# Assembly language programming

## Exercise 1

Write an Assembly language program that calculates the difference between two signed numbers. Program must return (=store) the difference in register R0. The returned difference must be a positive number.

Copy the template code from the workspace to your project and implement the computation.

```
__attribute__(( naked )) int difference(int a, int b)
{
    // the values passed to your program are in registers
    // R0 = a, R1 = b
    // write your code between push and pop instructions
    // make sure return value is in R0 after calculation
    asm volatile
    (
        "push {r4, r5, r6, r7} \n"
        // execute compare instruction to set flags before conditional branch
        // "cmp ..... \n"
        // some code here
        "pop {r4, r5, r6, r7} \n"
        "bx lr \n"
    );
}
```

## Notes on using labels

```
asm volatile
(
    "push {r4, r5, r6, r7} \n"
    "loc1:
     mov r4, #0 \n"
     // some instructions
     // execute compare instruction to set flags before conditional branch
    "beq loc1 \n"
    "pop {r4, r5, r6, r7} \n"
    "bx lr \n"
);
```

This is a label definition. **The definition ends with a colon** and label refers to the next instruction after label definition. You can jump to the instruction by using the label as the branch target address. You can define any number of labels provided that they have unique names.

Here we make a conditional branch. If the condition (EQ) is true then we branch to the label. In this example mov r4, #0 is the next instruction after the label so that would be instructions that is executed after the branch. Note than there is **no colon when we use the label in a branch instruction**.

See page 4-15 in Thumb-2SupplementReferenceManual.pdf (in workspace) for list available of instructions and QRC0001_UAL.pdf for quick reference of what each instruction does.