Project 1 Instruction Decoder

By: Ahmed Alkhawar, Naimah-Joy Chapman, Jorge Santos

11/11/21

Questions:

As part of your project, answer the following questions. You may discuss the concepts with

others in the class, but each of you must submit your answers in your own words.

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

* With the assumption that the memory has been cleared it will look this.

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?

* Out of the 256 opcodes there are 121 unused opcodes. Once every operation has been identified it was easier for us to figure out which opcodes were going unused. Any opcodes that were constants, had 3+ bytes and started with “001” or “01” were unused. Since these opcodes were unused they can be used for future expansions.

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?

* PUSH: To implement PUSH we would only have to add a pointer register.
* PULL: To implement PULL we would only have to add a pointer register.
* Compare ACC with a constant: We would have to add another register and set of flags. This would be for the subtraction and to hold the results.
* 2’s complement of ACC: ACC = -ACC

4. If executeInstruction() were divided into two parts, decode and execute, what additional

global resources would be needed for your simulator?

* There would need to be enough resources so that the results can be passed between the divided parts.
* In this case, many additional variables were needed. First of all, this is a bit field that determines the type of operation. Then each specific type of operation requires its own set of flags. For mathematic operation they are type of operation source / destination address, operation width (8/16 bits). For memory operation - direction (store/load), register and memory adressing mode. And for branch operation -type of branching.