

MICROPROCESSOR LAB EXPERIMENT 6

GROUP - 18

Deenabandhan N ee23b021

Sai Harshith Gajendra ee23b069

Krutarth Patel ee23b137

25 September 2024

Introduction :

- In this experiment, we will observe various aspects relating to the ARM assembly language.
- We will also write a few programs in ARM assembly. We will use Keil μ Vision software to run the ARM assembly language programs
- ARM stands for **Advanced RISC Machine** which is bit more advanced than AVR.

ARM and Keil μ vision :

- ARM architectures are the most common electronic design in the world.
- It is known for its speed.
- Using an ARM architecture gives hardware designers more control over their designs and performance, as well as more control over their supply chains.
- In this experiment , we use **LPC2378** microcontroller for operations.
- The tasks that we are going to perform here will help us know the basic operations in ARM.
- By using these we will analyse the various instructions available in ARM and get familiar with those.

Instructions used :

Instruction	Usage
LDR Rx,Rd	Load the value at the memory Rd in the register Rx
ADD Rx,Rm,k	Add the values in the registers Rm and K i.e this can be a value and store it in the register Rx .
MUL Rx,Rm,K	Multiply the values in the registers Rm and K i.e this can be a value and store it in the register
STR Rm,Rx	Store the address of the register Rx in the pointer Rm
MOV Rm1,Rm2	Copy the value stored in the register Rm1 to the register Rm2
SUB Rm1,Rm2,k	Subtract the values in the registers Rm and K i.e this can be a value and store it in the register Rx .
CMP Rm1,Rm2	Compare the values of the registers Rm1 and Rm2 and raise the following flags in the status registers <ul style="list-style-type: none"> - Sign flag - Negative flag - Carry flag if the first value is smaller than the second value.
DEC Rm	Decrease the value in the register Rm by an unit.
BEQ	It is a conditional statement to check if the zero flag is raised and move to the given pointer.
SWI &11	It is used to break the program counter and end the program.

Introductory tasks

Introduction :

- This task involves knowing the basic instructions given in the table.
- First two tasks are solved below.

Code - Task-1

```
1      AREA Program, CODE, READONLY
2      ENTRY
3          MOV r0,#11
4  stop
5      B stop
6      END
```

Explanation

- AREA , CODE , READONLY are assembler directives that direct the code to begin the program and run through it.
- MOV copies the value to the register.
- # means decimal format.

Code - Task-2

```
1      AREA Program, CODE, READONLY
2      ENTRY
3          MOV r0,&FFFFFFF
4  stop
5      B stop
6      END
```

Explanation

- This code is supposed to create an error but surprisingly it does not.
- \$ means it is in hexadecimal format.
- The error is because the value given at the second argument should be of 8 bits , but it exceeds.
- This is because ARM compiler 5 recognises the error and adds the necessary instructions on its own.

Microprocessor Lab Expo

C:\Users\ielab\Documents\experiment_7\proj - uVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0xFFFFFFFF
R1	0x00000000
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)	0x00000004
CPSR	0x00000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC	0x00000004
Mode	Supervisor
States	1
Sec	0.00000025

Disassembly

Address	Instruction	Comment
0x00000000	E3E00000	MVN R0, #0x00000000
0x00000004	B stop	
0x00000008	EAFFFFFFFF	B 0x00000004
0x0000000C	00000000	ANDEQ R0, R0, R0
0x00000010	00000000	ANDEQ R0, R0, R0

tasks

```

1 AREA Program, CODE, READONLY
2 ENTRY
3   MOV R0, #0xFFFFFFFF
4 stop
5   B stop
6 END

```

Command

Running with Code Size Limit: 32K
 Load "C:\Users\ielab\Documents\experiment_7\axf"
 BS \experiment_7\task1.s

Watch 1

Name	Value	Type
<Enter expression>		

Activate Windows
Go to Settings to activate Windows.

