## **Compound condition with OR**

• **Program control** of the if-statement with an compound condition "A or theRest":

```
if ( A or theRest ) ---->
                              Т
  statement1;
                              v
                                   ---+ TRUE
else
                            eval A |----+
  statement2;
                           ----+
                              1
                              | FALSE
                              v
                            ----+ FALSE |
                         eval theRest |-----|
                            ----+
                              TRUE
                              v
                                             statement1 <----+</pre>
                              Т
                              v
                           statement2
                              Т
                              Т
                              T
                              v
```

- If A is true, then A or "something" is also true, regardless what the "something" evaluates to.
- Therefore, we can branch (jump) to the then-part as soon as we find that A is true.
- Assembler construct that realizes the control flow of the *if*-statement with an compound condition "A or theRest" is as follows:

```
Evaluate "A" (CMP)

TRUE

Branch on TRUE (!!!) outcome of "condition" to here (A:) ---+

| | |

| FALSE |

V |

Evaluate "theRest" (another CMP) |

FALSE |

FALSE |

FALSE |

FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |

| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
| FALSE |
```

```
Branch on FALSE (!!!) outcome of "theRest" to here (B:) ----|----+
           | TRUE
                                                   v
                                                   "statement1" assembler code <-----
A:
    Branch always to there (C:) -----+
       "statement2" assembler code <-----|
B:
           v
                    _____
с:
           +<
           Т
           v
```

• Example:

```
int x, y, a;
if (x <= a || x >= b)
    x = x + 1;
else
    x = x - 1;
```

The **flow chart** of the **above program** is:



Assembler program for this compound if-statement: click here