

BRANCHING

Consider the task of adding a list of n numbers. The program outlined in Figure 2.9 is a generalization of the program in Figure 2.8. The addresses of the memory locations containing the n numbers are symbolically given as NUM1, NUM2, . . . , NUM n , and a separate Add instruction is used to add each number to the contents of register R0. After all the numbers have been added, the result is placed in memory location SUM. Instead of using a long list of Add instructions, it is possible to place a single

Add instruction in a program loop, as shown in Figure 2.10. The loop is a straight-line sequence of instructions executed as many times as needed. It starts at location LOOP and ends at the instruction Branch>0. During each pass through this loop, the address of

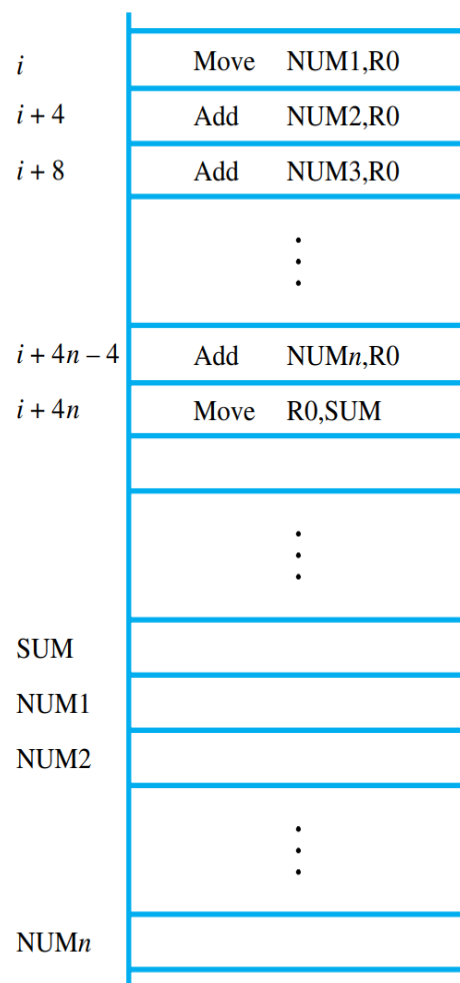


Figure 2.9 A straight-line program for adding n numbers.

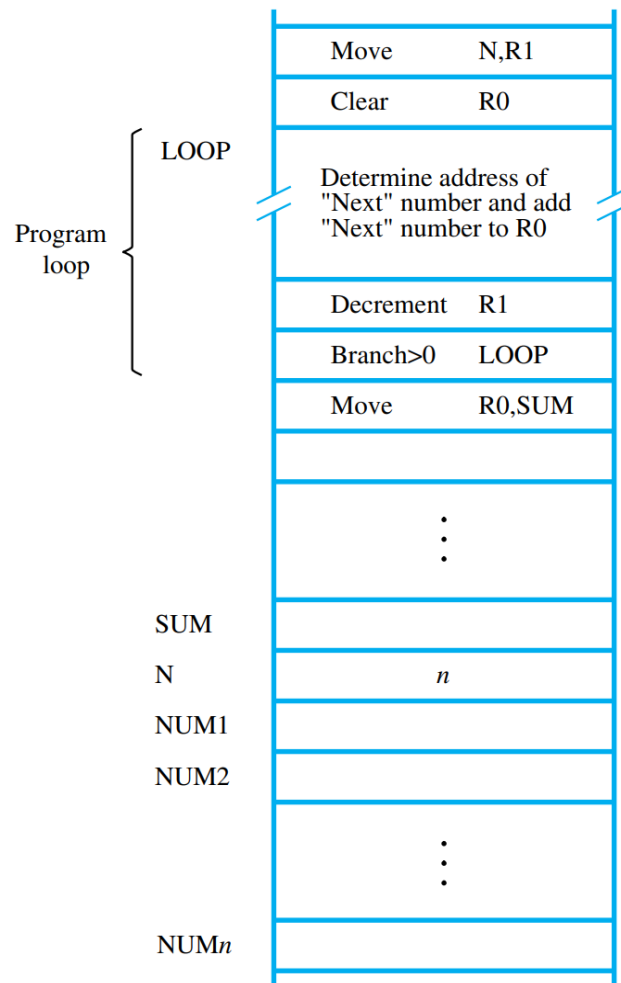


Figure 2.10 Using a loop to add n numbers.

the next list entry is determined, and that entry is fetched and added to R0. The address of an operand can be specified in various ways, as will be described in Section 2.5. For now, we concentrate on how to create and control a program loop. Assume that the number of entries in the list, n , is stored in memory location N, as shown. Register R1 is used as a counter to determine the number of times the loop is executed. Hence, the contents of location N are loaded into register R1 at the beginning of the program. Then, within the body of the loop, the instruction

Decrement R1

reduces the contents of R1 by 1 each time through the loop. (A similar type of operation is performed by an Increment instruction, which adds 1 to its operand.) Execution of the loop is repeated as long as the result of the decrement operation is greater than zero

We now introduce branch instructions. This type of instruction loads a new value into the program counter. As a result, the processor fetches and executes the instruction at this new address, called the branch target, instead of the instruction at the location that follows the branch instruction in sequential address order. A conditional branch instruction causes a branch only if a specified condition is satisfied. If the condition is not satisfied, the PC is incremented in the normal way, and the next instruction in sequential address order is fetched and executed. In the program in Figure 2.10, the instruction

Branch>0 LOOP

(branch if greater than 0) is a conditional branch instruction that causes a branch to location LOOP if the result of the immediately preceding instruction, which is the decremented value in register R1, is greater than zero. This means that the loop is repeated as long as there are entries in the list that are yet to be added to R0. At the end of the nth pass through the loop, the Decrement instruction produces a value of zero, and, hence, branching does not occur. Instead, the Move instruction is fetched and executed. It moves the final result from R0 into memory location SUM. The capability to test conditions and subsequently choose one of a set of alternative ways to continue computation has many more applications than just loop control. Such a capability is found in the instruction sets of all computers and is fundamental to the programming of most nontrivial tasks