# Linnaeus University

## 1DT301 - Computer Technology 1
## Assignment 1

Group number: Group I
Group member: Jiaxin Wang, Zejian Wang
Student ID: jw223jc@student.lnu.se
zw222bb@student.lnu.se

Task1

Run the program in LEGv8 simulator, number 17 is stored in register x2.

Task2

Translating machine code instructions to assembly code

(1) 11010010100000000001000000000010
According the LEGv8 reference sheet, it follows IW format(opcode + MOV_immediate + Rd) and can be presented as the following:
11010010100  0000000010000000  00010
Translated as "**MOVZ  X2, #128**" in assembly code.

(2) 11010010100000000001110011100100
It also follows IW format(opcode + MOV_immediate + Rd) and can be presented as:
11010010100  0000000011100111  00100
Translated as "**MOVZ  X4, #231**" in assembly code.

(3) 11001011000000100000000010000101
It follows R format(opcode + Rm + shamt + Rn + Rd) and can be presented:
11001011000  00010  000000  00100  00101
Translated as "**SUB  X4, X5, X2**" in assembly code.

(4) D360 0CA5
First translate this hexadecimal number to Binary numbers:
11010011011000000000110010100101, which follows R format(opcode + Rm + shamt + Rn + Rd) and can be presented as
11010011011  00000  000011  00101  00101
Translated as "**LSL  X5, X5, #3**" in assembly code.

## Task3

Calculating the following in LEGv8 assembly code.

$$4 \cdot 5 + 16 \cdot 11 + 25$$

This expression can be presented as $2^2 \cdot 5 + 2^4 \cdot 11 + 25$ and assembly code is the following:

**MOVZ X1, #5** //Store number 5 in register1

**LSL X1, X1, #2** // Shift what stored in register1 2 steps left and the value is $2^2 \cdot 5$

**MOVZ X2, #11** // Store number 11 in register2

**LSL X2, X2, #4** //Shift what stored in register2 4 steps left and the value is $2^4 \cdot 11$

**MOVZ X3, 25** //Store number 25 in register3

**ADD X0, X1, X2** //Add numbers in register1 to register 2 and store the result to register0

**ADD X0, X0, X3** //Add numbers in register0 to register3 and store the result to register0

Run the program and the result is 221 in decimal.

## Task4

calculate 1 893 423 + 443 924 in assembly program.

Those numbers are too large to directly be put in registers, so we change them into values on base 2.

1893423 is 00011100 1110010000101111 , the last 16 bits can be transferred to 58415 in base 10. The first 8 bits is literally 28 in base 10 but it needs to use LSL to shift left it 16 bits.

443924 is 0110 1100011000010100, the last 16 bits can be transferred to 50708 in base 10. The first 4 bits is literally 6 in base 10 but it needs to use LSL to shift left it 16 bits.

**MOVZ X1, #28, LSL 16** // shift 28 left for 16 bits and store the value in X1.

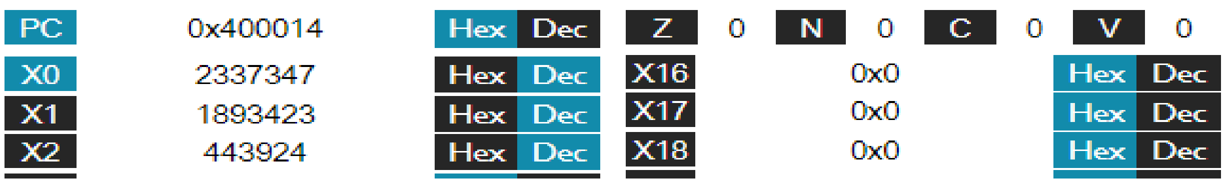
**MOVK X1, #58415, LSL 0** // Use MOVK to combine the value in X1 and 58415 in base 10

**MOVZ X2, #6, LSL 16** // shift 6 left for 16 bits and store the value in X2.

**MOVK X2, #50708, LSL 0** // Use MOVK to combine the value in X2 and 50708 in base 10

**ADD X0, X1, X2** // Add the values in X1 and X2 and store it in X0

The result is 2337647 in base 10.



## Task 5

The code is shown as below:

```
MOVZ    X1, #0                   // store the sum in register X1
MOVZ    X2, #1                   // store the number of items in register X2
loop:
        ADD     X3, X2, X2       // two times of the number of items
        SUBI    X3, X3, #1       // two times of the number of items
                                    minus one to get the value we need to
                                    add to the sum
        ADD     X1, X1, X3       // Add the value stored in X3 to sum
        ADDI    X2, X2, #1       // the number of items plus one
        SUBI    X4, X2, #50      // calculate the difference between the
                                    value in X2 with 50, because there are
                                    50 items that need to be added up, and
                                    store the difference in X4
        CBNZ    X4, loop         // If the difference is not zero, branch to
                                    loop and iterate, otherwise branch to
                                    exit
        B       exit
exit:
        ADDI    X1, X1, #99      // The 50th item (99) is added to the sum
```

The result can be seen as below:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PC | 0x400028 | Hex Dec | Z | 0 | N | 0 | C | 0 | V | 0 |
| X0 | 0 | Hex Dec | X16 | 0x0 | | | | | | Hex Dec |
| X1 | 2500 | Hex Dec | X17 | 0x0 | | | | | | Hex Dec |
| X2 | 50 | Hex Dec | X18 | 0x0 | | | | | | Hex Dec |
| X3 | 97 | Hex Dec | X19 | 0x0 | | | | | | Hex Dec |
| X4 | 0 | Hex Dec | X20 | 0x0 | | | | | | Hex Dec |

The sum is 2500 in X1.

Task 6

The code is shown as below:

```
MOVZ  X3, #0            // store the beginning of index in register X3
loop:
      LSL   X2, X3, #3    // Use LSL to get the byte address of each
                          index
      ADD   X2, X2, X7    // Add the byte address and the base
                          address together to get the real address
      LDUR  X4, [X2, #0]  // Load the value from the real address
                          stored in X2 to temporary register X4
      ADD   X0, X0, X4    // Add the values in X4 and X0 together
                          and store it in X0
      ADDI  X3, X3, #1    //  index plus 1
      SUBI  X5, X3, #6    // calculate the difference between the
                          value in X3 with 6, and store it in X5
      CBNZ  X5, loop      // if the difference in X5 is not zero, then
                          branch to loop and iterate, otherwise
                          always branch to exit
      B     exit
exit:
```

The result can be seen as below:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PC | 0x400058 | Hex Dec | Z | 0 | N | 0 | C | 0 | V | 0 |
| X0 | 22 | Hex Dec | X16 | 0x0 | | | | | | Hex Dec |
| X1 | 2 | Hex Dec | X17 | 0x0 | | | | | | Hex Dec |
| X2 | 268435496 | Hex Dec | X18 | 0x0 | | | | | | Hex Dec |
| X3 | 0x6 | Hex Dec | X19 | 0x0 | | | | | | Hex Dec |
| X4 | 2 | Hex Dec | X20 | 0x0 | | | | | | Hex Dec |
| X5 | 0x0 | Hex Dec | X21 | 0x0 | | | | | | Hex Dec |
| X6 | 0x0 | Hex Dec | X22 | 0x0 | | | | | | Hex Dec |

The result is 22 in X0.