Lab B2

TT0L - GROUP 0

Person 1	1111111111
Person 2	1111111111

Q1. Write ARM instructions to find the

I. 1’s complement and 2’s complement of a 32 bit number in location ‘X’ and store the result in memory.

```
1           MOV       R1, #0x1000
2           MOV       R2, #0xC0000034
3           STR       R2, [R1]
4
5           LDR       R0, [R1]
6
7           MVN       R5, R0 ;1's complement
8           ADD       R6, R5, #0x01 ;2's complement
9
10          MOV       R7, #0x7000
11
12          STR       R5, [R7], #0x04
13
14          STR       R6, [R7]
15
16          END
17
```

Start address: End address: 0x7004

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x1000	0xC0	0x0	0x0	0x34	0xC0000034
0x7000	0x3F	0xFF	0xFF	0xCB	0x3FFFFFCB
0x7004	0x3F	0xFF	0xFF	0xCC	0x3FFFFFCC

II. 1’s complement and 2’s complement of a 64-bit number in locations ‘X’ and ‘X+1’ (lower order first followed by higher order) and store the result in consecutive memory locations.

```
1         MOV       R1, #0x6300 ;mem location
2         MOV       R2, #0xFF000000 ;high order
3         MOV       R3, #0xC0000034 ;low order
4
5         STR       R3, [R1], #0x0004 ;Store value of R3 to R1 (0x6300), then increment R1 to 0x6304
6         STR       R2, [R1], #-0x0004 ;Store value of R3 to R1 (0x6304), then decrement R1 back to 0x6300
7
8         LDR       R0, [R1], #0x0004 ;Load value of R1 to R0, then increment R1 to 0x6304
9         LDR       R10, [R1], #-0x0004 ;Load value of R1 to R10, then decrement R1 back to 0x6300
10
11        MVN       R5, R0 ;1's complement
12        MVN       R11, R10 ;1's complement
13
14        ADDS      R6, R5, #0x0001 ;2's complement (the use of S to enable conditional flag)
15        ADC       R7, R11, #0x0000 ;2's complement (uses ADC to add together with the carry flag)
16
17        STR       R5, [R1, #0x0100]! ;Store value of R5 to R1 + 0x100 (0x6400)
18        STR       R11, [R1, #0x0004]! ;Store value of R11 to R1 + 0x4 (0x6404)
19        STR       R6, [R1, #0x0004]! ;Store value of R6 to R1 + 0x4 (0x6408)
20        STR       R7, [R1, #0x0004]! ;Store value of R7 to R1 + 0x4 (0x640C)
```

Start address: 0x6300 End address: 0x7000

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x6300	0xC0	0x0	0x0	0x34	0xC0000034
0x6304	0xFF	0x0	0x0	0x0	0xFF000000
0x6400	0x3F	0xFF	0xFF	0xCB	0x3FFFFFCB
0x6404	0x0	0xFF	0xFF	0xFF	0xFFFFFF
0x6408	0x3F	0xFF	0xFF	0xCC	0x3FFFFFCC
0x640C	0x0	0xFF	0xFF	0xFF	0xFFFFFF

Q2.

```
1           MOV       R1, #0x0009
2           MOV       R2, #0x0008
3           MOV       R11, #0x0000
4
5           MOV       R9, #0x7000 ;mem addr for end result
6           MOV       R0, #0x6000 ;mem addr for operand 1 & 2
7
8           STR       R1, [R0], #0x0004
9           STR       R2, [R0], #-0x0004
10
11          LDR       R3, [R0], #0x0004
12          LDR       R4, [R0]
13
14          ADD       R5, R3, R4
15          CMP       R5, #0x000A ;compare by substracting 0x000A from the value of R5 (0x0011)
16          BLT       STORE ;skip to STORE (line 24-25) if the value of R5 is less than 0x000A
17
18          ADD       R6, R5, #0x0006
19          AND       R7, R6, #0x000F
20          MOV       R11, #0x0001
21
22 STORE
23          STR       R7, [R9], #0x0004 ;store the value of R7 to R9 (0x7000), then increment R9 by 0x0004
24          STR       R11, [R9] ;store the value of R7 to R9 (0x7004)
```

Start address: 0x6000 End address: 0x7500

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x6000	0x0	0x0	0x0	0x9	0x9
0x6004	0x0	0x0	0x0	0x8	0x8
0x7000	0x0	0x0	0x0	0x7	0x7
0x7004	0x0	0x0	0x0	0x1	0x1

