1 Explanation of the code

1.1 Module

The module operation is applied as follows. Assuming A mod B.

• 1. Using the POP and PUSH mechanisms showed in the textbook.

• 2. For positive number, we store A into the register, and keep subtracting B from A. When A becomes negative, instead of storing A to register, we extract the value stored inside the register and store it directly to R0. Then we display it.

• 3. For negative number, we keep adding B to A until it become a positive number, then we store it into R0. Then we display it.

1.2 XOR

For XOR operation, we have:

• 1. We extract each bit by started at 1, and each time after extract 1 bit, we left shift it after using it to clear the carries produced by ADD operation to extract next bit.

• 2. Then, After we extract the bit, we applied ADD operation to get XOR value on that bit.

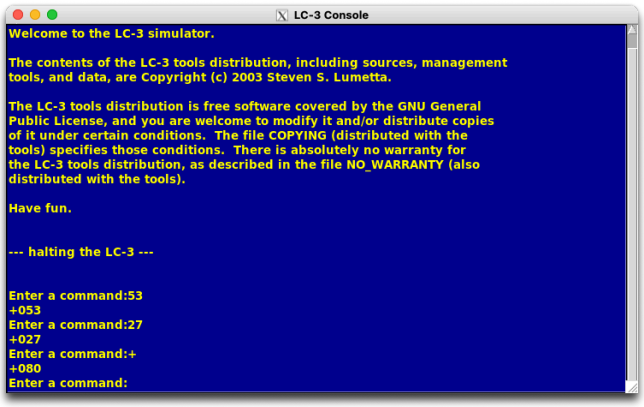
• 3. Don't forget to use the bit extractor (in bullet 1) to clear the carries. Then left shift the bit extractor.

1.3 Other explaination

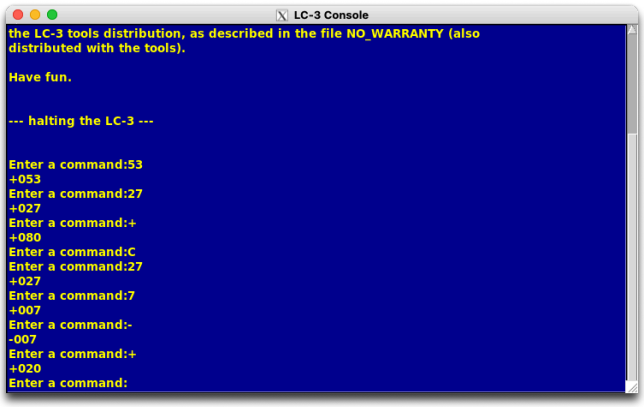
• 1. Remember that the PCoffset in LC3 is only [-256,255]. So when some label is out of range, we can create other symbol and using '.FILL' instruction to get the symbol that is out of range. For LEA some symbols which are out of range, use '.FILL' and LD instead of LEA.

• 2. The code of POP part in second edition is wrong, for R6 was pointing at the "Bottom" of the stackBase and when checking in POP function, the R0 is pointing at the stackBase. So we should modify the 'BRz Underflow' to 'BRp Underflow'. An alternative way is to delete 'ADD R0,R0,#1', which is adopted by 3rd edition.

