Process Description of an Assembly Conditional Branch Statement

Josiah Schmitz

Research and Argument

ENG 2140

9/16/2024

**Introduction**

In the LC3 instruction set architecture used for the assembly language, a branch statement is used to control the flow of commands in the program. It can be used to create conditional statements, similarly to if statements or while loops in other languages such as Java or Python. The branch statement’s usage consists of three steps: the Executable Instruction, the Branch Statement itself, and the location the statement branches to. This location will either be a loop/exit label or the code immediately following the branch statement. See below in Figure 1.

**LOOP LABEL**

Label statement that branch statement goes to if condition is met

Alternate location in program for a label where a branch could go

**EXIT LABEL**

**EXECUTABLE INSTRUCTION**

Code that is be executed before a branch statement

**BRANCH STATEMENT**

Conditional statement that can move the program to a different location

**FIGURE 1** Usage of a branch statement (by Josiah Schmitz)

**CONDITION IS NOT MET**

**CONDITION IS MET**

**CONDITION IS MET**

Code after branch statement that is executed if condition is not met

**REST OF PROGRAM**

A screenshot of a computer

Description automatically generated

**FIGURE 2** Implementation of branch statement (by Josiah Schmitz)

**Discussion**

**Executable Instruction**

Before the code even encounters the branch statement, it must perform some type of operation on a register in the program. In *Figure 2*, this operation is a simple addition between two registers. The sign (positive, negative, or zero) of the result of the operation will be held until the next line of code and be used to test the branch statement.

**Branch Statement**

If the next line is a branch statement, as seen on line 3 in *Figure 2*, the sign of the result is tested to see if it matches up with the sign referenced in the branch statement. This conditional sign is denoted by the letter after “BR” in the branch statement. Here, the branch statement is checking if the result is negative (n), but it could also check it is positive (p), zero (z), or some combination of the three (pn, nz, pz, or pnz). If the result’s sign matches the condition, then the program jumps to the label following the condition, in this case “LOOP” if the sign is negative. If the condition is not true, the program moves on to the next line of code like normal. In *Figure 2*, it would test the next branch statement, jumping to EXIT if the condition is true.

**Branch Location**

A branch statement can jump the program to any location in the script, depending on where the label is placed. For example, in *Figure 2*, the branch will either jump to “LOOP” if the result’s sign was positive or “EXIT” if it was zero, allowing the program to either continue adding R1 and R2 or move on to another instruction, like performing an AND operation on R1 and R2. This is like how a while loop would work in Java or Python: repeating the same instruction (R0 = R1 + R2) as long as certain conditions are met (R0 is negative).

**Conclusion**

Branch statements and their ability to act as conditionals are a fundamental concept in Assembly programming. They test the sign of an operation’s result and move the program to another instruction depending on that sign. If the condition is not met, the program passes by the branch statement and continues as normal.