CT Lab: Arithmetic Operations

1 Introduction

In this lab you will write your own assembly programs to perform additions and subtractions on the CT Board.

2 Learning Objectives

- You can apply addition and subtraction operations in assembly programs.
- You understand the implications of these operations on the processor flags.
- You can implement additions, which exceed the word size of the processor.

3 Task 1 – Sum and Difference

3.1 Introduction

You shall write a program that adds and subtracts two 8 bit values. To be able to observe the carry and overflow flags, the two 8 bit values will be shifted to the left by 24 bits. The right part will be filled with zeroes.

3.2 Implementation

Open the given project frame *sum_diff* and expand it, so that one 8 bit value is read from the DIP-switches S15 to S8 (operand A) and another from S7 to S0 (operand B). Both operands shall be expanded to 32 bit, as mentioned in the introduction. Display the most significant byte of the sum on LED7 to LED0 and the most significant byte of the difference on LED23 to LED16.

> - The instruction **`LSLS`** (i.e. **`LSLS R1, R1, #24`**) shifts the content of register R1 to the left by 24 bits. The right part gets filled with zeroes. You will learn about the shift instructions in a future lesson.

Additionally you shall display the flags set by these two operations on the LEDs. The flags from the addition shall be displayed on LED15 to LED12 and the ones from the subtraction on LED31 to LED28 (See Figure 1).

> - To read the processor flags, you can use the **`MRS`** instruction.
> i.e.: **`MRS R1, APSR`** copies the content of the application program status register to register **`R1`**.
> - The four flag bits are positioned from bit 31 to bit 28. To display these on the LEDs, you need to shift those bits to the right by 24 bits. Use the **`LSRS`** instruction.
> i.e.: **`LSRS R1, R1, #24`** shifts the content of R1 to the right by 24 bits.

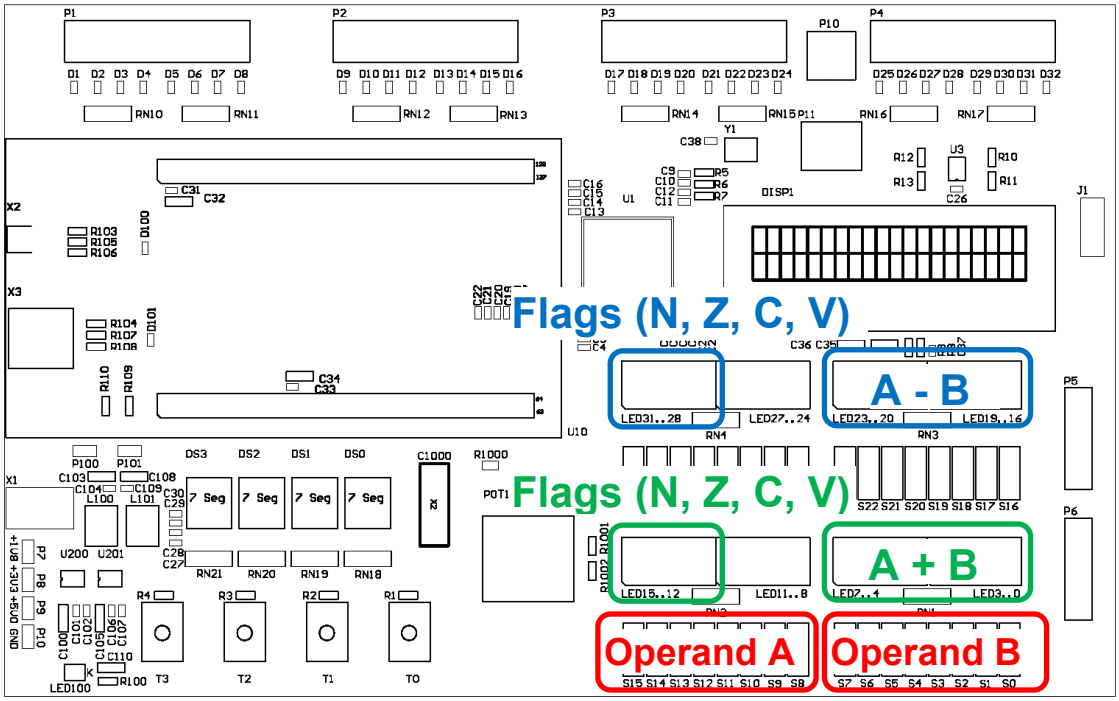


Figure 1: In and output of values for the sum and difference

3.3 Verification

Calculate the expected results based on the following table, i.e. sum, difference and flags.

		Addition: A + B						Subtraction: A – B				
A	B	Result	N	Z	C	V		Result	N	Z	C	V
0x82	0x12	10010100	1	0	0	0		01110000	0	0	1	1
0x34	0x72	10100110	1	0	1	0		11000010	1	0	0	0
0xC2	0x87	01001001	0	0	1	1		00111011	0	0	1	0
0xA3	0x62	00000101	0	0	1	0		01000001	0	0	1	1
0x67	0x99	00000000	0	1	1	0		11001110	1	0	0	1

- Note for subtraction:
  C = '0' means 'borrow', while C = '1' means, that 'borrow' is not necessary.

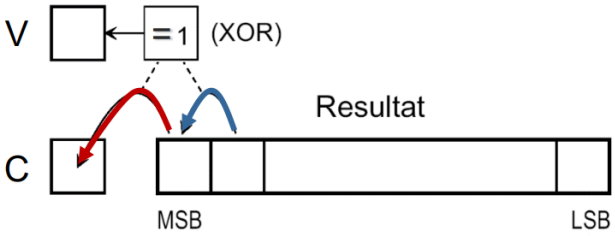
Use the filled in table to verify the correct behavior of your program. Are all results and flags shown correctly?

