Appendix G: Flowcharts and Pseudocode

Flowcharts

If you have taken any previous programming courses, you are probably familiar with flowcharting. Flowcharts use graphic symbols to represent different types of program operations. These symbols are connected together into a flowchart to show the flow of execution of a program. The more commonly used symbols are as follows:

Start and End points

Start and End points are commonly represented as rounded rectangles or ovals containing the words "Start" or "End". Small circle is another way to show them.

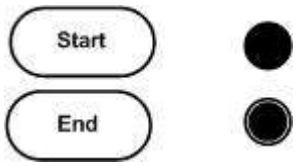

Figure G-1: Start and End Points

Decisions

The conditions are represented in diamonds.

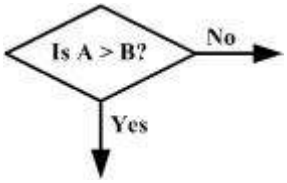

Figure G-2: a Decision that Compares A with B

Process

The processing steps are represented using rectangles.

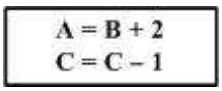

Figure G-3: a Process Sample

Inputs and outputs

Inputs and outputs are represented as parallelogram.

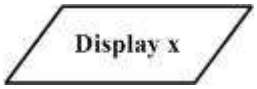

Figure G-4: an Output Sample

Subroutines

Calling subroutines are represented as shown below

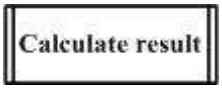

Figure G-5: a Subroutine Call Sample

Pseudocode

Flowcharting has been standard practice in industry for decades. However, some find limitations in using flowcharts, such as the fact that you can't write much in the little boxes, and it is hard to get the “big picture” of what the program does without getting bogged down in the details. An alternative to using flowcharts is pseudocode, which involves writing brief descriptions of the flow of the code. Figures G-6 through G-10 show flowcharts and pseudocode for commonly used control structures.

Structured programming uses three basic types of program control structures: sequence, control, and iteration. Sequence is simply executing instructions one after another. Figure G-6 shows how sequence can be represented in pseudocode and flowcharts.

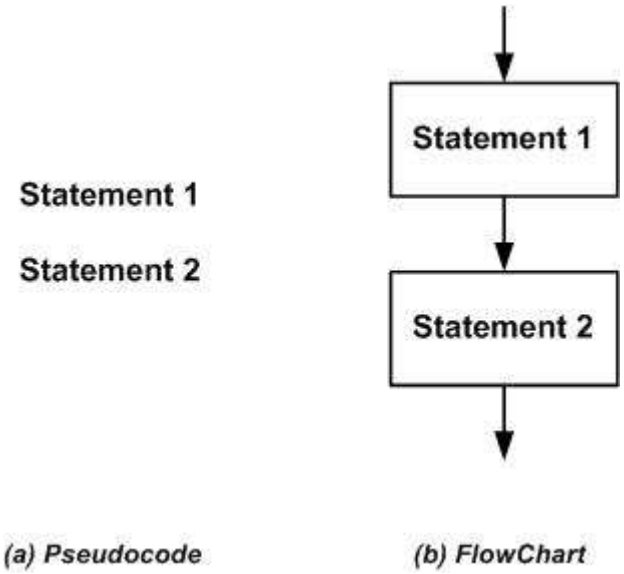

Figure G-6: SEQUENCE Pseudocode versus Flowchart

Note in Figures G-6 through G-11 that “statement” can indicate one statement or a group of statements.

Figure G-7 through G-9 show two control programming structures: IF-THEN-ELSE and IF-THEN in both pseudocode and flowcharts.

```
IF (condition) THEN
      Statement 1
ELSE
      Statement 2
```

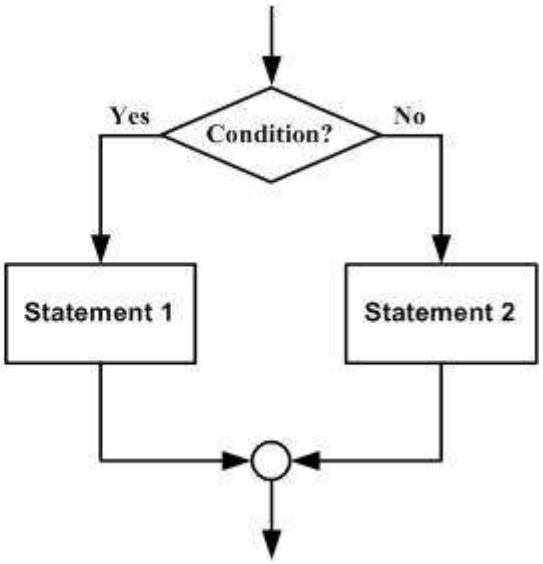

Figure G-7: IF THEN ELSE Pseudocode versus Flowchart

```
IF (A = B) THEN
      C = 2
ELSE
      C = 5
```