2 The result of the code

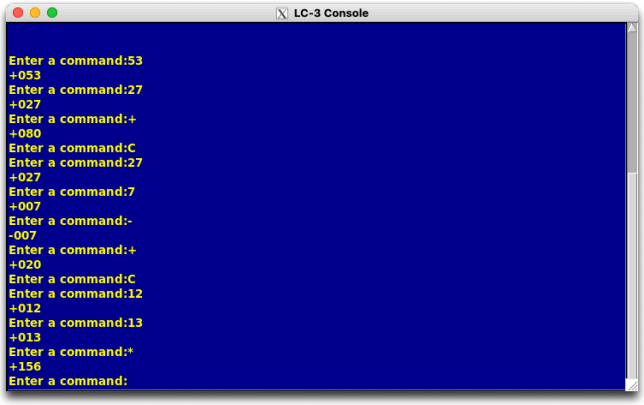


(a) ADD

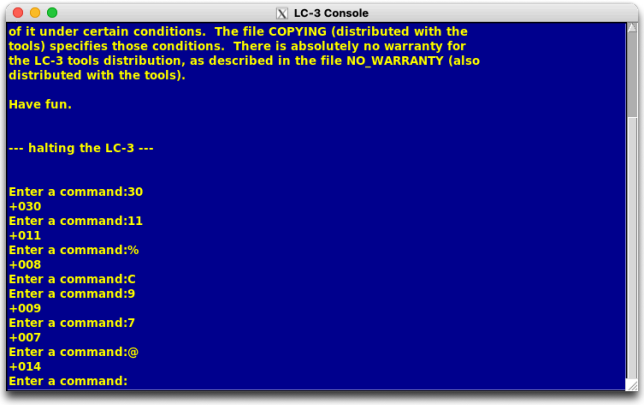


(b) MINUS

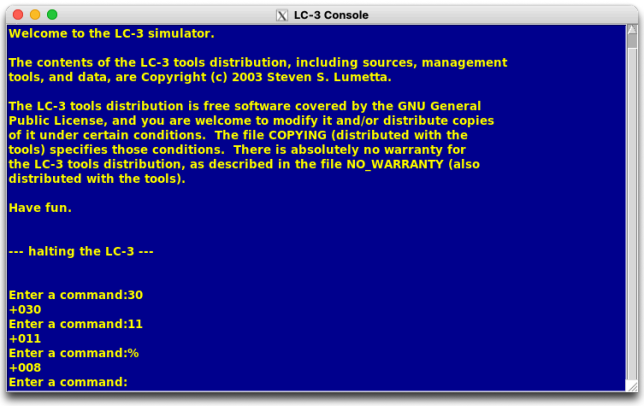
 All experiments please refer to the assembly codes.



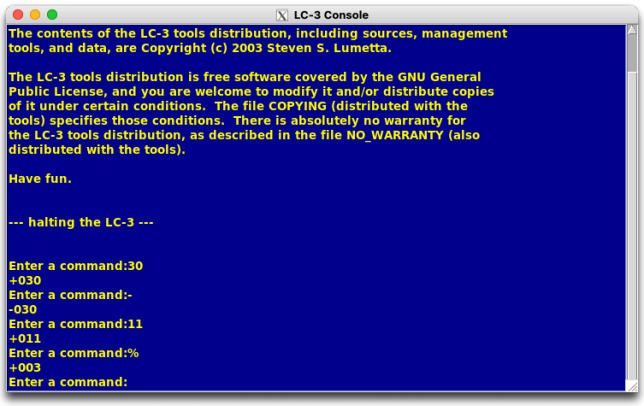
(c) MULTIPLY



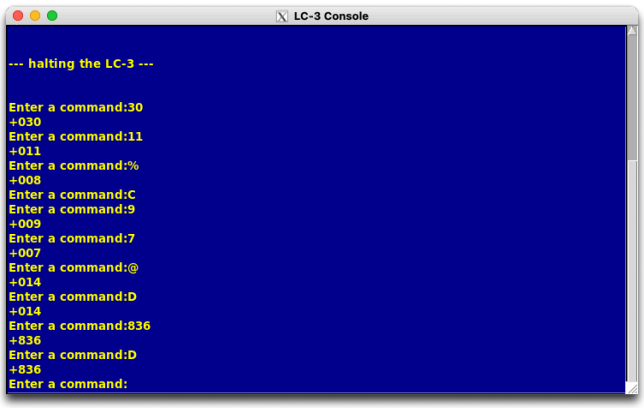
(d) XOR



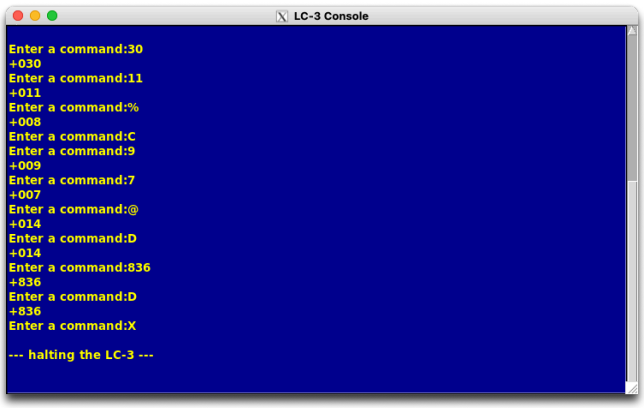
(e) POSITIVE MOD



(f) NEGATIVE MOD



(g) DISPLAY



(h) EXIT

 All experiments please refer to the assembly codes.