**CS1005 Logic & Computation  
Lab sheet 4: More on Turing Machines**

This lab will

* Reinforce and deepen your understanding of **arithmetic over binary numbers**,
* Help you to develop your skills in the **design of computational solutions** within the framework of Turing machines.

1. Write down the binary numbers corresponding to decimal 1 to 20 inclusive.

1 = 1

2 = 10

3 = 11

4 = 100

5 = 101

6 = 110

7 = 111

8 = 1000

9 = 1001

10 = 1010

11 = 1011

12 = 1100

13 = 1101

14 = 1110

15 = 1111

16 = 10000

17 = 10001

18 = 10010

19 = 10011

1. = 10100
2. Do the following additions in binary notation:

Example: 3+7 = 11 + 111 = 1010

* + - * 0+0 **= 0+0 = 0**
      * 1+0 **= 1+0 = 1**
      * 0+1 **= 0+1 = 1**
      * 1+1 **= 1+1 = 10**

1. Do the following multiplications in binary notation

Example: 3\*4 = 11 \* 100 = 1100

* 1\*1 = 1\*1 = 1
* 2\*1 = 10\*1 = 10
* 1\*2 **= 1\*10 = 10**
* 2\*2 **= 10\*10 = 100**
* 4\*2 **= 100\*10 = 1000**
* 7\*2 **= 111\*10 = 1110**
* 13\*2 = 1101\*10 = 11010
* 13\*4 = 1101\*100 = 110100
* 13\*8 = 1101\*1000 = 1101000
* 13\*16 = 1101\*10000 = 11010000
* What is the rule for multiplication by 2 in binary? Add a zero on the end
* What are the rules for multiplication by 4, 8 and 16 in binary? Add two zeros (4) and three zeros (8) add four zeros (16)

4. Create a Turing machine function (table) to

* **Add 2 to any number** in binary arithmetic,

Hint: use 2 copies of the +1 Turing table from the lecture. Join them together carefully into one big table, making sure that the first half ‘runs into’ the second half without HALTing prematurely. You will also need to make sure that the states in the second half of the big table have different names from the states in the first half.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current State** | **Current cell content** | **Value to write** | **Direction to move** | **New state to enter** |
| START | \* | \* | Left | ADD |
| ADD | 0 | 1 | Right | RETURN |
| ADD | 1 | 0 | Left | CARRY |
| ADD | \* | \* | Right | HALT |
| CARRY | 0 | 1 | Right | RETURN |
| CARRY | 1 | 0 | Left | CARRY |
| CARRY | \* | 1 | Left | OVERFLOW |
| OVERFLOW | Ignored | \* | Right | RETURN |
| RETURN | 0 | 0 | Right | RETURN |
| RETURN | 1 | 1 | Right | RETURN |
| RETURN | \* | \* | No Move | HALT |

* and demonstrate its operation on
* 1+2. Hint: encode the ‘1’ on the tape, and run your TM+2 on that tape.
* 2+2 Hint: encode the ‘2’ on the tape, and run your TM+2 on that tape

5. (a) Design a simple Turing function (table) to

* **multiply any number by 2** in binary arithmetic, and demonstrate its operation on
* 1\*2
* 3\*2

(b) There is another more complicated way to design a Turing function to multiply by 2 in binary arithmetic. Can you sketch it?

6. Show how you would modify your answer to (5) to make a TM function to multiply any number by 4 in binary arithmetic, and demonstrate its working on 3\*4 where 3 in binary is on the input tape. Give the final tape.

7. Why are algorithms called algorithms?

8. Which languages are interpreted, compiled or byte-coded?

* Java
* Perl
* C
* C++
* Ruby
* Prolog
* Python
* Fortran
* Scratch