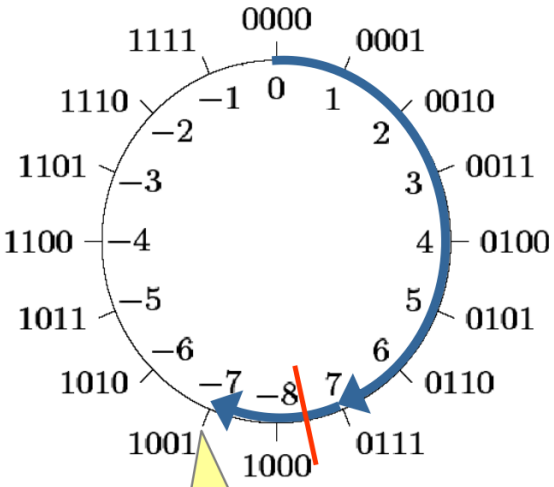
Processor always calculate C and V

N-Flag 1, wenn zu vorderst eine 1 steht
Z-Flag 1, wenn es null geht
V-Frag Addition 1, wenn overflow gibt nur die vordersten zwei bits anschauen. siehe bild unten
C-Flag Addition 1, wenn es einen carry gibt, Substraction 0, wenn borrow genutzt wird

Signed

- Overflow carry to MSB has different value than carry from MSB





- Examples for 4 cases

0 1	1 0	0 0	1 1
0111	1010	0001	0011
+0010	+1101	+1110	+1111
0 1 1 0	1 0 0 0	0 0 0 0	1 1 1 1
1001	0111	1111	0010
V = 1	V = 1	V = 0	V = 0

128

Figure 2: 8 bit circle of numbers

4 Task 2 – 64 Bit Addition

4.1 Implementation

Write an assembly program which has a 64 bit summation variable (unsigned interpretation). Every time you press button T0, the program shall read a 32 bit input value from the DIP-switches and shall add the read value to the summation variable. The sum shall be displayed continuously on the LCD display (See Figure 3). Use the given project frame *add64*.

- Use the binary interface of the LCD display (See page LCD Binary Interface on the CT-Wiki).

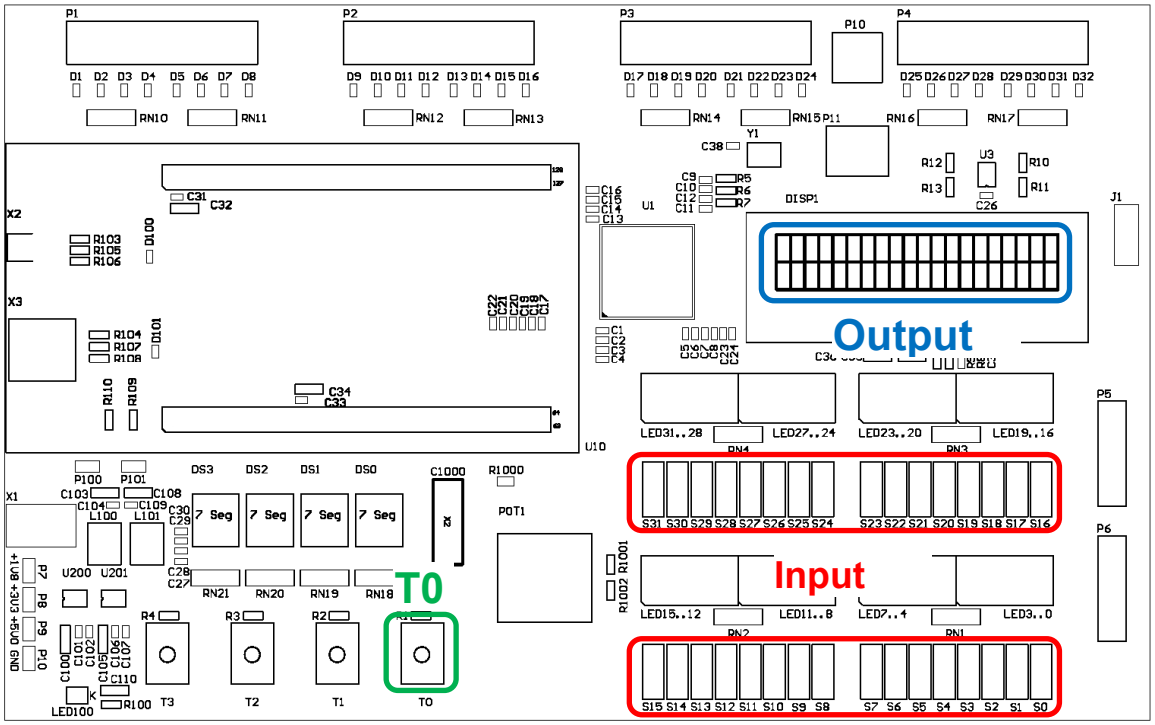


Figure 3: In and output of values for the 64 bit addition

4.2 Verification

Verify the results of your program, especially the overflow from one word to the other.

5 Task 3 – Arithmetic Commands (optional)

Assemble, link and load the given project *arith_operations* with the debugger. Execute the program step by step. Watch and comprehend the changes in the registries and flags.

6 Grading

The working programs have to be presented to the lecturer. The student has to understand the solution / source code and has to be able to explain it to the lecturer.

Criteria	**Weight**
The program meets the requirements described in Task 1	2/4
The program meets the requirements described in Task 2	2/4