

Experiment 4

Aim:

To write an ARM Assembly Language to

- a) Add two 64 bit numbers.
- b) Add ten 32 bit numbers.

Tool Used:

Keil uVision4

Theory:

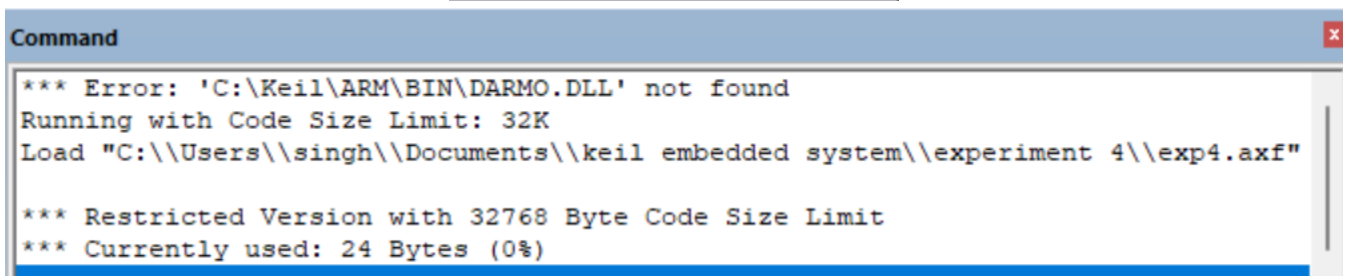
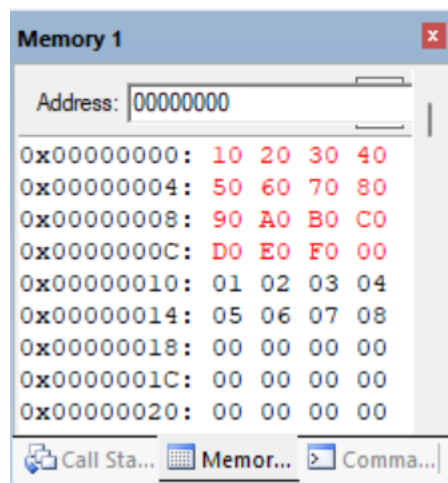
LDM load multiple register locations with starting address mentioned. ! is used in LDM for updating pointer, else same value will be updated in all registers. STM load the value into consecutive memory locations with starting address mentioned. ADDCS adds the value if the carry flag is set.

a) Add two 64 bit numbers.

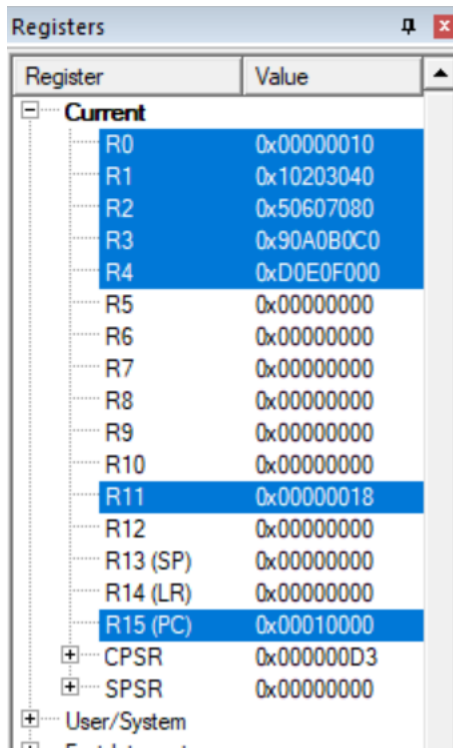
Code:

```
AREA PROGRAM, CODE, READONLY
ENTRY
MAIN
LDR R0, =0X00000000
LDM R0!, {R1-R4}
ADDS R6, R2, R4
ADCS R5, R1, R3
LDR R7, =0X00000010
STM R7!, {R5-R6}
END
```

Output:



Register Contents



Register	Value
Current	
R0	0x00000010
R1	0x10203040
R2	0x50607080
R3	0x90A0B0C0
R4	0xD0E0F000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000018
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00010000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	

b) Add Ten 32 bit numbers.

