SCC.150 ASSEMBLER PRACTICAL 3
WEEK 14

The task for this practical is to write an assembler program which can multiply two integer numbers. However, the program should not use the *mult* instruction to accomplish this. Instead, the program should perform the multiplication using the *Ethiopian Multiplication* (also known under other names). Multiplication of two numbers is achieved using shift operations and simple addition. The next sections detail how the Ethiopian multiplication works and define the requirements of the program.

This practical is the second coursework for the SCC150 course. You must upload your code in Moodle by the end of week 16. Because this practical will be marked using an automated marking tool, you should precisely follow the instruction, otherwise you risk loosing marks and failing the coursework.

Please be aware that the resulting program will be assessed in the Week18 practical. You will need to sit together with a TA and answer questions about your code. During that session we will also receive feedback.

1. Ethiopian Multiplication

Two numbers a and b can be multiplied by repeatedly halving a (discarding remainders) and doubling b (see table below). The value of b is added to the result if the value in a is odd. The procedure stops when a reached 1. In the shown example we compute $r = a \cdot b$ with $a = 17$ and $b = 34$. 17 is odd and therefore $r = r + b = 0 + 34$. Halving $a = 17$ results in $a = 8$ when discarding the remainder. Doubling $b = 34$ results in $b = 68$. $a = 8$ is even and therefore we do not add $b = 68$ to the result r. The process continues until we reach $a = 1$ and $b = 544$. 1 is odd and therefore $r = r + b = 34 + 544 = 578$. a has reached 1 and the algorithm terminates leaving the multiplication result of $r = a \cdot b = 17 \cdot 34 = 578$.

a	b
17	34
8	~~68~~
4	~~136~~
2	~~272~~
1	544
	578

2. Hints

The previously outlined Ethiopian Multiplication can be implemented using assembler instructions. `sll` and `srl` are shift operations which can be used to implement doubling and halving (multiplication by 2, division by 2). `and` can be used to implement a test to check if a number is odd or even (is the last bit set?).

3. Program Requirements

A program for Ethiopian Multiplication should be constructed. The program should read the two numbers a and b to be multiplied from memory. Then a function (procedure) should be called which receives a and b as parameters. The function then carries out the multiplication and returns the result. The result is then stored in memory and the result is printed on screen using a *syscall*. In detail the requirements are:

- a should be read from address *0x10010004*.
- b should be read from address *0x10010008*.
- After reading a and b from memory a procedure should be called which implements the ethiopian multiplication.
- The procedure should return the multiplication result which is then stored at address *0x10010000*.
- In addition, the multiplication result should be printed on screen by the main program using a *syscall*.