hiện t2 = t2 - 1.

t3 = 6 (0x00000006) vì thực hiện t3 = 2 * t3.

Khi ta đổi điều kiện thành s1 = 2 < s2 = 3 :

```
.text
start:
        addi s1, zero, 2
        # addi s1, zero, 5
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        blt s2, s1, else
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

Sự thay đổi của bộ nhớ các thanh ghi:

- Khi thực hiện **blt** thì thanh ghi pc sẽ chỉ đến địa chỉ câu lệnh điều **addi** trong **then** do lúc này s2 > s1 và có giá trị 0x00400018.
- Thanh ghi s1 và s2 sau lệnh thứ nhất sẽ có giá trị lần lượt là 2 và 3 do phép **addi**.
- Sau các lệnh **addi** ở *start* thì t1, t2, t3 có giá trị lần lượt là 1, 2 ,3.

	t1	**t2**	**t3**
Sau lệnh **blt**	0x00000001	0x00000002	0x00000003
Sau lệnh **addi** (then)	0x00000002	0x00000002	0x00000003
Sau lệnh **addi** (then)	0x00000002	0x00000002	0x00000001

Nhận xét:

Gán s1 = 2 và s2 = 3, t1 = 1, t2 = 2, t3 = 3 :

t1 = 2 (0x00000002) vì thực hiện t1 = t1 + 1.

t2 = 2 (0x00000002) không thay đổi.

t3 = 1 (0x00000006) vì thực hiện t3 = 0 + 1.

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.

```
.data
        A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
.text
        li s1, 0
        la s2, A
        li s3, 9
        li s4, 1
        li s5, 0
loop:
        bge s1, s3, endloop
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw t0, 0(t1)
        add s5, s5, t0
        add s1, s1, s4
        j loop
endloop:
```

Sự thay đổi của các thanh ghi :

- Sau lệnh **la** thanh ghi s2 có giá trị là 0x10010000 lưu địa chỉ của mảng A.
- Sau các lệnh **li** các thanh ghi s1, s3, s4, s5 có giá trị lần lượt là 0, 9, 1, 0.
- Sau lần chạy đầu tiên của lệnh **bge** do s1 < s3 nên chỉ có thanh ghi pc thay đổi thêm 4 byte thành 0x0040001c.

	t1	**t0**	**s1**	**s5**
Sau vòng lặp 1	0x10010000	0x00000001	0x00000001	0x00000001
Sau vòng lặp 2	0x10010004	0x00000003	0x00000002	0x00000004
Sau vòng lặp 3	0x10010008	0x00000002	0x00000003	0x00000006

Các vòng lặp khác tương tự.

Đến vòng lặp thứ 9 :

- t1 có giá trị 0x10010020
- t0 có giá trị 0x00000006
- s1 có giá trị 0x00000009
- s5 có giá trị 0x0000002d
- pc có giá trị 0x00400018 ngay sau khi thực hiện **j loop**

Sau đó thực hiện lệnh **bge** :

- pc có giá trị 0x00400038 và **endloop.**

Nhận xét :

Ta thấy với mảng đã cho, bước nhảy là 1 và tổng ban đầu khởi tạo là 0, chương trình thực hiện đúng như dự đoán, thực hiện 9 vòng lặp tương ứng với 9 phần tử trong mảng và in ra tổng của 9 phần tử này là **2d**(45).

Thay đổi bộ giá trị :

- A: 1, 3, 2, 5
- s3 = 4
- s4 = 2
- s5 = 0

```
.data
        A: .word 1, 3, 2, 5
        # A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
.text
        li s1, 0
        la s2, A
        li s3 ,4
        # li s3, 9
        li s4, 2
        li s5, 0
loop:
        bge s1, s3, endloop
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw t0, 0(t1)
        add s5, s5, t0
        add s1, s1, s4
        j loop
endloop:
```

Sau khi chạy chương trình, t1 có giá trị 0x10010008, t0 có giá trị 0x00000002, s1 có giá trị 0x00000004, s5 có giá trị 0x00000003. Chương trình chạy đúng.

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.

