EXPERIMENT 1

Aim:

Write program in ARM Assembly language for the below Experiments:

a. Load Value to register R0 and copy it in to other general purpose registers using move command.
b. Add two values with
   1. Immediate Addressing
   2. Direct Addressing
   3. Indirect Addressing

Tool used: Keil uVision4

Theory:

The ARM is a Reduced Instruction Set Computer (RISC) system. In ARM Processor, all instructions are 32 bits long. Most instructions execute in a single cycle. Every instruction can be conditionally executed. LDR instruction is used to Load the value in the register and MOV instruction is used to move the content of one register to another register. Here we have loaded the value of R0 using '=' keyword.

Three types of addressing are:

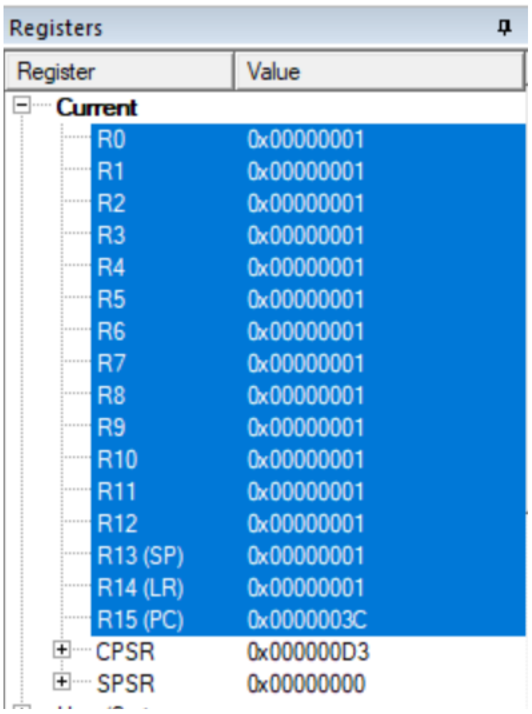
1. **Immediate Addressing**: It is used to load a constant value into register. Example: MOV R0, #10
2. **Register Direct Addressing**: It is used to move data between two registers. Example: MOV R0, R1
3. **Register Indirect Addressing**: It is used to load data from an address stored in a register. Example: LDR R0, [R1]

Code:

a) Copy content of R0 into R1 to R14

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, =0X00000001
 MOV R1, R0
 MOV R2, R0
 MOV R3, R0
 MOV R4, R0
 MOV R5, R0
 MOV R6, R0
 MOV R7, R0
 MOV R8, R0
 MOV R9, R0
 MOV R10, R0
 MOV R11, R0
 MOV R12, R0
 MOV R13, R0
 MOV R14, R0
 END
```

Register Output:



```
Registers
Register        Value
Current
  R0            0x00000001
  R1            0x00000001
  R2            0x00000001
  R3            0x00000001
  R4            0x00000001
  R5            0x00000001
  R6            0x00000001
  R7            0x00000001
  R8            0x00000001
  R9            0x00000001
  R10           0x00000001
  R11           0x00000001
  R12           0x00000001
  R13 (SP)      0x00000001
  R14 (LR)      0x00000001
  R15 (PC)      0x0000003C
  CPSR          0x000000D3
  SPSR          0x00000000
```

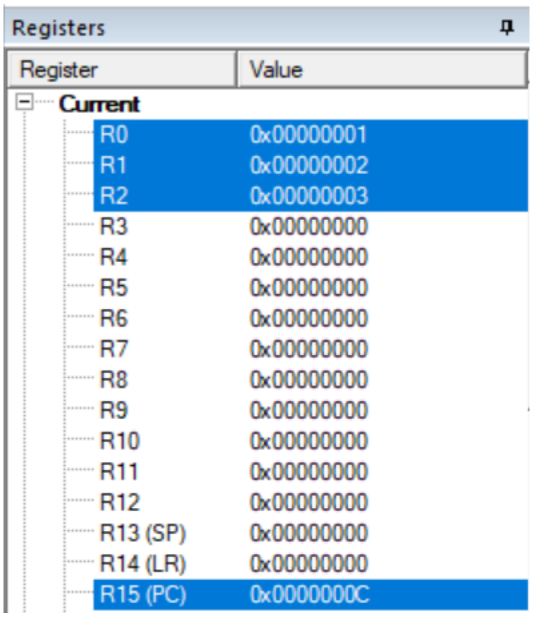
b) Modes of Addressing:

1. Immediate Addressing:

Code:

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
;immediate addressing mode
 LDR R0,=0X00000001
 LDR R1,=0X00000002
 ADD R2,R1,R0
 END
```

Register Output:



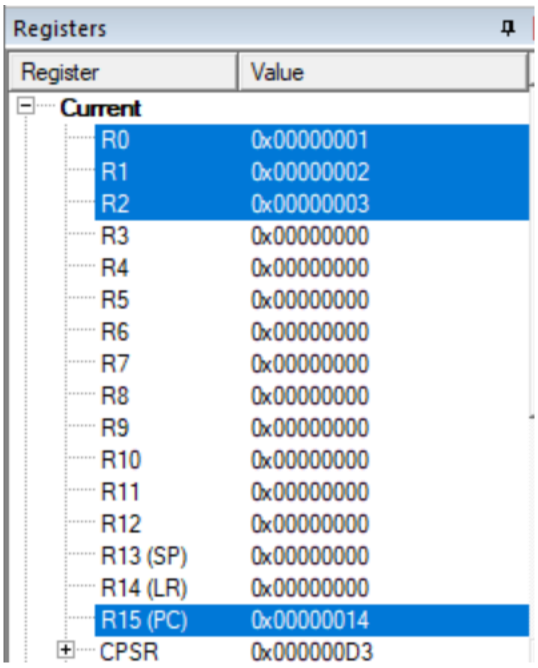
```
Registers
Register        Value
Current
  R0            0x00000001
  R1            0x00000002
  R2            0x00000003
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x0000000C
```

