EXAMPLE 4

OBJECTIVE
This simple example has the following objectives:

- Introduce *hex numbers and hex symbols*, a, b, c, d, e, f.
- Introduce *single precision and double precision* registers
- Introduce *hex arithmetic*.

PROGRAM
Ex4.asm program is identical to ex3.asm, but uses hex numbers. It performs the arithmetic operation $07 + 06 = 0d$. To achieve this, the program does the following:

- Load the number $07$ into accA using LDAA, i.e., $07 \rightarrow A$
- Load the number $06$ into accB using LDAB, i.e., $06 \rightarrow B$
- Add the number in accA (i.e., $07$) with the number in accB (i.e., $06$) $A + B \rightarrow A \uparrow A \downarrow 0d$

During this process, the number $07$ was incremented six times by one. Sequentially, it went through $08, 09, 0a, 0b, 0c, and finally, 0d$. Hence, to achieve $07 + 06$ we can choose two paths:

- Path 1: start with $07$ in accA and increment six times using the INCA opcode.
- Path 2: Start with $07$ in accA and $06$ in accB and do the addition using ABA.

EXECUTION
Open THRSim11. Maximize THRSim11 window. Close the Commands window. Open file Ex3.asm. Assemble file. Tile the windows. Set break point at the line containing SWI. Set the display of A, B registers to ‘Hexadecimal’. Reset registers D, X, Y. Your screen should look like this:
Note that, in single-precision 8-bit arithmetic, hex numbers are always represented with two digits, even if the most significant digit (MSD) is zero. For example, the single-precision accA and accB are represented as $00 and $00. In double precision 8-bit arithmetic, hex numbers are always represented with four digits, even if the more significant ones are zero. For example, double precision accD and index registers X and Y are all represented as $0000. This convention helps us discern quickly between the single precision and the double precision registers.

Use the ‘Step’ button to step through the program. Press the Step button once. The registers do not change. Press again. Your screen should look like this:

During the operation, number 7 has been loaded into accA as $07. Also changed is accD, which results from concatenation of accA:accB. Since accA is $07 and accB is $00, accD is $0700. This concatenation process is always done in hex format, and is not apparent when decimal display is used (check that this is so by looking into Example 1 through Example 3).

From here on, two paths are possible. They will be followed in sequence.
**PATH 1**

Step again. AccA has incremented by one from $07 to $08. The screen should look like this:

Continue stepping and watch accA going sequentially through $0b, $0c, $0d. Finally, the screen should look like this:
This is result of the addition $07 + $06 = $0d, achieved through successive incrementation.
**PATH 2**

The PC is placed at beginning of Path 2. Step once. The accA is reloaded with $07. Your screen looks like this:

During the operation, number 6 has been loaded into accB as $06. Since accA is $07 and accB is $06, accD is $0706.
Now, step again to perform the addition $07 + 06 = 0d$. The screen looks like this:

During the last operation, AccA and AccB have been added together. The result of the operation $07 + 06 = 0d$ appears in accA. The number 06 is still preserved in AccB.

**WHAT YOU HAVE LEARNED**

In this simple exercise, you have understood how hex arithmetic works:

- In hex arithmetic, the base of the number system is *sixteen* (hexadecimal).
- Sixteen independent symbols exist for representing the first sixteen numbers in the hex system: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f.
- The greatest number that can be represented with a single hex symbol is fifteen.
- In single-precision 8-bit arithmetic, hex numbers are always represented with two digits, even if the most significant digit (MSD) is zero. For example, you saw during this exercise how the single-precision accA and accB took values $07$ and $06$. Later on, accA became $0d$.
- In double precision 8-bit arithmetic, hex numbers are always represented with four digits, even if the more significant ones are zero.
- In hex arithmetic, single-precision numbers can be built into double precision numbers through *concatenation* (i.e., by putting them side by side). For example, when accA was $07$ and accB was $06$, accD, which is simply accA:accB, was $0706$. This was not so in decimal arithmetic (check that this is so by looking into Example 2 and Example 3).
- New words: hex arithmetic, hexadecimal, single precision, double precision, 8-bit arithmetic, concatenation.