Spring 2021
EEE212 Microprocessors
Homework Assignment 1
24.03.2023 08:30 - 03.04.2023 13:00

In this homework assignment, you will practice how to utilize arithmetic/logic instructions, look-up tables, and the stack of 8051. This lab assignment has two interconnected parts. Please read the notes and the assignment requirements carefully since they are pretty important in terms of evaluation.

Important Notes:

- Your submission will be checked using **a Proteus simulation**. Thus, make sure that it works on a Proteus simulation.
- This is an individual assignment. You can cooperate but you have to submit your **OWN** code. Any kind of plagiarism will not be tolerated. Codes will be compared manually by assistants and by Turnitin software after the lab.
- The deadline is strict. Submit your code before the deadline.

Part 1: Divisibility (30 pts)

In this section of the homework assignment, you will write a subroutine that primarily tests if an integer (the divisor) divides another integer (the dividend), where both integers are between 0 and 255, including 0 and 255. Successful implementation of this subroutine is required for Part 2.

Details:

- Assume that `B` contains the divisor, and the accumulator (`ACC`) contains the dividend.
- Set the carry flag (`CY`) if the divisor fully divides the dividend, and have `CY` cleared if otherwise.
- If the divisor is 0, have the overflow flag (`OV`) set.
- Preserve the initial value of the divisor at `B`. (This will be useful for Part 2.)
- You do not need to preserve the initial value of the dividend at `ACC`.

Hint:

- Note that the OV flag is automatically set when an attempt is made to divide by zero (check the DIV AB instruction).

Part 2: Euler's Totient Function (70 pts)

In this section of the assignment, you will have one 8051 MCU compute the number of positive integers smaller than N that are relatively prime (i.e. have a greatest common divisor of 1) with N where N is an unsigned positive integer between 1 and 255, including 1 and 255. A formula to calculate this value, represented by "$\phi(N)$", is given in Equation 1:

$$\phi(N) = N\left(\frac{p_1 - 1}{p_1}\right)\left(\frac{p_2 - 1}{p_2}\right)\left(\frac{p_3 - 1}{p_3}\right)...\left(\frac{p_n - 1}{p_n}\right) \quad (1)$$

where n is the number of unique primes that divide N, and p_i ($\forall i \in \{1, 2, ..., n\}$) are the unique prime numbers that divide N.

For instance, the unique prime factors of 250 are 2 and 5. Thus, $\phi(250) = 250\left(\frac{1}{2}\right)\left(\frac{4}{5}\right) = 100$.

Have the string '`PHI(`' be displayed on the first line on an LCD before taking your input. Take a decimal number with at most three digits as your keypad input (N), and print it on the first line, following the aforementioned string. Then, when the A button on the keypad is pressed, close the bracket, print a '`=`' character, and display the result of $\phi(N)$, which will be computed using the formula given in Equation 1.

Details:

- Your code does **not** need to run indefinitely without reset. In other words, while the LCD is displaying the result of a previous query, the MCU does not need to be able to accept new input from the keypad.
- While taking input, have each digit be displayed immediately when the corresponding digit is pressed on the keypad. The closing parenthesis of '`PHI(___)`' should be displayed only when the A button is pressed.
- The primes smaller than 256 are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, and 251. You may store these values in a look-up table.
- You are not required to be able to display any result for inputs that are not integers between 1 and 255.
- For those who are curious, $\phi(N)$ is formally called *Euler's totient function.*

Hints:

- You will need to convert the keypad input to a hexadecimal value that can be worked on.
- Treat $N = 1$ as a special case even though the mathematical formula given in Equation 1 technically works for it ($\phi(1) = 1$).

- Do not forget that $N \neq 0$. You do not need to devise a special case for it.
- Try to do all necessary divisions *before* any necessary multiplication. If this approach is followed, one will avoid the need to perform 16-bit arithmetic operations.
- Try to completely avoid using the `SUBB` instruction since it undesirably interacts with `CY`. Use `DEC` instead to subtract a prime by 1.
- `XCH A, B` exchanges the bytes stored at `ACC` and `B` with each other.
- Try using different conditional jumps for different situations.
- After calculating $\phi(N)$, remember that you still need to display it on the LCD screen, which is not a trivial task since $\phi(N)$ might not be a single-digit number.