2. Register Direct Addressing:

Code:

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
;direct addressing mode
 LDR R0,VALUE1
 LDR R1,VALUE2
 ADD R2,R1,R0
 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000001
VALUE2 DCD &00000002
 END
```

Register Output:



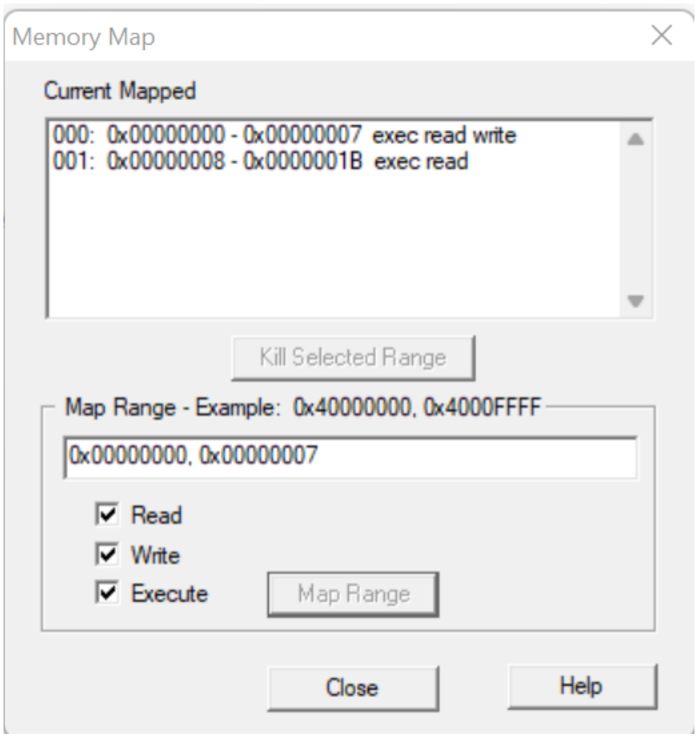
```
Registers
Register        Value
Current
  R0            0x00000001
  R1            0x00000002
  R2            0x00000003
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x00000014
  CPSR          0x000000D3
```

3. Register Indirect Addressing:

Code:

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
;indirect addressing mode
 LDR R1,VALUE1
 LDR R2,VALUE2
 LDR R3, [R1]
 LDR R4, [R2]
 ADD R5, R3, R4
 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000000
VALUE2 DCD &00000004
 END
```

Memory Map:



Current Mapped

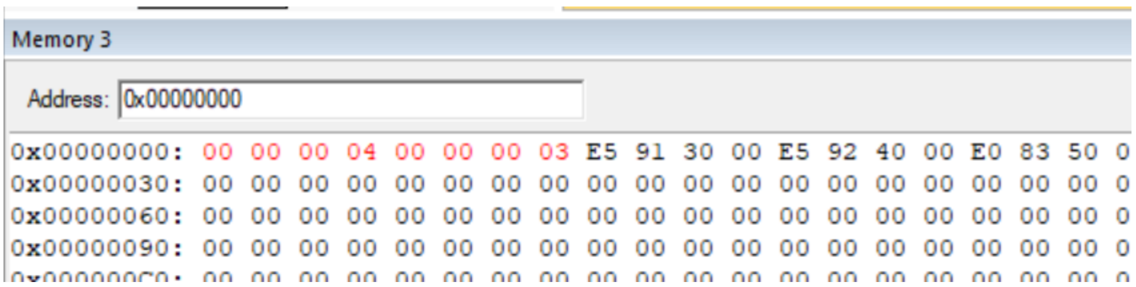
```
000:  0x00000000 - 0x00000007  exec read write
001:  0x00000008 - 0x0000001B  exec read
```

Map Range - Example: 0x40000000, 0x4000FFFF

0x00000000, 0x00000007

- [✓] Read
- [✓] Write
- [✓] Execute

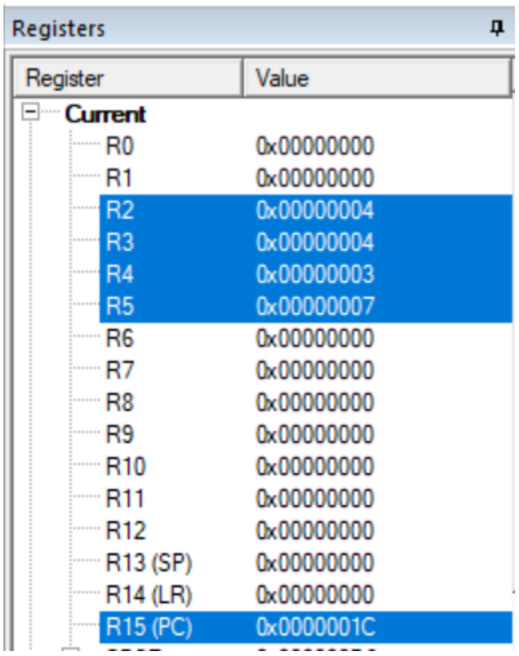
Value assignment in Memory



Address: 0x00000000

```
0x00000000: 00 00 00 04 00 00 00 03 E5 91 30 00 E5 92 40 00 E0 83 50 0
0x00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0x00000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0x00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0x000000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
```

Register Output:



```
Registers
Register        Value
Current
  R0            0x00000000
  R1            0x00000000
  R2            0x00000004
  R3            0x00000004
  R4            0x00000003
  R5            0x00000007
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x0000001C
```

Result:

All the parts of the experiments are performed successfully and their results are also verified correctly.