```
.data
        test: .word 0
.text
        la s0, test
        lw s1,0(s0)
        li s2, 5
        li s3, 3
        li t0, 0
        li t1, 1
        li t2, 2
        beq s1, t0, case_0
        beq s1, t1, case_1
        beq s1, t2, case_2
        j default
case_0:
        addi s2, s2, 1
        j continue
case_1:
        sub s2, s2, t1
        j continue
case_2:
        add s3, s3, s3
        j continue
default:
continue:
```

Sự thanh đổi của các thanh ghi:

- Sau **la** s0 có giá trị 0x10010000, sau **lw** s1 có giá trị lưu trữ trong s0 là 0.

- Sau các lệnh **li,** các thanh ghi s2, s3, t0, t1, t2 có giá trị lần lượt là 5, 3, 0, 1, 2.
- Khi chạy đến câu lệnh điều kiện **beq** thứ nhất, do s1 = 0 = t0 nên ngay lập tức sẽ nhảy đến *case_0* và thanh ghi pc trỏ đến câu lệnh **addi** trong *case_0*.
- Sau câu lệnh **addi** (*case_0*) thì s2 có giá trị 6 và kết thúc chương trình khi thực hiện câu lệnh tiếp theo là **j continue**.
- Thanh ghi pc có giá trị tăng thêm 4 byte sau mỗi câu lệnh cho đến khi gặp lệnh **beq** đầu tiên, sau lệnh này, do có sự rẽ nhánh nên thanh ghi pc chuyển từ 0x00400020 thành 0x00400030.

Chạy với các trường hợp test = 1 và 2 :

```
.data
        test: .word 1 # 0
.text
        la s0, test
        lw s1,0(s0)
        li s2, 5
        li s3, 3
        li t0, 0
        li t1, 1
        li t2, 2
        beq s1, t0, case_0
        beq s1, t1, case_1
        beq s1, t2, case_2
        j default
case_0:
        addi s2, s2, 1
        j continue
case_1:
        sub s2, s2, t1
        j continue
case_2:
        add s3, s3, s3
        j continue
default:
continue:
```

```
.data
        test: .word 2 # 1 # 0
.text
        la s0, test
        lw s1,0(s0)
        li s2, 5
        li s3, 3
        li t0, 0
        li t1, 1
        li t2, 2
        beq s1, t0, case_0
        beq s1, t1, case_1
        beq s1, t2, case_2
        j default
case_0:
        addi s2, s2, 1
        j continue
case_1:
        sub s2, s2, t1
        j continue
case_2:
        add s3, s3, s3
        j continue
default:
continue:
```

- Trường hợp test = 1 : s2 = 4 -> Chương trình chạy chính xác.
- Trường hợp test = 2 : s3 = 6 -> Chương trình chạy chính xác.

Assignment 4:

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

a. i < j :

```
.text
start:
        addi s1, zero, 2
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        bge s1, s2, else # i >= j (a)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

```
.text
start:
        addi s1, zero, 5
        # addi s1, zero, 2
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        bge s1, s2, else # i >= j (a)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

- Sau các câu lệnh **addi** s1, s2, t1, t2, t3 có giá trị tương ứng là 2(5), 3, 1, 2, 3.
- Với s1 = 2 và s2 = 3 (tương ứng i = 2 và j = 3) thì điều kiện không đúng nên sau khi chạy lệnh **bge** chương trình sẽ thực hiện tiếp *then.* Chương trình thực hiện tiếp và cho giá trị t1 là **0x00000002** (do thực hiện t1 + 1), t3 là **0x00000001** (do thực hiện 0 + 1), t2 không đổi. Sau đó kết thúc chương trình sau lệnh **j endif,** thanh ghi pc có giá trị từ **0x00000020** thành **0x0000002c**
- Với s1 = 5 và s2 = 3, điều kiện đúng nên khi chạy xong **bge,** thanh ghi pc sẽ trỏ đến lệnh **addi** trong *else* và có giá trị chuyển từ **0x00400014** sang **0x00400024**. t1 sẽ có giá trị giữ nguyên là 1, t2 có giá trị **0x00000001** và t3 có giá trị **0x00000006**.

b. i >= j

