Questions:  
  
1. What opcode will blank memory initialized to 0x00 look like to the processor?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Function | Register | Method | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Function is 0 which means that the store function is executed. Register is 0 which represents the accumulator ACC, and last is the method with is 00 which means that the operand is used as an address.

STOR ACC, [address]

STOR ACC, [0x0000]

2. 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?

There are 105 unused opcodes in our instruction set. This same amount can be added within an instruction set for future expansion of the processor.

3. 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?

All the operations would need an additional opcode assigned to them.

To compare ACC with a constant an additional register would need to be added for the subtraction and hold the result by setting the flags.

PUSH and PULL would need an additional register to point to the memory where the stack is contained but it would not need anything else.

For 2’s complement, branch and memory operations could be executed as special opcodes. However, for all the patterns for mathematical are used so adding would be complicated.

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

To divide executeInstruction() into decode and execute you would need global resources such as the operation, the source, and the destination to pass the data from one part to the other.