EE 212 FALL 23-24 Lab1

Arda Can Aras & Uras Kargı

September 2023

Please read the notes and the assignment requirements carefully since they are essential for evaluation.

Assignment Requirements

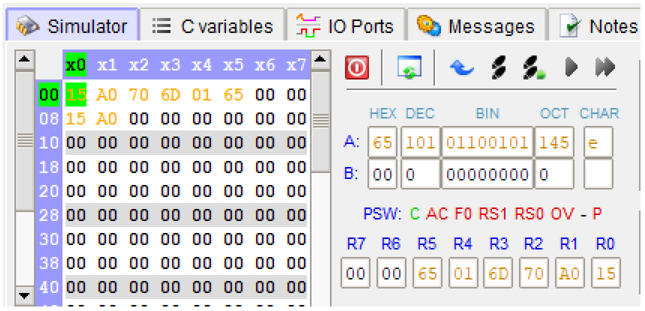
- Your submission will be checked using the MCU 8051 IDE simulator for part 1 and Proteus simulation software for part 2 during lab hours. Thus, ensure your code works before the lab session.
- Lab assignment has two parts. Therefore, you should upload two different files, one for each part. Please upload your files in '.txt' or '.pdf' format.
- The deadline is strict. Submit your code before the deadline. **You cannot change your uploaded codes during lab hours. You will show your demos based on your uploaded codes**.
- This is an individual assignment. You can cooperate but must submit your **OWN** code. plagiarism will **NOT** be tolerated. After the lab, the codes will be compared manually by assistants and by TURNITIN software.
  .

1 Part 1 (40 Pts)

For this part, you do not need to show your result on the LCD of the Proteus or take inputs from the keypad of the Proteus. You will show your results on the MCU 8051 IDE simulator ONLY. (See Figure 1)

In this part of the assignment, you write a program that executes the following: **16-bit numbers will be divided by 8-bit numbers. The dividends will be 16-bit unsigned numbers, and the divisors will be 8-bit unsigned numbers.**. Please note that the result of this operation is indeed 16-bit.

You take inputs (dividend, divisor) at the start of your program. Store the dividend in registers R1, R0 (High part of dividend in R1 and low part of dividend in R0) and divisor in register R2. To summarize, please include the following lines at the start of your program (During your demos, your TA's change



	x0	x1	x2	x3	x4	x5	x6	x7
00	15	A0	70	6D	01	65	00	00
08	15	A0	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00
18	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00
28	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00
38	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00

	HEX	DEC	BIN	OCT	CHAR
A:	65	101	01100101	145	e
B:	00	0	00000000	0	

PSW: C AC F0 RS1 RS0 OV - P

R7	R6	R5	R4	R3	R2	R1	R0
00	00	65	01	6D	70	A0	15

Figure 1: MCU 8051 IDE Simulator Interface

the dividend or divisor via these code lines and check whether your code works):

```
MOV R0, #...; Specify the low part of the dividend
MOV R1, #...; Specify the high part of the dividend
MOV R2, #...; Specify the divisor
```

Store the quotient in registers R4, R3 (High part of quotient in R4 and low part of quotient in R3) and the remainder in register R5. (Since quotient and dividend are 16-bit in general, you need two 8-bit registers of 8051). You do not need to convert hexadecimal results to decimal.

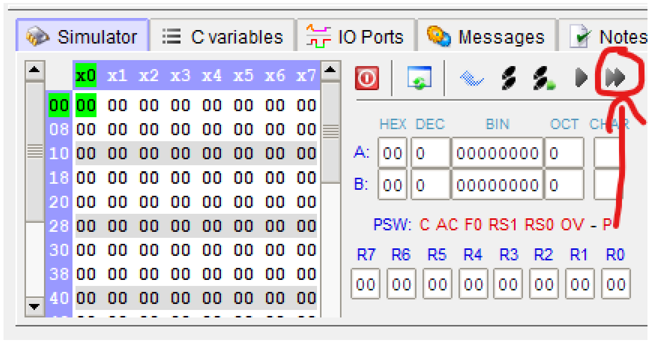
Show your results (i.e., quotient in registers R4, R3 and reminder in register R5) after the simulation ends (see Figure 1).

