EXAMPLE 7

OBJECTIVE
This simple example has the following objectives:

- Introduce 2-digit hex arithmetic with carry
- Introduce the ‘lost carry’ concept
- Introduce MSB carry

PROGRAM
Ex7.asm program performs the following arithmetic operations

- $15 + 27 = 42$ (decimal) using the opcode mnemonics LDAA, LDAB, ABA
- $0f + 1b = 2a$ (hex half carry - from bit 3) using the opcode mnemonics LDAA, LDAB, ABA
- $ff + 01 = 00$ (hex MSB carry) using the opcode mnemonics LDAA, ADDA
- $255 + 1 = 0$ (decimal equivalent of a hex MSB carry) using the opcode mnemonics LDAA, ADDA

Note: To facilitate understanding, the same numbers are used in both decimal and the hex cases.

CODING AND EXECUTION
Open THRSim11. Maximize THRSim11 window. Close the Commands window. Open file Ex7.asm. Assemble file. Tile the windows. Set a break point at the line containing SWI. Reset registers D, X, Y. Your screen should look like this:
Use the ‘Step’ button to step through the program. Press the Step button once. The registers do not change. You get to the beginning of a section in your program.

**SECTION: 15 + 27 = 42 (DECIIMAL) USING THE OPCODE MNEMONICS LDAA, LDAB, ABA**

Set the display option of accA and accB to Decimal.

Use the Step button to step through this section of your program. After 2 steps, the screen looks like this:

During the previous 2 operations, the numbers 15 and 27 have been loaded into accA and accB, respectively.

Press the step button once. The screen looks like this:
Congratulations! The first section in this example has been completed. You have achieve the decimal addition 15 + 27 = 42. AccA shows A 42, which is the expected result.

Note:

- A decimal carry happen during this operation, since, in the LSD, 5 + 7 = 12, ‘keep 2 and carry 1’.
- In the result, the LSD is 2. This signifies that of the result with carry, which was 12, the least significant digit, 2, was retained, and the most significant digit was carried in the next higher position.
- In the result, the MSD is 4. This signifies that the MSD of the result was obtained through the addition of the most significant digits of the constituents, 1 and 2, and the carry 1, i.e., 1+2+1 = 4.

SECTION: $0F + $1B = $2A (HEX HALF-CARRY) USING THE OPCODE MNEMONICS LDAA, LDAB, ABA

Reset accA and accB by typing zero into them. Set the display option of accA and accB to hex.

Your screen looks like this:
Use the Step button to step through this section of your program. After 2 steps, the screen looks like this:

During the previous 2 operations, the numbers $0f$ and $1b$ have been loaded into accA and acc B, respectively.

Press the step button once. The screen looks like this:
Congratulations! The second section in this example has been completed. You have achieved the hex addition $0f + $1b = $2a. AccA shows A $2a, which is the expected result.

**Note:**
- A hex half carry happen during this operation, since, in the LSD, $f + b = $1a, ‘keep a and carry 1’.
- In the result, the LSD is a. This signifies that of the result with carry, which was $1a, the least significant digit, a, was retained, and the most significant digit, 1, was carried in the next higher position.
- In the result, the MSD is 2. This signifies that the MSD of the result was obtained through the addition of the most significant digits of the constituents, 0 and 1, and the carry 1, i.e., $0 + 1 + 1 = 2$.

Note that the decimal equivalent of your result is $1a = 42$, i.e., the same as in the previous section.
$FF + $01 = $00 (MSB CARRY) USING THE OPCODE MNEMONICS LDAA, ADDA

Reset accA and accB by typing zero into them. Set the display option of accA and accB to hex.

Use the Step button to step through this section of your program. After the first step, your screen looks like this:

During this operation, the number $ff has been loaded into accA.

Press again the step button. Your screen looks like this:
During this operation, $01 was added to the number $ff existing in accA. If more than two hex digit could be displayed, the operation would have looked like $0ff + $001 = $100. Alas, the 8-bit architecture of our microcontroller does not allow for more than two hex digits. Hence, the third digit in the result is lost. (We have just experienced a 'lost carry'). The result is $00. The event that you have just witnessed is called hex MSB carry. However, the fact that this event took place has been recorded by the microcontroller logic by setting the bit 0 in the condition code (CC) register. Before this operation, the CC register was %1111000; after this operation, the CC register has become %11110101. The bits that have been set by this operation are:

- C = 1 (bit 0 in CC register), signifying that a MSB carry took place during the operation
- Z = 1 (bit 2 in CC register), signifying that the result of the operation was zero

Also affected by this operation was the bit N (CC register bit 3) which switched from 1 to 0. The meaning of this will be explained in a later example, in which negative numbers are studied.

255 + 1 = 0 (DECIMAL EQUIVALENT OF HEX MSB CARRY) USING THE OPCODE MNEMONICS LDAA, ADDA

Reset accA and accB by typing zero into them. Set the display option of accA and accB to decimal. After the first step, your screen looks like this:

During this operation, the decimal number 255 has been loaded into accA. (This decimal number is the decimal equivalent of the hex number $ff used in the previous section.)

Press again the step button. Your screen looks like this:
During this operation, 1 was added to the number 255 existing in accA. The result should have been 256. No decimal carry happened. However, the result is too large to be stored in 2-digit hex format, since $256 = 100$. Hence, the third highest digit in the hex representation is lost, and only the last two digits are retained. The displayed result $00$, i.e., 0 in decimal. This proves again that, internally, the MCU works in hex!
WHAT YOU HAVE LEARNED

In this example, you have learned the following:

- In hex arithmetic, the symbol furthest to the left is also called the *most significant digit* (MSD). In number $1d$, the most significant digit is 1. It signifies the number of sixteens contained in the number. In our case, we have only one sixteen, since $1 \times $sixteen = sixteen

- In hex arithmetic, the symbol furthest to the right is called the *least significant digit* (LSD). In number $1d$, the least significant digit is d. It signifies the number of units contained in the number. In our case, we have thirteen units, since the hex number d represents the number thirteen, and $d \times $units =thirteen.

- The hex number $1d$ has the value twenty-nine, since sixteen plus thirteen makes twenty-nine.

- You have observed a hex MSB carry. In our case, the machine has 2-digit hex precision; hence, it cannot store numbers with more than two digits. When to the number $ff$ we added the number $01$, the result should have been $100$ (Proof: $0ff + 001 = 100$). But our MCU we cannot store more than 2 hex digits in single precision, hence, the leading 1 is lost ('lost carry'). The phenomenon is called *MSB carry*. (The name MSB signifies *most significant bit* and it refers to the binary equivalent of the hex number).

- You have also observed the decimal representation of a hex MSB carry. This happened when to the decimal number 255 we added the number 1. The result was 0, instead of 256 as you would have expected. This illustrates the range limitation of your MCU (8-bit, 2-hex). Also illustrated is the fact that, internally, MCU works in hex. YOU SHOULD TRAIN YOURSELF TO THINK IN HEX TO BETTER UNDERSTAND THE WORKING OF THE MCU!

- New words and notations: most significant digit (MSD), least significant digit (LSD), 8-bit arithmetic, most significant bit (MSB), addition with carry, hex arithmetic with carry, 'lost carry', hex half carry, MSB carry.
(This page is left intentionally blank)