Cấu trúc Rẽ nhánh, và Cấu trúc LOOP

1. Cấu trúc rẽ nhánh

- Ngôn ngữ bậc cao: **IF ...THEN...ELSE**
- Ngôn ngữ Assembly sẽ dùng các thanh ghi

Có 2 loại cấu trúc rẽ nhánh:

a) Rẽ nhánh không có điều kiện (Unconditional Control Instructions)

Cú pháp:

j <label> *# nhảy (jump) tới đoạn chương trình bắt đầu với <label>*

b <label> *# nhảy (branch) tới đoạn chương trình bắt đầu với <label>*

b) Rẽ nhánh có điều kiện (conditional Control Instructions)

- Ngôn ngữ bậc cao, ta có các biểu thức so sánh: =, >, >=, <, <=, !=
- Ngôn ngữ Assembly ta không có các biểu thức so sánh như trên, mà ta phải dùng các tập lệnh như sau:

Cú pháp:

- *beq <Rsrc>, <Src>, <label>* *# Nếu Rsrc == Src thì nhảy đến <label>.*
  # (beq: Branch equal)
- *bgt <Rsrc>, <Src>, <label>* *# Nếu Rsrc > Src thì nhảy đến <label>.*
  # (bgt: branch greater than)
- *bge <Rsrc>, <Src>, <label>* *# Nếu Rsrc >= Src thì nhảy đến <label>.*
  # (bge: branch greater than or equal)
- *blt <Rsrc>, <Src>, <label>* *# Nếu Rsrc < Src thì nhảy đến <label>.*
  # (blt: branch less than)
- *ble <Rsrc>, <Src>, <label>* *# Nếu Rsrc <= Src thì nhảy đến <label>.*
  # (ble: branch less than or equal)
- *bne <Rsrc>, <Src>, <label>* *# Nếu Rsrc != Src thì nhảy đến <label>.*
  # (bne: branch not equal)

Ví dụ 1: Viết chương trình nhập vào một số nguyên, xuất kết quả kiểm tra số vừa nhập là số chẵn hay số lẻ

```
.data
	Thongbao: .asciiz "Moi ban nhap vao so nguyen:"
	msgSochan: .asciiz "So ban vua nhap la so chan"
	msgSole: .asciiz "So ban vua nhap la so le"
.text
.globl main
main:
	# Print a string
	li $v0,4
	la $a0, Thongbao
	syscall

	# Read a integer
```

```
        li $v0,5
        syscall
        move $t0, $v0    # move gia tri vua nhap vao $t0

        # So sanh chan hay le
        # chia cho 2 lay du, neu so du <1 thi so chan, nguoc lai
thi so le
        li $t1,1
        rem $t2,t0,2                # $t2= $t0%2 (Chia cho 2 lay
du)
        blt $t2,$t1,sochan          # Neu so du $t2 < 1
sole:
        # Print a string so le
        li $v0,4
        la $a0, msgSole
        syscall
        j Exit

sochan:
        # Print a string so chan
        li $v0,4
        la $a0, msgSochan
        syscall
Exit:
        # Thoat
        li $v0,10
        syscall
```

2. Vòng lặp

- Ngôn ngữ bậc cao ta có: FOR (i=0;i<=n;i++); do ...while; While()
  VD: Chương trình tính tổng các số từ 1 đến n

```
Sum = 0;
For (i=1; i<n; i++)
        Sum = sum + i;

Print kết quả của sum;
```

- Ngôn ngữ Assembly sẽ viết như sau:

```
li  $t0, 0        # $t0 = sum = 0
li  $t1, 1        # $t1 = i=1
lw $t2, n         # $t2 = n
LoopSum:
        bgt  $t1,$t2, ExitLoop        # Nếu i>n thì thoát khỏi vòng lặp
        add $t0, $t0, $t1             # sum = sum + i
```

```
        add $t1,$t1,1           # i = i+1
        j LoopSum

ExitLoop:
        # print ket qua sum
```

Bài tập thực hành tự làm:

1/ Viết chương trình nhập vào một số nguyên N, tính tổng các số từ 1 đến N

2/ Viết chương trình nhập vào một số nguyên N, tính tổng bình phương các số từ 1 đến N

3/ Viết chương trình nhập vào một số nguyên N, tính tổng các số chẵn từ 1 đến N

4/ Viết đoạn code nhập vào một số nguyên, nếu đó là số chia hết cho 3 thì thông báo ra màn hình.

5/ Viết đoạn code nhập vào một số nguyên, xuất ra thông báo đó là số dương, số âm hay số 0.

6/ Viết đoạn code cho nhập vào 2 số nguyên a và b, xác định a > b hay b > a hay 2 số bằng nhau.

7/ Viết đoạn cho nhập một số nguyên dương n, và nếu người dùng nhập số âm hoặc số 0 thì yêu cầu nhập lại cho đến khi nào nhận được giá trị nguyên dương.

8/ Nhập vào hai số nguyên dương a và b, tính tổng các số nguyên dương có giá trị nằm trong đoạn [a, b].