

Lab-4

Obj-1: Perform Addition and Subtraction of two 32-bit numbers using data processing addressing mode (with immediate data).

Program:

```
AREA PROG1, CODE, READONLY
```

```
ENTRY
```

START

```
MOV R0, #0X40
```

```
MOV R1, #0X50
```

```
ADDS R2, R0, R1
```

```
SUBS R3, R0, R1
```

```
MULS R4, R0, R1
```

MY_EXIT

```
B MY_EXIT
```

```
END
```

Result :

The screenshot displays the Keil uVision IDE interface. On the left, the 'Registers' window shows the current state of the ARM registers. On the right, the 'Disassembly' window shows the assembly code being executed, and the 'OBJ3.ASM' window shows the source code.

Registers Window:

Register	Value
R0	0x00000040
R1	0x00000050
R2	0x00000090
R3	0xffffffff
R4	0x00001400
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000

Disassembly Window:

```
9: MY_EXIT
0x00000010 E0140190 MULS    R4, R0, R1
10: B MY_EXIT
->0x00000014 EAffffff B      0x00000014
0x00000018 00000000 ANDEQ   R0, R0, R0
0x0000001C 00000000 ANDEQ   R0, R0, R0
0x00000020 00000000 ANDEQ   R0, R0, R0
0x00000024 00000000 ANDEQ   R0, R0, R0
```

OBJ3.ASM Window:

```
1 AREA PROG1, CODE, READONLY
2 ENTRY
3 START
4 MOV R0, #0X40
5 MOV R1, #0X50
6 ADDS R2, R0, R1
7 SUBS R3, R0, R1
8 MULS R4, R0, R1
9 MY_EXIT
10 B MY_EXIT
11 END
```

INPUT

ML	DATA
-	0X40 (R0)
-	0X50 (R1)

OUTPUT

ML	DATA
-	0X90 (R2)
-	0Xffffff0 (R3)
-	0X1400 (R4)

OR

Objective-1

Program:

```
AREA prg1, CODE, READONLY
ENTRY
START
    LDR R0,=0xAB000002
    LDR R1,=0x1200000c
    adds R2,R0,R1
    subs R3,R0,R1
    mul R4,R0,R1
my_exit b my_exit
END
```

RESULT:

The screenshot shows a debugger window with two main panes. The left pane, titled 'Registers', displays a list of registers (R0 to R15, CPSR, SPSR) and their current values. The right pane, titled 'Disassembly', shows the assembly code being executed, with a yellow highlight on the instruction at address 0x00000014: `EAFFFFFFE B 0x00000014`. Below the disassembly pane, a window titled 'OBJ3.ASM' shows the source code of the program, with the instruction `my_exit b my_exit` highlighted in green. The registers pane shows the following values: R0: 0xAB000002, R1: 0x1200000C, R2: 0xBD00000E, R3: 0x98FFFFFF6, R4: 0x28000018, R5: 0x00000000, R6: 0x00000000, R7: 0x00000000, R8: 0x00000000, R9: 0x00000000, R10: 0x00000000, R11: 0x00000000, R12: 0x00000000, R13 (SP): 0x00000000, R14 (LR): 0x00000000, R15 (PC): 0x00000014, CPSR: 0xA00000D3, SPSR: 0x00000000.

Register	Value
R0	0xAB000002
R1	0x1200000C
R2	0xBD00000E
R3	0x98FFFFFF6
R4	0x28000018
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0xA00000D3
SPSR	0x00000000

```
8:      mul R4,R0,R1
0x00000010 E0040190 MUL      R4,R0,R1
9: my_exit b my_exit
->0x00000014 EAFFFFFFE B      0x00000014
0x00000018 AB000002 BLGE      0x00000028
0x0000001C 1200000C ANDNE      R0,R0,#0x0000000C
0x00000020 00000000 ANDEQ      R0,R0,R0
0x00000024 00000000 ANDEQ      R0,R0,R0

OBJ3.ASM
2      ENTRY
3 START
4      LDR R0,=0xAB000002
5      LDR R1,=0x1200000c
6      adds R2,R0,R1
7      subs R3,R0,R1
8      mul R4,R0,R1
9 my_exit b my_exit
10     END
11
12
```

INPUT

ML	DATA
-	0XAB000002
-	0X1200000C

OUTPUT

ML	DATA
-	0Xbd00000e
-	0X98ffffff6
-	0X28000018

Objective 2: Perform Addition, Subtraction, and Multiplication of two 32-bit numbers using load/store addressing mode.

Program:

```

        AREA prg1, CODE, READONLY
        ENTRY
START
        LDR R0,=0X40000000
        LDR R1,[R0],#4
        LDR R2,[R0],#4
        ADDS R3,R1,R2
        STR R3,[R0],#4
        SUBS R4,R1,R2
        STR R4,[R0],#4
        MUL R5,R1,R2
        STR R5,[R0]
my_exit
        B my_exit
        END

```

RESULT

Memory 1	
Address:	0x40000000
0x40000000:	40 00 00 00 50 00 00 00 90 00 00 00 F0 FF FF FF 00 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001C:	00 00
0x40000038:	00 00
0x40000054:	00 00
0x40000070:	00 00

INPUT

ML	DATA
0X10100000	0X40
0X10100004	0X50

OUTPUT

ML	DATA
0X10100008	0X90
0X1010000C	0Xffffff0
0X10100010	0X1400

Objective-3: Perform the logical operations (AND, OR, XOR, and NOT) on two 32-bit numbers using load/store addressing mode

Program

```

        AREA prg1, CODE, READONLY
        ENTRY
START
        LDR R0,=0X40000000
        LDR R1,[R0],#4
        LDR R2,[R0],#4
        ANDS R3,R2,R1

```

```

STR R3,[R0],#4
ORR R4,R2,R1
STR R4,[R0],#4
EOR R5,R2,R1
STR R5,[R0],#4
MVN R6, R1
STR R6,[R0]

my_exit
B my_exit
END

```

RESULT:

Memory 1																															
Address: 0x40000000																															
0x40000000:	40	00	00	00	50	00	00	00	40	00	00	00	50	00	00	00	10	00	00	00	BF	FF	FF	FF	00	00	00	00	00	00	00
0x4000001C:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x40000038:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x40000054:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x40000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
.....	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

INPUT		OUTPUT	
ML	DATA	ML	DATA
0X10100000	0X40	0X10100008	0X40
0X10100004	0X50	0X1010000C	0X50
		0X10100010	0X10
		0X10100014	0Xffffffbf