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter

Watch 1 | Call Stack | Locals |

Real-Time Agent: Target Reset | Simulation | t1: 0.00000025 sec | L5 C1 | CAP. NUM. SCRL. OVR. R/W

Task-3

Introduction :

- In this task , we will execute the given assembly code and check the value at each registers.

Code

```
1      AREA Reset, CODE, READONLY
2      ENTRY
3          LDR r0, = 7
4          MUL r1, r0, r0
5          LDR r2, = 4
6          MUL r1, r2, r1
7          LDR r3, = 3
8          MUL r3, r0, r3
9      ADD r1, r1, r3
10     stop
11          B stop
12
13     END
```

Explanation :

- Let us analyse it line by line.

Line-1:

```
1      LDR r0, = 7
```

This creates a label to number 7 and assigns the value 7 to register **R0**

R0 = 0x00000007

Line-2:

```
1      MUL r1, r0, r0
```

This multiplies the value stored in **R0** with that of **R0** and stores it in the register **R1**.

R0 = 0x00000007

R1 = 0x00000031

Line-3:

```
1      LDR r2, = 4
```

This creates a label to number 7 and assigns the value 4 to register **R2**

R0 = 0x00000007

R1 = 0x00000031

R2 = 0x00000004

Line-4:

```
1      MUL r1, r2, r1
```

This multiplies the value stored in **R2** with that of **R1** and stores it in the register **R1**.

R0 = 0x00000007

R1 = 0x000000C4

R2 = 0x00000004

Line-5:

1 `LDR r3, = 3`

This creates a label to number 3 and assigns the value 4 to register **R3**

R0 = 0x00000007

R1 = 0x000000C4

R2 = 0x00000004

R3 = 0x00000003

Line-6:

1 `MUL r3, r0, r3`

This multiplies the value stored in **R0** with that of **R3** and stores it in the register **R3**.

R0 = 0x00000007

R1 = 0x000000C4

R2 = 0x00000004

R3 = 0x00000015

Line-7:

1 `ADD r1, r1, r3`

This adds the value stored in **R1** with that of **R3** and stores it in the register **R1**.

R0 = 0x00000007

R1 = 0x000000D9

R2 = 0x00000004

R3 = 0x00000015

Task-4

Introduction :

- In this task , we will obtain the tenth number in a Fibonacci sequence.

Code

```
1 AREA Reset, CODE, READONLY
2     ENTRY
3         LDR r0, #8 ;counter
4         LDR r1, #0 ;second last term
5         LDR r2, #1 ;last term
6     LOOP
7         MOV r3,r1 ;copying content to temporary register
8         MOV r1,r2 ;updating second last term
9         ADD r2,r3 ;calculating last term
10        SUBS r0,r0,#1 ;decrement counter
11        BNE LOOP
12    stop
13        B stop
14    END
```

Explanation :

- r0 is the counter. The program stops when r0 becomes zero.
- r1 stores the second last term of the sequence.
- r2 stores the last calculated term of the sequence.

Task-5

Introduction :

- In this task, we will divide a 32-bit binary number by a 16-bit binary number and store the quotient as well as the remainder.

Code

```

1      AREA Program, CODE, READONLY
2      ENTRY
3          LDR r0,Num1
4          LDR r1,Num2
5          MOV r2,#0
6
7      Loop
8          CMP r1,#0 ;test division by 0
9          BEQ Error1
10         CMP r0,r1 ;is the dividend less than the divisor ?
11         BMI Result ;if yes, then we are done
12         ADD r2,#1 ;add one to quotient
13         SUB r0,r1
14         B Loop
15
16     Error1
17         MOV R3, #0xFFFFFFFF ;error flag (-1)
18
19     Result
20         LDR R4, [R4] Remainder ;store the remainder and quotient
21         STR R0,[R4]
22         LDR R5, [R5] Quotient
23         STR R2,[R5]
24         SWI &11
25
26     Num1 DCD &00000009
27     Num2 DCD &00000005
28
29         ALIGN
30         AREA Data2, DATA, Readwrite
31
32     Quotient DCD 0
33     Remainder DCD 0
34
35     END

```

