

Experiment 3: Advanced Assembly Coding

29.10.2023

*Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr*

"Yes. Well, Dr. Shuman tells me that in theory there is nothing the computer can do that the human mind cannot do. The computer merely takes a finite amount of data and performs a finite amount of operations on them. The human mind can duplicate the process."

The president considered that. He said, "If Shuman says this, I am inclined to believe him - in theory. But, in practice, how can anyone know how a computer works?"

Brant laughed genially. "Well, Mr. President, I asked the same question. It seems that at one time computers were designed directly by human beings. Those were simple computers, of course, this being before the time of the rational use of computers to design more advanced computers had been established."

...

Nine times seven, thought Shuman with deep satisfaction, is sixty-three, and I don't need a computer to tell me so. The computer is in my own head.

And it was amazing the feeling of power that gave him.

Isaac Asimov, The Feeling Of Power

1 Introduction and Preliminary

In this experiment we will obtain more experience on the MSP430 board and assembly coding.

2 Part 1 (30 pts): Modulus Operation

Write the assembly code which calculates the modulus operation. Your code should calculate $\text{mod}(A,B)$ and save the result in a register. An example skeleton code is given below.

```
1      mov.w      #122d,   R8
      mov.w      #10d,    R9
3
modulus ...
5
;At the end we have 2 in R8.
```

3 Part 2 (40 pts): Numbers Divisible by Given Numbers

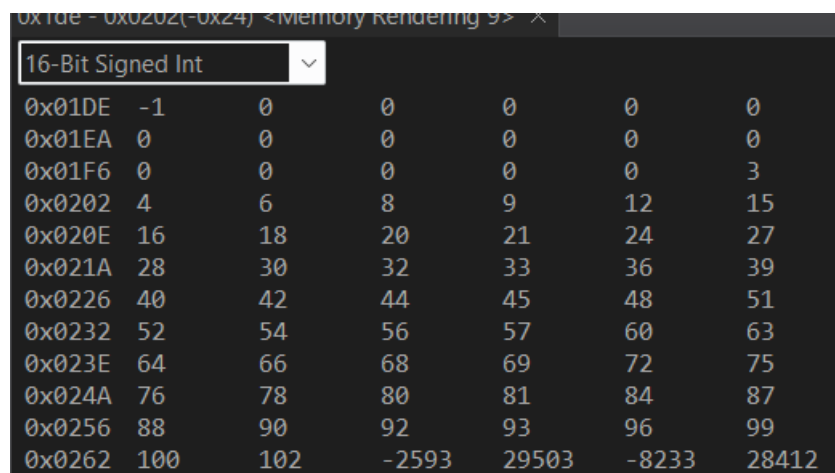
Using the modulus operation, write the assembly code to create an array containing the 50 positive numbers which can be divided by 3 or 4. Save them to a memory address. To allocate space in memory, you can use ".space" directive at the bottom part of the source file.

```
1      .data
arr      .space 100
```

Here, 100 bytes of memory are allocated to keep the divisible numbers. To reach the address of the array you can use "mov.w #arr, R10". The following two lines are used to write 8 and 15 to the array.

```
2      mov.w #arr, R10
      mov.w #8d, 0(R10)
4      add #2d, R10
      mov.w #15d, 0(R10)
```

You can change the code to create a loop to evaluate all numbers starting from 1. Check the memory browser using the address of the array. The result should be as given below.



0x01DE	-1	0	0	0	0	0
0x01EA	0	0	0	0	0	0
0x01F6	0	0	0	0	0	3
0x0202	4	6	8	9	12	15
0x020E	16	18	20	21	24	27
0x021A	28	30	32	33	36	39
0x0226	40	42	44	45	48	51
0x0232	52	54	56	57	60	63
0x023E	64	66	68	69	72	75
0x024A	76	78	80	81	84	87
0x0256	88	90	92	93	96	99
0x0262	100	102	-2593	29503	-8233	28412

4 Part 3 (30 pts): Reverse the list

After filling arr, fill another list named arr2 with the reverse order of arr. The result should be as given below.