1.1 Example

- Inputs: R1:R0 = A0:15 (40981 in decimal), R2=70 (112 in decimal)
- Quotient in R4:R3=01:6D, Reminder R5 = 65
- 40981 = 365 * 112 + 101, preserve 365 (quotient) in R4:R3 and 101 (reminder) in R5
- See Figure 1 for the result of this example. You can observe register values after the simulation ends. You will show your results similarly.
- See Figure 2; you can start the simulation by clicking on the sign with the red arrow. It takes some time to complete the process. When it finishes, you can see results on the registers.

1.2 Grading of Part 1

- Show the quotient properly (30 pts)
- Show the remainder properly (10 pts)

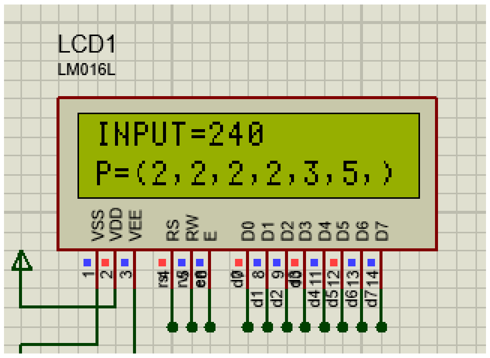


```
    x0 x1 x2 x3 x4 x5 x6 x7
00  00 00 00 00 00 00 00 00
08  00 00 00 00 00 00 00 00
10  00 00 00 00 00 00 00 00
18  00 00 00 00 00 00 00 00
20  00 00 00 00 00 00 00 00
28  00 00 00 00 00 00 00 00
30  00 00 00 00 00 00 00 00
38  00 00 00 00 00 00 00 00
40  00 00 00 00 00 00 00 00

    HEX DEC  BIN       OCT CHAR
A:  00  0    00000000  0
B:  00  0    00000000  0

PSW: C AC F0 RS1 RS0 OV - P
R7 R6 R5 R4 R3 R2 R1 R0
00 00 00 00 00 00 00 00
```

Figure 2: How to start Simulation in MCU 8051 IDE Simulator

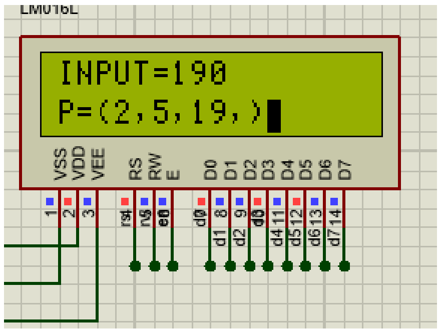
2 Part 2 (60 pts)

In this part of the assignment, you must write an unsigned positive integer in the range [1,255] by the Product of Prime Factors. You can use the part A code for this part. For this part, you need to show your result on the LCD of the Proteus and take inputs from the keypad of the Proteus. Display the Product of Prime Factors in brackets separated by a comma. For instance, (2,5,5,5) needs to be displayed for 250. Check Figures 3 and 4 to understand better. Have the string 'INPUT=' be displayed on the first line on an LCD before taking your input. Take a decimal number with at most three digits in the desired range as input, and print it on the first line following the aforementioned string. Then, when the A button on the keypad is pressed, clear this line and print the results Product of Prime Factors.



```
INPUT=240
P=(2,2,2,2,3,5,)
```

Figure 3: Product of Primes Example



```
INPUT=190
P=(2,5,19,)
```

Figure 4: Product of Primes Example

2.1 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. You can reset the Proteus simulation if you want to enter new input.
- While taking input, have each digit be displayed immediately when the corresponding digit is pressed on the keypad.
- 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. There are 54 primes.
- You are not required to be able to display any result for inputs that are not integers between 1 and 255.

2.2 Grading of Part 2

- Product of Prime Factors (50 pts)
- Satisfying Display Requirements (10 pts)