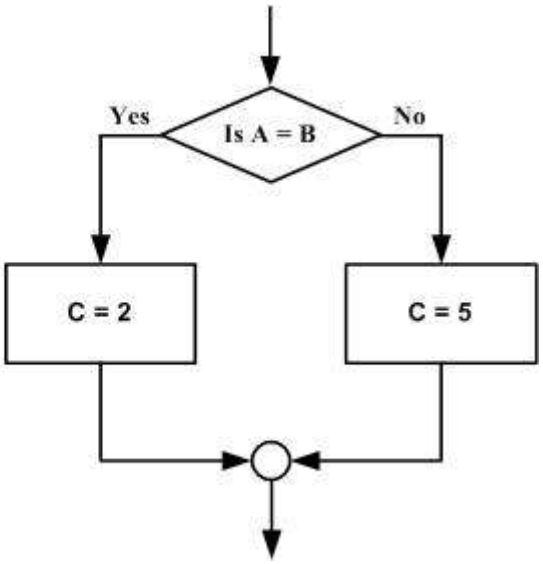

Figure G- 8: an IF THEN ELSE Sample

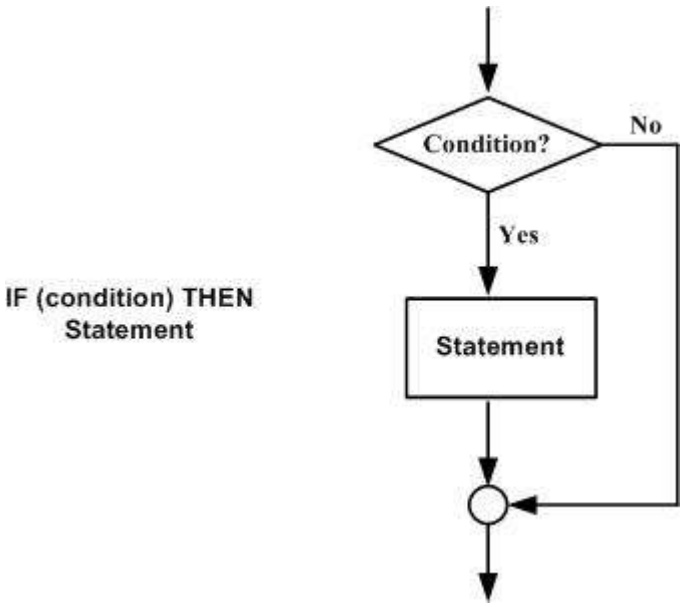

Figure G-9: IF THEN Pseudocode versus Flowchart

Figures G-10 and G-11 show two iteration control structures: REPEAT UNTIL and WHILE DO. Both structures execute a statement or group of statements repeatedly. The difference between them is that the REPEAT UNTIL structure always executes the statement(s) at least once, and checks the condition after each iteration, whereas the WHILE DO may not execute the statement(s) at all because the condition is checked at the beginning of each iteration.

