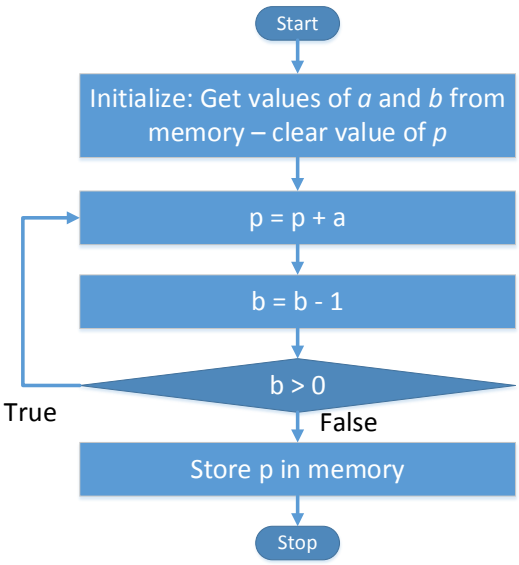
Your Name: _____ netid:_____ Group #:
Name: _____ netid:_____
Name: _____ netid:_____
Name: _____ netid:_____

# ECE 120 Worksheet 12: Programming in LC-3 machine language

Shown on the right is a flow chart for a constrained version of *multiplication by repeated addition* algorithm (it works only when *b*>0). In this discussion, you will develop a similar flowchart, but for a different problem: *division by repeated subtraction*. You will then write a program in LC-3 binary language that implements this algorithm.

Use the following notation/terminology:

- Q = Quotient
- N = Numerator (dividend)
- D = Denominator (divisor)
- R = Remainder.




For simplicity we will assume that both dividend and divisor are strictly positive numbers. Your algorithm should find both Quotient and Remainder.

Your final program written in the LC-3 binary language should use for input and output the memory locations just after you HALT your program:

- Input
  - Dividend (N)
  - Divisor (D)
- Output
  - Quotient (Q)
  - Remainder (R)

# 1. From problem statement to an algorithm

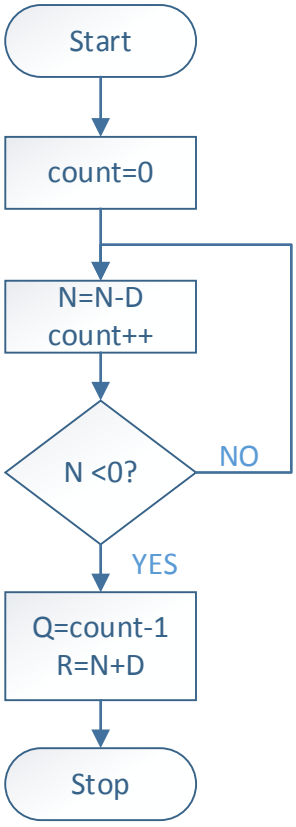
Describe in clear English the sequential processes (algorithm) you will need to follow to perform *division by repeated subtraction* to compute N÷D. Your computation should allow finding both quotient, Q, and remainder, R.

**Continuously subtract D from N until the result becomes negative. Count the number of such subtractions. Q becomes (# of subtractions -1) and R becomes (the negative result + D).**

# 2. Algorithm refinement

Convert your algorithm form Part 1 into a high-level flow chart using sequential, iterative, and conditional constructs. Use English statements and math expressions to describe your steps.

Hint: your flowchart at this stage of refinement should be at the level of details of the flowchart shown on page 1 of this discussion booklet.



## 3. Flowchart refinement

Trace through your flow chart, determine how many values you need to use, and assign registers to each of these values. Will you need to use any registers for multiple values?

Redraw your flow chart using RTL statements with your desired registers.

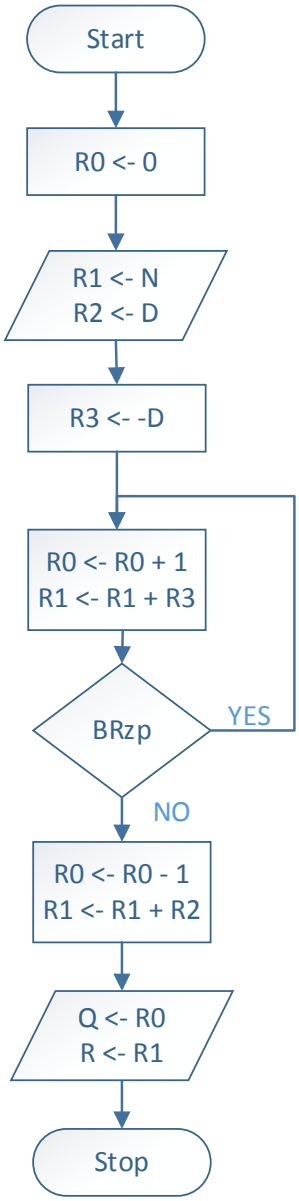
**We will use the following registers:**

**R0 ← count**

**R1 ← N**

**R2 ← D**

**R3 ← -D**



## 4. From flowchart to program

**This question will NOT be graded. Try to do as much as you can, since it is good practice for the final exam.**

Translate your flowchart into an LC-3 binary program. Values of *N*, *D*, *Q*, and *R* should be stored in memory in the locations immediately after you HALT. Your program should start at address x3000.

| Memory Address | Binary instruction | Assembly instruction | Comments |
|---|---|---|---|
| x3000 | **0101 000 000 1 00000** | **AND R0, R0, #0** | **; R0 ← 0 (count)** |
| x3001 | **0010 001 000001011** | **LD R1, xB** | **; R1 ← N** |
| x3002 | **0010 010 000001011** | **LD R2, xB** | **; R2 ← D** |
| x3003 | **1001 011 010 111111** | **NOT R3, R2** | **; R3 ← -D** |
| x3004 | **0001 011 011 1 00001** | **ADD R3, R3, #1** | |
| x3005 | **0001 000 000 1 00001** | **ADD R0, R0, #1** | **; count++** |
| X3006 | **0001 001 001 0 00 011** | **ADD R1, R1, R3** | **; N ← N - D** |
| x3007 | **0000 011 111111101** | **BRzp #-3** | |
| x3008 | **0001 000 000 1 11111** | **ADD R0, R0, #-1** | **; R0 ← R0 - 1** |
| x3009 | **0001 001 001 0 00 010** | **ADD R1, R1, R2** | **; R1 ← R1 + R2** |
| x300A | **0011 000 000000100** | **ST R0, x4** | **; Q ← R0** |
| x300B | **0011 001 000000100** | **ST R1, x4** | **; R ← R1** |
| x300C | **1111 0000 00100101** | **HALT** | **stop** |
| x300D | **0000 0000 0000 0110** | **x0006** | **N = 6 as an example** |
| x300E | **0000 0000 0000 0101** | **x0005** | **D = 5 as an example** |
| x300F | **0000 0000 0000 0000** | | **Q** |
| x3010 | **0000 0000 0000 0000** | | **R** |
| x3011 | | | |
| x3012 | | | |
| x3013 | | | |
| x3014 | | | |
| x3015 | | | |
| x3016 | | | |
| x3017 | | | |
| x3018 | | | |
| x3019 | | | |