```
.text
start:
        # addi s1, zero, 5
        addi s1, zero, 2
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        blt s1, s2, else # i < j (b)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

```
.text
start:
        addi s1, zero, 5
        # addi s1, zero, 2
        addi s2, zero, 3
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        blt s1, s2, else # i < j (b)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

	t1	t2	t3
s1 = 2, s2 = 3	0x00000001	0x00000001	0x00000006
s1 = 5, s2 = 3	0x00000002	0x00000002	0x00000001

- Sự thanh đổi của các thanh ghi tương tự Assignment 1, sự khác biệt nằm ở điều kiện.

c. i + j <= 0

```
.text
start:
        addi s1, zero, 5
        # addi s1, zero, 2
        addi s2, zero, 3
        add s3, s1, s2
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        # blt s1, s2, else # i < j (b)
        blt zero, s3, else # i + j > 0 (c)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

```
.text
start:
        # addi s1, zero, 5
        addi s1, zero, -5
        # addi s1, zero, 2
        addi s2, zero, 3
        add s3, s1, s2
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        # blt s1, s2, else # i < j (b)
        blt zero, s3, else # i + j > 0 (c)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

	t1	t2	t3
s1 = 5, s2 = 3	0x00000001	0x00000001	0x00000006
s1 = -5, s2 = 3	0x00000002	0x00000002	0x00000001

- Sự thanh đổi của các thanh ghi tương tự Assignment 1, sự khác biệt nằm ở điều kiện.

d. i + j > m + n (với m và n là các giá trị đã được lưu trong các thanh ghi)

```
.text
start:
        addi s1, zero, 5
        # addi s1, zero, -5
        # addi s1, zero, 2
        addi s2, zero, 3
        add s3, s1, s2
        addi s4, zero, 4 # m
        addi s5, zero, 1 # n
        add s6, s4, s5
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        # blt s1, s2, else # i < j (b)
        # blt zero, s3, else # i + j > 0 (c)
        bge s6, s3, else # i + j <= m + n (d)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

```
.text
start:
        # addi s1, zero, 5
        # addi s1, zero, -5
        addi s1, zero, 2
        addi s2, zero, 3
        add s3, s1, s2
        addi s4, zero, 4 # m
        addi s5, zero, 1 # n
        add s6, s4, s5
        addi t1, zero, 1
        addi t2, zero, 2
        addi t3, zero, 3
        # bge s1, s2, else # i >= j (a)
        # blt s1, s2, else # i < j (b)
        # blt zero, s3, else # i + j > 0 (c)
        bge s6, s3, else # i + j <= m + n (d)
then:
        addi t1, t1, 1
        addi t3, zero, 1
        j endif
else:
        addi t2, t2, -1
        add t3, t3, t3
endif:
```

	t1	t2	t3
s1 = 2, s2 = 3	0x00000001	0x00000001	0x00000006
s1 = 5, s2 = 3	0x00000002	0x00000002	0x00000001

- Sự thanh đổi của các thanh ghi tương tự Assignment 1, sự khác biệt nằm ở điều kiện.

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

```
.data
        A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
.text
        li s1, 0
        la s2, A
        li s3 ,9
        li s4, 1
        li s5, 0
loop:
        blt s3, s1, endloop # n < i(a)
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw t0, 0(t1)
        add s5, s5, t0
        add s1, s1, s4
        j loop
endloop:
```

	t1	t0	s1	s5
Sau vòng lặp 1	0x10010000	0x00000001	0x00000001	0x00000001
Sau vòng lặp 2	0x10010004	0x00000003	0x00000002	0x00000004
Sau vòng lặp 3	0x10010008	0x00000002	0x00000003	0x00000006

Các vòng lặp khác tương tự.

