DT228-1 Micros Lab. “Fun with flags”

Key topics: conditional jumps, multi-word arithmetic, byte ordering.

Introduction.

In the previous lab you saw how the arithmetic flags in the Cortex M0 processor were changed by the ALU when it performed different calculations. In this lab you will see how these flags are used to control program flow and perform multi-word arithmetic.

Tasks

(a) Download and extract the support zip file for this lab. This file contains a two programs. The first program (ArmAssembler2) adds together an array of constant values to produce a 32 bit result. Run this program in the debugger and carefully note the state of the arithmetic flags in the xPSR (program status register) each time the CMP instruction is executed. Verify that the conditional jump instruction that follows is behaving correctly.

Modify this program so that it behaves in a manner similar to the following ***while*** loops:

```
Count = 4;
 :
while (Count !=0)
{
  :
 Count--;
}
```

```
Count = 0;
 :
while (Count <=4 )
{
  :
 Count++;
}
```

```
Count = 4;
 :
while (Count >=0 )
{
  :
 Count--;
}
```

(b) The second program (ArmAssembler3) adds together two 64 bit numbers. The Cortex M0 processor has 32 bit registers and must perform 64 bit arithmetic in two stages. It must first add the low order “words” (lower 32 bits in this case). It must then add the two high order words being careful to include any carry (if one exists) from the low order addition. Run the program and experiment with different operands to confirm that the addition is being carried out correctly (note the difference between the two variations of the add instruction). Change the ADCS instruction to an ADDS instruction and note how the calculations are now incorrect.
Modify the program so that it subtracts two 64 bit values – verify that it is including borrows from the lower order subtraction.
Hint: The subtract with borrow instruction is SBCS
Point of interest: All data in this program is stored in ***Little Endian*** format.