Branch and Loop

If else

```
if(rdi < 5) {
    rsi = 0;
} else {
    rsi = 1;
}
```

cmp & rflags

- cmp Rx, Ry|a: so sánh giá trị thanh ghi(Rx) với 1 thanh ghi(Ry) hoặc 1 số(a) (bằng phép Rx - Ry|a)
- Kết quả được biểu diễn qua thanh ghi đặc biệt: rflags
  - Flag có giá trị = 1 hoặc 0
  - ZF (Zero Flag): = 1 nếu kết quả của thao tác trước = 0
  - CF (Carry Flag) = 1 nếu kết quả của tính toán unsigned quá lớn hoặc quá bé để lưu trong register
  - OF (Overflow Flag) = 1 nếu kết quả của tính toán signed quá lớn để chứa trong register
  - SF (Sign Flag) = 1 Nếu kết quả tính toán là số âm
- Ta cần các instruction thực hiện hành vi dựa trên rflags

Jcc (jump if condition is met)

- JE/JZ <where>: Jump if equal/zero (ZF=1)
- JNE/JNZ <where>: Jump if not equal/zero (ZF=0)
- JA/JAE <where>: Jump if above (unsigned) (CF=0) (or equal (ZF=1))
- JB/JBE <where>: Jump if below (unsigned) (CF=1) (or equal (ZF=1))
- JL/JLE <where>: Jump if less (signed) (SF≠OF) (or equal (ZF=1))
- JG/JGE <where>: Jump if greater (signed) (SF=OF) (or equal (ZF=1))

<where>

- Để thuận tiện khi code, người ta sử dụng label

```
    cmp rdi, 5
    jl .L1
    <code if rdi >= 5>
.L1:
    <code if rdi < 5>
```

JMP: unconditional

- Sau khi branch ra, ta cần thiết phải merge lại đường đi -> unconditional jump

```
if(rdi < 5) {
    rsi = 0;
} else {
    rsi = 1;
}
```

```
    cmp rdi, 5
    jl .L1
    mov rsi, 1
    jmp .L2
.L1:
    mov rsi, 0
.L2:
    <same path>
```

Micro-optimization

- Jump instructions cost cpu cycles -> time (load memory from far away,...)
- Optimization by prediction
- C’s unlikely & likely macro

```
#define unlikely(c) __builtin_expect(!!(c), 0)
#define likely(c) __builtin_expect(!!(c), 1)
```

- likely code path: jump not taken, right after jump instruction
- unlikely code path: jump taken to code at far away, jump back merged path
- Use with caution (no effect on modern CPU?)

Loop

```
for(int i = 0; i < 5; i++) {
    dosomething();
}
```

```
    mov rdi, 0
.L1:
    cmp rdi, 5
    jge .L2
    <do somthing>
    add rdi, 1
    jmp .L1
.L2:
    <out of loop>
```

Resources

- https://code-examples.net/en/q/27f0ccb
- https://www.reddit.com/r/cpp/comments/ap12od/performance_benefits_of_likelyunlikely_and_such/
- http://felixcloutier.com/x86/index.html