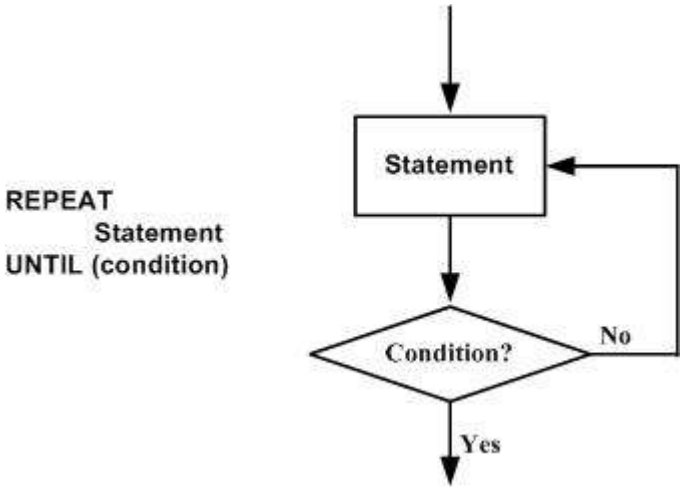

Figure G-10: REPEAT UNTIL Pseudocode versus Flowchart

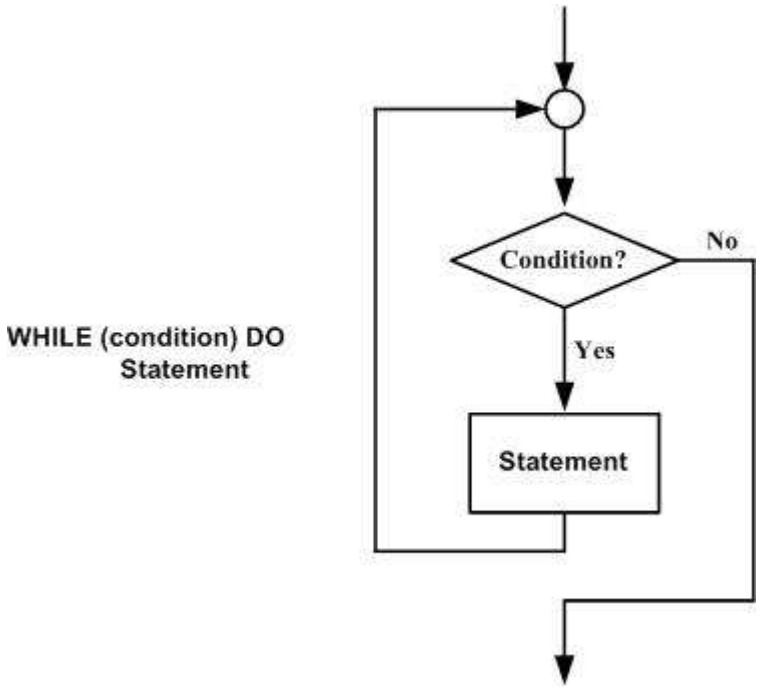

Figure G-11: WHILE DO Pseudocode versus Flowchart

Program G-1 finds the sum of numbers between 1 and 10. Compare the flowchart versus the pseudocode for Program G-1 (shown in Figure G-12). In this example, more program details are given than one usually finds. For example, this shows steps for initializing and changing values. Another programmer may not include these steps in the flowchart or pseudocode. It is important to remember that the purpose of flowcharts or pseudocode is to show the flow of the program and what the program does, not the specific Assembly language instructions that accomplish the program's objectives. Notice also that the pseudocode gives the same information in a much more compact form than does the flowchart. It is important to note that sometimes pseudocode is written in layers, so that the outer level or layer shows the flow of the program and subsequent levels show more details of how the program accomplishes its assigned tasks.

Program G-1

```
int main ()
{
        int sum = 0;
        int value = 1;

        do{
                sum = sum + value;
                value++;
        }while(value <= 10);

        printf ("%d", sum);
}
```

```
value = 1
sum = 0
REPEAT
  sum = sum + value
  value = value + 1
UNTIL (value > 10)
Display sum
```

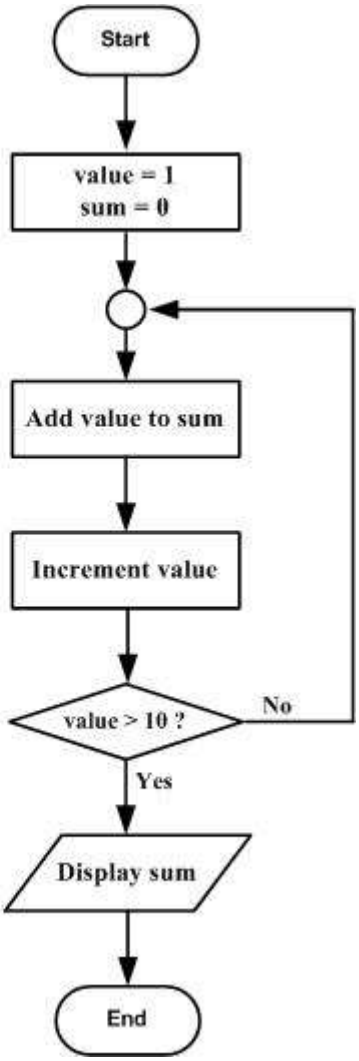

Figure G-12: Pseudocode versus Flowchart for Program G-1