1. What opcode will blank memory initialized to 0x00 look like to the processor?
   1. 0000\_0000

(msb) 0000 -> memory op identifier

0 -> store operation  
0 -> store to ACC register  
00 (lsb) -> address

1. Of the 256 possible opcodes we can get from and 8-bit opcode, how many are not being used

in our instruction set, i.e., how many instructions could we add for future expansions of our

Processor?

Math ops: 1x8x4x4 = 128  
Mem ops: 1x2x2x3 = 12   
Branch ops: 1x7 = 52 (round down to 7)  
Plus 2 special op codes  
Sum: 149 opcodes  
8-bits -> 256 diff options…  
Thus 107 additional opcodes could possibly be implemented.

1. What would we need to add to our simulator to be able to include the following instructions: compare ACC with a constant, PUSH to or PULL from the stack, and take the 2's complement of ACC?

An opcode that has room for operand(s) and stuff would be required for all.  
Push/pull would need a register to hold top of stack.  
Two’s complement could be implemented using other standard arithmetic/bit operations.

1. If executeInstruction() were divided into two parts, decode and execute, what additional global resources would be needed for your simulator?

My implementation did attempt to split all functions into more modular methods. But if you were to implement it fully separately, it could be useful to have the (source and destination) operands as global variables.