**Check for X**

* Ascii for X is 88
* Check to see if user input = 88. Do that the same way you did it to check for Negative “-“
* If it is “X” then go say the goodbye message and HALT
* If not, continue to go check fo LF

**Check for LineFeed, “LF”, the enter key**

* Ascii for LF is 10
* Check to see if user input is 10
* If it is NOT LF then that means user is still inputting digits. Go to PROCESSDIGIT using BRanches
  + First convert it from ASCII to the actual binary number
    - i.e. if user inputted 5. ASCII for 5 is 53. Do the **ASCII – 48** to get the actual binary representation of 5.
  + After processing the real digit, put it in the right digit place
    - i.e. if user presses 1 and then 2 then you want your binary number to be 12
    - do this by **digit = digit \* 10 + the previously processed digit** 
      * digit starts out as 0. So the first digit u process looks like
      * **new digit = 0 \* 10 + first digit**
    - return to GETC and wait for another digit input
* EXAMPLE of 123 being processed:
  + ASCII of 1 is 49.
  + 49-48 = 1
  + Digit = 0 \* 10 + 1
  + Digit = 1
  + Return to GETC and grab the 2
  + ASCII of 2 is 50
  + 50-48 = 2
  + Digit = 1\*10 + 2
  + Digit = 12
  + Return to GETC and grab the 3
  + ASCII of 3 is 51.
  + 51-48 = 3
  + Digit = 12\*10 + 3
  + Digit = 123
  + RETURN TO GETC and do all the checking for LF and NEGs again
* If it is LF go to CHECKFLAG.

**CHECKFLAG**

* is flag = 1? If it is then take the digit and take the complement of it and add 1
* i.e. !123 + 1
* go to MAINMASK
* if flag = 0 do nothing and go to MAINMASK

**MAINMASK**

* **MASK** is given as x8000 .. x4000 .. x2000 each in one address memory away from each other
  + For example. X8000 is in address 1. X4000 is in address 2. X2000 is in address 3. And so on
* Set a count variable to be 15. Set ANOTHER count variable (call it pointer) to 0.
* Load the ADDRESS of MASK that was given to u (LEA)
* Add the ADDRESS + pointer variable
* Use LDR to attain the contents inside the new address
* BinaryDigit = digit AND mask
  + If BinaryDigit == 0 then PRINT 0
  + If BinaryDigit == 1 then PRINT 1
  + Both afterwards: Decrease Count by 1. Increase Pointer by 1 and restart masking loop by returning back to beginning
* **Repeat till count < 0**