Đến vòng lặp thứ 10 :

- t1 có giá trị 0x10010024
- t0 có giá trị 0x00000000
- s1 có giá trị 0x0000000a
- s5 có giá trị 0x0000002d
- pc có giá trị 0x00400038 ngay sau khi thực hiện **j loop**

Sau đó thực hiện lệnh **blt** :

pc có giá trị 0x0040003c và **endloop.**

b. sum < 0

```
.data
        A: .word 1, 3, -5, 5, 4, 7, 8, 9, 6
.text
        li s1, 0
        la s2, A
        li s3 ,9
        li s4, 1
        li s5, 0
loop:
        # blt s3, s1, endloop # n < i(a)
        blt s5, zero, endloop # sum < 0 (b)
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw t0, 0(t1)
        add s5, s5, t0
        add s1, s1, s4
        j loop
endloop:
```

	t1	t0	s1	s5
Sau vòng lặp 1	0x10010000	0x00000001	0x00000001	0x00000001
Sau vòng lặp 2	0x10010004	0x00000003	0x00000002	0x00000004
Sau vòng lặp 3	0x10010008	0xfffffffb	0x00000003	0xffffffff

Sau vòng lặp 3, do s5 (sum) lúc này có giá trị -1 nhỏ hơn 0 nên lệnh **blt** có điều kiện đúng và sẽ endloop.

c. A[i] == 0

```
.data
        A: .word 1, 0, -5, 5, 4, 7, 8, 9, 6
.text
        li s1, 0
        la s2, A
        li s3 ,9
        li s4, 1
        li s5, 0
        li t0, -1
loop:
        # blt s3, s1, endloop # n < i(a)
        # blt s5, zero, endloop # sum < 0 (b)
        beq t0, zero, endloop # A[i] == 0 (c)
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw t0, 0(t1)
        add s5, s5, t0
        add s1, s1, s4
        j loop
endloop:
```

	t1	t0	s1	s5
Sau vòng lặp 1	0x10010000	0x00000001	0x00000001	0x00000001
Sau vòng lặp 2	0x10010004	0x00000000	0x00000002	0x00000001

Sau vòng lặp 2, lúc này t0 (A[1]) đã có giá trị 0 nên khi xét điều kiện, chương trình endloop.

Assignment 6:

Source Code :

```
.data
        A: .word 1, 3, 2, -5, 4, -6
.text
        li s1, 0 # i
        la s2, A # A address
        li s3 ,6 # n
        li s4, 1 # step
        li s5, 0 # max
loop:
        bge s1, s3, endloop # i >= n, endloop
        add t1, s1, s1
        add t1, t1, t1
        add t1, t1, s2
        lw  t0, 0(t1)
        blt t0, zero, absolute # neu A[i] < 0, lay tri tuyet doi
        blt s5, t0, findmax # neu A[i] > max, max = A[i]
        add s1, s1, s4 # step + 1
        j loop
absolute:
        sub t0, zero, t0 # t0 = 0 - t0
        blt s5, t0, findmax
        add s1, s1, s4
        j loop
findmax:
        add s5, zero, t0
```

```
        add s1, s1, s4
        j loop
endloop:
```

- Kết quả chạy với A: 1, 3, 2, -5, 4 là:
  s5 (max) = **0x00000005** -> Chính xác.
- Kết quả chạy với A: 1, 7, 2, -5, 4, -15 là:
  s5 (max) = **0x0000000f** (15) -> Chính xác
- Thử với các trường hợp khác cũng cho kết quả tương tự.

Kết luận :

- Các lệnh **Brand** sẽ yêu cầu điều kiện chính xác để chuyển vị trí, còn các lệnh **Jump** không cần điều kiện để thực hiện.