The compare and branch instructions

- The **control constructs** "if", "if-else", ("switch") "while-loop", "for-loop" and "do-while-loop" is handled in assembler code by (only) 2 kinds of instructions:
    - **Compare instruction**: compares 2 values and set the status flags (N, Z, V, C) according to the compared result
    - **Branch instruction(s)**: makes the CPU "jump" to a certain memory location when a ceratin condition is satisfied. And if that condition is **not** satisfied, the CPU continues execution with the next instruction (the one after the branch instruction).
- **The Compare Instruction in M68000**:

```
CMP.s  <ea>, Dn       Computes the difference Dn - <ea>
                      and set the status flags (N,Z,V,C) according
                      to the outcome of the subtraction

                      The effect is to compare the value in
                      register Dn against the value specified
                      by <ea>

                      The register Dn is NOT changed !
                      Only the flags are updated
```

- Here is a program that illustrate the effect of CMP: click here **DEMO**

  Look carefully at the flags when you run the program.
- Note: **CMP** by itself is not doing much (if any).

  You will only appreciate its power only when the CMP instruction is followed by a **conditional branch/jump** instruction ! (discussed next)
- **Conditional and Unconditional branching in M68000**
    - There are 6 **conditional branch** instructions and they all work the same way. I will discuss one of them in details first, and then the other 5 will be described just briefly.
    - I will use "Branch if LESS THAN" (BLT) for the discussion.

```
BLT  LABEL         If the status flags, at the moment that this
                   instruction is executed, indicates that the
                   CMP operation resulted in a "LESS THAN"
                   condition, the CPU is instruction to "branch"
                   or "jump" to the memory location marked by
                   LABEL

                   If the status flags indicates the otherwise
                   (i.e., "NOT LESS THAN"), then the CPU will
                   continue the execution with the next
                   instruction (i.e., the instruction that
                   follows the BLT instruction)
```

    - Example:

```
      MOVE.L A, D0      D0 = A
```

```
        CMP.L   B, D0       Compare D0 (A) against B
        BLT     L1          Branch to location L1 if A < B
        MOVE.L X, D7
        ....
        ....
    L1: MOVE.L Y, D7
        ....
```

- If A < B, the "`BLT L1`" instruction will make the CPU jump to the memory location where the "`MOVE.L Y, D7`" is at, and continue the program execution from that point onward (and it will not return back where it came from).
- If A >= B, the "`BLT L1`" instruction does nothing at all (condition is false) and the CPU will continue the execution with the next instruction "`MOVE.L X, D7`"
- Notice that the **operand A** is inside the **destination operation** in the **`CMP.L`** instruction.

  The **BLT** will test "**A** < ..." when you use **variable A** as the **destination operation**

- NOTE: be **very careful** with the order of the operands in CMP !

  The **meaning** of the jump is ***changed*** if you used variable **B** as the **destination operand** in the **CMP.L** instruction:

```
        MOVE.L B, D0        D0 = B
        CMP.L   A, D0       Compare D0 (B) against A
        BLT     L1          Branch to location L1 if B < A
        MOVE.L X, D7
        ....
        ....
    L1: MOVE.L Y, D7
        ....
```

- Here is a program to illustrate the effect of BLT: click here **DEMO**

  Change the program to move #4 into D0 and run it again to see the difference.

- **Important Note**:

  the order of the operand in the compare has a **profound effect** of the result, since A < B is quite **different** than B < A (which is A > B !!!)

- Here is the list of all 6 **conditional** branches in M68000:

  - **BEQ**: branch if condition flags indicate last CMP instruction resulted in the **equal** condition (this one is easy: the Z flag is set !)
  - **BNE**: branch if condition flags indicate last CMP instruction resulted in the **not equal** condition (this one is easy too: the Z flag is reset !)
  - **BLT**: branch if condition flags indicate last CMP instruction resulted in the **less than** condition
  - **BLE**: branch if condition flags indicate last CMP instruction resulted in the **less than or equal** condition
  - **BGT**: branch if condition flags indicate last CMP instruction resulted in the **greater than** condition

  - **BGE**: branch if condition flags indicate last CMP instruction resulted in the **greater than or equal** condition

- And then there is one **unconditional** branch instruction in M68000:

  - **BRA**: branch always (a sure branch)

- **Note:**

  > - You know **all there is to know** about **assembler instructions** to construct **"if"**, **"if-else"**, **"while"**, **"for"** and **"do-while"** control statements in any high level programming language.
  >
  >   But doing it **corrrectly** requires that you **understand** what is going on when each of these **control statements** are executed.
  >
  >   **I.e.**: you need to **understand** the ***flow* of control** of the **program**

  We will examine each control statement and give the "blue print" on how the control statement can be translated into assembler code.

- **A note of the *implementation* of the `branch` instruction**

  - **Recall:**

    ```
    bra   LABEL
    ```

    in cause the **CPU** to **"branch"** to the **program instruction** stored at the **label `LABEL`**

  - In **other words**:

    > - The ***next* instruction** that is **fetched** (and executed) by the **CPU** is at the **address `LABEL`**

    This will **happen** when we **update** the **Program Counter** in the **CPU** with the **address value** of **`LABEL`** !!!

  - So **`bra  LABEL`** is **implemented** as follows:

    > - Store the **value** of the **address `LABEL`** into the **Program Counter**

  - BTW, if we can **update** the **Program Counter** with a **move** instruction in **M68000**, then the **`bra  LABEL`** instruction would be **equivalent** to the following **move** instruction:

```
bra  LABEL        <=====>        move.l    #LABLE, ProgramCounter
```