BCT is used to implement counted loops - loops in which the number of iterations is known before entrance to the loop body occurs. First the number of iterations is loaded into operand 1, a register which will control the loop. Operand 2 specifies a target address to branch to when the end of the loop body is encountered. Each time the BCT is executed the register denoted by operand 1 is decremented by 1. (The subtraction occurs in 2’s complement arithmetic.) If the result in operand 1 is not zero, the branch is taken to the target address specified in operand 2. If the result is zero, execution continues with the instruction following the BCT. Here is an example.

```
LA R8,3     SET THE NO. OF ITERATIONS TO 3
LOOP EQU *
...  (LOOP BODY GOES HERE)
BCT R8,LOOP DECREMENT LOOP, BRANCH BACK IF NOT ZERO
```

In the example above, 3 is loaded into R8. The loop body is executed and the BCT is executed. This causes R8 to be decremented by 1. Since the result, 2, is not zero, a branch occurs back to “LOOP”. The loop body is executed again, and BCT reduces R8 to 1. Again, a branch is taken to “LOOP”. The loop body is executed a third time and R8 is reduced to 0 by the execution of BCT. Since the result in R8 is equal to zero, the branch is not taken and execution continues with the instruction which follows the BCT.

**Examples**

**Some Unrelated BCTs**

R4 = X’00000004’
R5 = X’00000001’
R6 = X’00000000’

BCT R4, HERE R4 = X’00000003’, BRANCH IS TAKEN TO “HERE”
BCT R5, THERE R5 = X’00000000’, NO BRANCH IS TAKEN
BCT R6, YON R6 = X’FFFFFFFF’, (-1 in 2’s complement), BRANCH IS TAKEN TO “YON”

**Tips**

1. Be sure to initialize the register that controls the loop. In the last example above, the loop would execute the loop body $2^{31}$ times, running through all possible values, before R6 contained a 0.