Q3.

a)

```
1           MOV       R1, #0x08
2           MOV       R2, #0x5000
3           STR       R1, [R2]
4           LDR       R3, [R2]
5
```

View Memory Contents

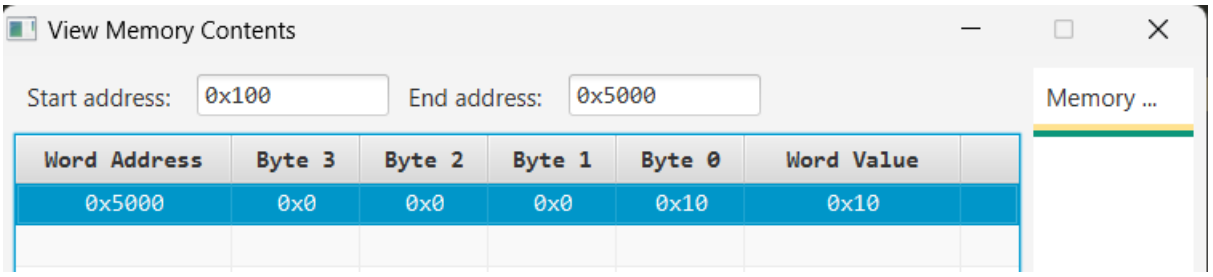
Start address: 0x100 End address: 0x5000

Memory ...

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x5000	0x0	0x0	0x0	0x8	0x8

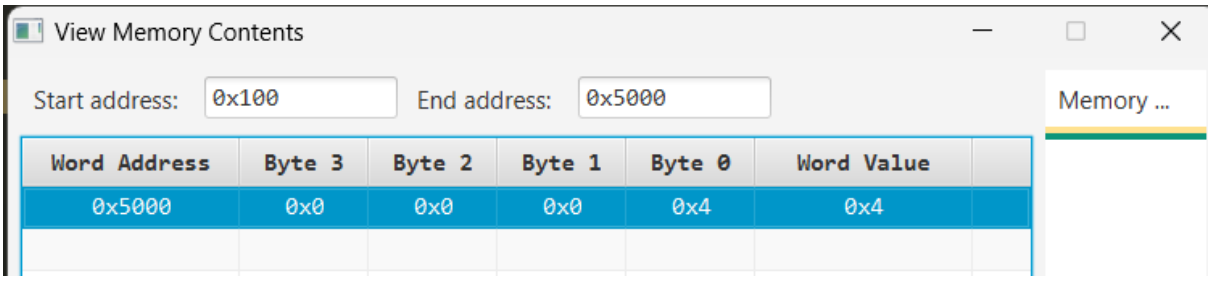
b)

```
1           MOV     R1, #0x08
2           MOV     R2, #0x5000
3           STR     R1, [R2]
4           LDR     R3, [R2]
5           MOV     R5, R3, LSL #1 ; multiplication by 2
6           STR     R5, [R2],#4
```



c)

```
1           MOV     R1, #0x08
2           MOV     R2, #0x5000
3           STR     R1, [R2]
4           LDR     R3, [R2]
5           MOV     R6, R3, LSR #1 ; division by 2
6           STR     R6, [R2]
```



Q6.

i) LOAD IMMEDIATE 30: The instruction loads the immediate value 30 into the accumulator.

Accumulator value: 30

ii) LOAD DIRECT 30: The instruction loads the value stored in memory location 30 into the accumulator.

Accumulator value: 40

iii) LOAD INDIRECT 30: The instruction loads the value stored in the memory location pointed to by the value stored in memory location 30 into the accumulator. The value stored in memory location 30 is 40, so the value at the memory location 40 is loaded into the accumulator.

Accumulator value: 50

iv) LOAD IMMEDIATE 10: The instruction loads the immediate value 10 into the accumulator.

Accumulator value: 10

v) LOAD DIRECT 40: The instruction loads the value stored in memory location 40 into the accumulator.

Accumulator value: 50

vi) LOAD INDIRECT 10: The instruction loads the value stored in the memory location pointed to by the value stored in memory location 10 into the accumulator. The value stored in memory location 10 is 20, so the value at the memory location 20 is loaded into the accumulator.

Accumulator value: 30