

The purpose of this specific program, named "Mult.asm," is to multiply the values stored in the RAM locations R0 and R1 and store the result in RAM location R2.

- @2: Initialize the value in RAM[2] (R2) to 0. This will be used to store the result of the multiplication.
- @i: Initialize a variable i in RAM to 0. This will be used as a loop counter.
- (LOOP): This is a label that marks the beginning of a loop.
- @i: Load the value of i into the D register.
- @0: Load the value in RAM[0] (R0) into the D register.
- D=D-M: Subtract the value in RAM[0] from the value of i and store the result in the D register.
- @END: Jump to the END label if the result in the D register is greater than or equal to 0.
- D;JGE: Conditional jump instruction. If D is greater than or equal to 0, jump to the END label.
- If the jump condition is not met, the program continues with the following instructions:
 - @1: Load the value in RAM[1] (R1) into the D register.
 - @2: Load the current value in RAM[2] (R2) into the D register.
 - M=D+M: Add the values in the D and A registers and store the result in RAM[2] (R2). This effectively multiplies the values in RAM[0] (R0) and RAM[1] (R1) and accumulates the result in R2.
 - @i: Load the value of i into the D register.
 - M=M+1: Increment the value of i in RAM.
 - @LOOP: Jump back to the LOOP label to repeat the multiplication process.
- (END): This is the label where the program jumps to when the loop condition is no longer met. The multiplication process is complete.
- @END: Unconditional jump instruction to itself. This acts as an infinite loop to ensure the program doesn't proceed beyond the END label.

This assembly program implements a simple multiplication algorithm using a loop structure. It repeatedly adds the value in RAM[1] to the accumulator in RAM[2] until the loop counter (i) becomes greater than or equal to the value in RAM[0]. The final result of the multiplication is stored in RAM[2].

CPU Emulator (2.5) - D:\Downloads\nand2tetris\nand2tetris\projects\04\mult\Mult.hack

File View Run Help

Slow Fast Animate: Program flow View: Script Format: Decimal

ROM	Asm	RAM
0	@2	0
1	M=0	1
2	@0	2
3	D=M	3
4	@8	4
5	D;JGT	5
6	@19	6
7	0;JMP	7
8	@2	8
9	D=M	9
10	@1	10
11	D=D+M	11
12	@2	12
13	M=D	13
14	@0	14
15	D=M-1	15
16	M=D	16
17	@8	17
18	D;JGT	18
19	@19	19
20	0;JMP	20
21		21
22		22
23		23
24		24
25		25
26		26
27		27
28		28

```

}
set RAM[0] 2, // restores the arguments in case the program used the
set RAM[1] 4,
output;

set PC 0,
set RAM[0] 6, // sets test arguments
set RAM[1] 7,
set RAM[2] -1; // tests that product was initialized to 0
repeat 210 {
ticktock;
}
set RAM[0] 6, // restores the arguments in case the program used the
set RAM[1] 7,
output;
  
```

D 0

ALU

D Input: 0

M/A Input: 19

ALU output: 0

PC 19 A 19

End of script - Comparison ended successfully