What is Being Done?

The program multiplies:

$$\texttt{AC := AC} \times \texttt{N}$$

- **AC** initially contains the multiplicand (say Y).
- **N** is stored in memory location `mult`.
- Multiplication is performed via repeated addition: $AC + AC + \cdots$ (N times).

Program Logic:

- Adds `AC` to a product register `N` times.
- A loop:
  - Adds current `AC` to `prod`.
  - Decreases `N` (in `mult`) by 1.
  - Tests if `N` == 0 using a `BZ` (Branch if Zero).
  - If not, loops back.

Key Memory Locations

Label	**Purpose**
`one`	Constant value 1
`mult`	Multiplier N
`ac`	Initial AC value (i.e., Y)
`prod`	Final product / result

Assembly Program Breakdown

Line	Label	Instruction (Assembly)	Explanation
0	`one`	`00...01`	Memory location holding constant 1
1	`mult`	`N`	Multiplier N stored here
2	`ac`	`00...00`	Backup of AC (value Y)
3	`prod`	`00...00`	To store the product (starts at 0)
4		`ST ac`	Store initial AC (Y) into `ac`
5	`Loop`	`LD mult`	Load N into AC
6		`BZ exit`	Exit loop if $N == 0$
7		`LD one`	Load 1
8		`MOV DR, AC`	Move 1 into DR
9		`LD mult`	Reload N
10		`SUB`	$N := N - 1$
11		`ST mult`	Store updated N
12		`LD ac`	Load multiplicand (Y)
13		`MOV DR, AC`	Move Y into DR
14		`LD prod`	Load current product
15		`ADD`	Add Y to product
16		`ST prod`	Store updated product
17		`BRA Loop`	Jump back to loop start
18	`exit`		End of program

Limitations of Accumulator-Based CPUs

1. **Only one accumulator:**
    - Intermediate results must be moved to and from memory repeatedly.
2. **Few CPU registers:**
    - Values like 1, N, Y, and product are fetched repeatedly from memory.
3. **Slower execution:**
    - More memory accesses = slower performance.
    - If values could stay in dedicated CPU registers, the program would:
        - Run faster
        - Be shorter (fewer instructions)

Summary

- This program multiplies two numbers via repeated addition using an accumulator and data register.
- It demonstrates the single-address, register-scarce design of accumulator-based CPUs.
- The architecture leads to inefficient memory use, limiting performance in data-heavy computations.