

Programming (Control) Structures Guide

Jorge Vasconcelos

Programming Control Structures refer to the way computer instruction flow is managed. In principle, instructions are executed one after another, in the same way they were stored in the computer memory ([von Neumann's model](#)). However, computing solutions hardly require strictly sequential programs, instead, some instructions have to be executed conditionally and many others have to be repeated several times.

Historically, such kind of behavior was classified as (see http://en.wikipedia.org/wiki/Structured_program_theorem.)

1. **Sequence**,
2. **Selection**, and
3. **Iteration**.

Computer languages with statements that directly implement those structures are known as **structured languages**.

For example, in C or C++, a sequence of instructions is denoted by the symbols { and }, the selection of actions occurs with the statement *if*, or *if-else*, and iteration is accomplished with the loops *for*, *while*, and *do-while*.

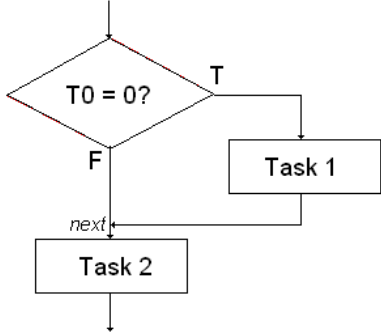
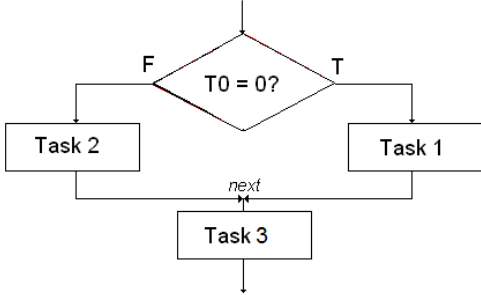
Assembly languages are not structured in the sense that they do not provide specific instructions to control instruction-flow; however, that is easily implemented, and is an essential feature of any language compiler. At this point, the flowcharts corresponding to each structure result quite useful for the compiler designer or assembly programmer..

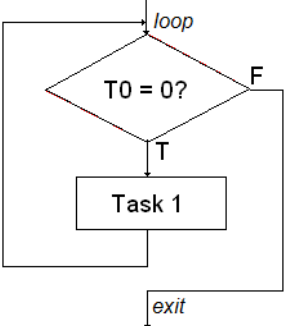
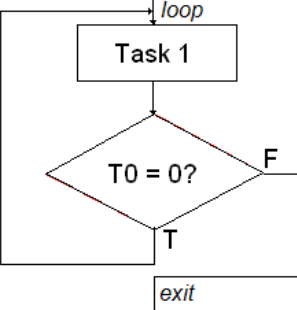
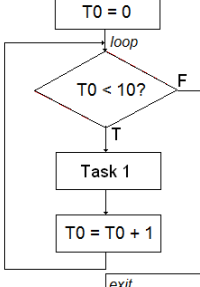
Programming (Control) Structures Guide

Jorge Vasconcelos

(Abridged section)

Sequence structure flowchart	Assembly template	C/C++
<pre>graph TD; Start([Start]) --> Task1[Task 1]; Task1 --> Task2[Task 2]; Task2 --> Task3[Task 3]; Task3 --> End([End]);</pre>	<p>Start: ; label for begining</p> <p> nop ; replace by instruction 1</p> <p> nop ; replace by instruction 2</p> <p> nop ; replace by instruction 3</p> <p>End: ; end of the sequence</p>	<pre>int main () { // Start function1 (); function2 (); function3 (); return 0; } // End</pre>

Selection structure flowchart I	Assembly template	C/C++
 <pre> graph TD Entry(()) --> Decision{T0 = 0?} Decision -- T --> Task1[Task 1] Decision -- F --> Task2[Task 2] Task1 --> Next((next)) Task2 --> Next Next --> Exit(()) </pre>	<pre> bne \$t0, \$0, next ; t0 is not 0, goto next nop ; replace by instruction 1 next: nop ; replace by instruction 2 </pre>	<pre> if (t0==0) function1(); // next function2(); </pre>
Selection structure flowchart II	Assembly template	C/C++
 <pre> graph TD Entry(()) --> Decision{T0 = 0?} Decision -- T --> Task1[Task 1] Decision -- F --> Task2[Task 2] Task1 --> Next((next)) Task2 --> Next Next --> Task3[Task 3] Task3 --> Exit(()) </pre>	<p><i>< Insert your code here ></i></p>	<pre> if (t0==0) function1(); else function2(); function3(); // next </pre>
Selection structure flowchart II	Assembly template	C/C++
<p><i>< Insert your chart here ></i></p>	<p><i>< Insert your code here ></i></p>	<pre> Switch (n) { case 0: function1(); break; case 2: function2(); break; default: function3(); break; } </pre>

Iteration structure flowchart I	Assembly template	C/C++
 <pre> graph TD Start(()) -- loop --> Cond{T0 = 0?} Cond -- T --> Task1[Task 1] Task1 --> Cond Cond -- F --> Exit((exit)) </pre>	<pre> loop: beq \$t0, \$0, next ; t0 is 0, goto exit nop ; replace by instruction 1 j loop exit: nop ; replace by instruction n </pre>	<pre> //loop while (t0 = 0) { function1(); } //exit </pre>
Iteration structure flowchart II	Assembly template	C/C++
 <pre> graph TD Start(()) -- loop --> Task1[Task 1] Task1 --> Cond{T0 = 0?} Cond -- T --> Task1 Cond -- F --> Exit((exit)) </pre>	<p>< Insert your chart here ></p>	<p>< Insert your chart here ></p>
Iteration structure flowchart III	Assembly template	C/C++
 <pre> graph TD Init[T0 = 0] -- loop --> Cond{T0 < 10?} Cond -- T --> Task1[Task 1] Task1 --> Inc[T0 = T0 + 1] Inc --> Cond Cond -- F --> Exit((exit)) </pre>	<p>< Insert your chart here ></p>	<p>< Insert your chart here ></p>