The `while` statement

- **Program control** of the **`while`** statement:

```
while ( condition )     +---------->|
   statement            |           |
                        |    +--------------+ FALSE
                        |    |  condition   |--------+
                        |    +--------------+        |
                        |           |                |
                        |           | TRUE           |
                        |           |                |
                        |           V                |
                        |       statement            |
                        |           |                |
                        +-----------+                |
                                                     |
                                    +<---------------+
                                    |
                                    V
```

- Assembler construct that realizes the control flow of the **`if`** statement is:

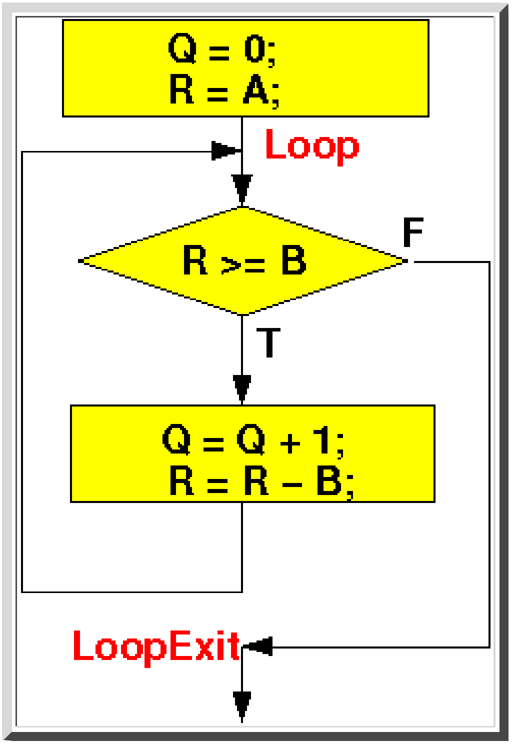
```
LOOP:
   +--> Evaluate "condition" (CMP)
   |              |
   |              V
   |    Branch on the FALSE outcome of "condition" to A ------------+
   |              |                                                  |
   |              | TRUE                                             |
   |              |                                                  |
   |              V                                                  |
   |    Translate "statement" into assembler code                    |
   |              |                                                  |
   |              V                                                  |
   +--------- BRA LOOP                                               |
                                                                     |
          A:      +<-------------------------------------------------+
                  |
                  V
```

- Example 1: division by repeated subtraction

```
int A, B;
int Q, R;

(Computes: Q = A/B
           R = A%B)

Q = 0;                        MOVE.L #0, Q
R = A;                        MOVE.L  A, R

while ( R >= B )       Loop:  MOVE.L R, D0
{                             CMP.L   B, D0
   Q = Q + 1;                 BLT     LoopExit
   R = R - B;
}                             MOVE.L Q, D0
                              ADD.L   #1, D0
                              MOVE.L D0, Q
```

```
                     MOVE.L R,  D0
                     SUB.L  B,  D0
                     MOVE.L D0,  R

                     BRA    Loop

                LoopExit:
```

The **flow chart** of the **program segment**:



- Here is the assembler program, you can assemble it yourself: click here

  The following assembler program does the same thing, just "faster". But since the focus of the course is understanding how the computer works, and not make it run faster, this program will not be discussed. If you're curious, take a look. Basically, don't use the memory if you don't have to: click here

- **Observation**

  - The **while** statement:

    ```
       while ( C )
       {
          s1;
          s2;
          ...
       }
    ```

    give rise to an **assembler program** with the following **structure**:

```
WhileStart:
              instructions to perform a compare
              specified by the condition C

              branch of FALSE of the condition C to label WhileEnd


              instructions to perform s1
              instructions to perform s2
              ....

              bra WhileStart

WhileEnd:
```

- **Note:**

  - The **statements** inside the **while body** can be **"translated"** into **assembler code** ***independently*** from the **while-statement**

  - Use this **property** to **handle**:

    - *Nesting* of a **while** statment inside another **while** statment
    - Nesting of a **if/if-else** statment inside a **while** statment