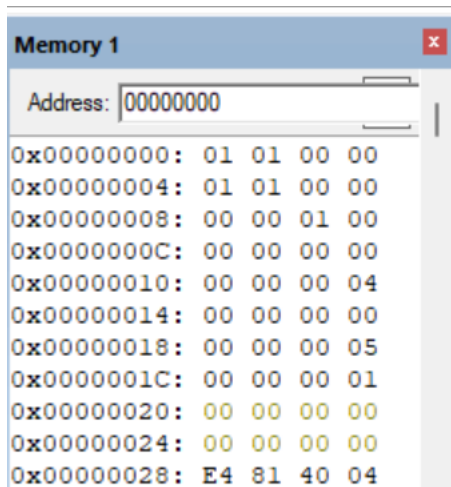
Code:

```
AREA PROGRAM, CODE, READONLY
ENTRY
MAIN
    LDR R0, =0x00000000
    LDR R1, =0x00000020
    MOV R3, #9
    LDR R4, [R0]

    LOOP ADD R0, R0, #4
        LDR R5, [R0]
        ADDS R4, R4, R5
        ADCS R7, R7, #1
        SUBS R3, R3, #1
        BNE LOOP

    STR R4, [R1], #4
    STR R7, [R1]
    END
```

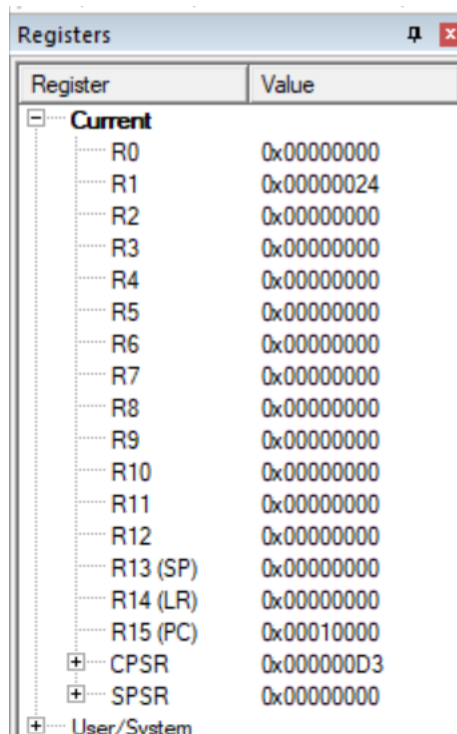
Output:



A screenshot of a 'Memory 1' window. At the top, there is a text field labeled 'Address:' containing '00000000'. Below this, a list of memory addresses and their corresponding hexadecimal values is displayed. The values are shown in groups of four hex digits. Addresses 0x00000020 through 0x00000024 are highlighted in yellow.

Address	Value
0x00000000	01 01 00 00
0x00000004	01 01 00 00
0x00000008	00 00 01 00
0x0000000C	00 00 00 00
0x00000010	00 00 00 04
0x00000014	00 00 00 00
0x00000018	00 00 00 05
0x0000001C	00 00 00 01
0x00000020	00 00 00 00
0x00000024	00 00 00 00
0x00000028	E4 81 40 04

Register Contents



A screenshot of a 'Registers' window. It displays a table of registers and their current values. The registers are grouped into 'Current', 'CPSR', 'SPSR', and 'User/System' categories. The 'Current' group is expanded, showing registers R0 through R15, R13 (SP), R14 (LR), and R15 (PC). The 'CPSR' and 'SPSR' registers are also shown, along with the 'User/System' category.

Register	Value
Current	
R0	0x00000000
R1	0x00000024
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00010000
CPSR	0x000000D3
SPSR	0x00000000
User/System	

Result:

The experiments on add operations have been performed and verified to be correct.