MICROPROCESSOR LAB EXPERIMENT 8

GROUP - 18
Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

October 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 μVision software to run the ARM assembly language programs
- ARM stands for **Advanced RISC Machine** which is bit more advanced than AVR.

ARM and Keil μvison :

- 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 memory **Rd** into register **Rx**.
ADD Rx, Rm, k	Add values in register **Rm** and constant **k**; store in register **Rx**.
MUL Rx, Rm, K	Multiply values in register **Rm** and constant **K**; store in register **Rx**.
STR Rm, Rx	Store address of register **Rx** in pointer **Rm**.
MOV Rm1, Rm2	Copy value from register **Rm1** to register **Rm2**.
SUB Rm1, Rm2, k	Subtract value of **Rm2** or constant **k** from **Rm1**; store in **Rm1**.
CMP Rm1, Rm2	Compare values of **Rm1** and **Rm2**, setting flags if **Rm1 ¡ Rm2**: Sign, Negative, Carry.
DEC Rm	Decrease value in register **Rm** by one.
BEQ	Conditional jump if zero flag is set; branches to pointer.
SWI &11	Terminates the program by breaking the program counter.

Tasks 1 & 2

Code - Task-1

```
PINSEL10 EQU  0xE002C028
FIO2DIR  EQU  0x3FFFC040
PINSEL4  EQU  0xE002C010
FIO2PIN  EQU  0x3FFFC054
        AREA LED,CODE,READONLY
        ENTRY
        EXPORT SystemInit
        EXPORT __main
```

Explanation

- AREA , CODE , READONLY are assembler directives that direct the code to begin the program and run throgh it.
- **E**QU is used to assign the values on the right wherever the values on the left are present in the code.

Code - Task-2

```
SystemInit
        LDR R0, =PINSEL10;
        LDR R1, [R0]
        MOV R2, #0x0;
        STR R2, [R0]
```

Explanation

- Loads the value at PINSEL10 into R1.
- Loads the value at the address stored in R0.
- Moving 0x0 in R2.
- Storing the value in R2 to the address stored in R0.

Task 3 & 4

Code - Task-3

```
1       LDR R0, =PINSEL4;
2       LDR R2, =0xFFFF0000;
3       STR R2,[R0]
```

Explanation :

- Loading into R0 from the address in PINSEL4
- Loading into R2 a constant value (0xFFFF0000)
- Storing the value in R2 to the address stored in R0.

Code - Task-4

```
1       LDR R0, =FIO2DIR
2       LDR R2, =0x0000FFFF
3       STR R2, [R0]
```

Explanation :

- Loading into R0 from the address in FIO2DIR
- Loading into R2 a constant value (0x0000FFFF)
- Storing the value in R2 to the address stored in R0.

Task-5 & 6

Code - Task-5

```
__main
forever
        LDR R0, =FIO2PIN;
        MOV R2, #0xF0
        STR R2,[R0]
        b forever
        END
```

Explanation :

- Loading into R0 from the address in FIO2PIN.
- Moving #0xF0 into R2
- Storing the value in R2 to the address stored in R0.

Code - Task-6

```
__main
forever
        LDR R0, =FIO2PIN;
        MOV R2, #0xF0
        STR R2,[R0]
        LDR R3, =0x0000FFFF
loop
        SUBS R3,R3,#1
        BNE loop
        MOV R2, #0x00
        STR R2, [R0]
        LDR R3, =0x0000FFFF
loop1
        SUBS R3,R3,#1
        BNE loop1

        b forever
        END
```

Explanation :

- R3 is a counter using for adding delays
- loop and loop1 add delay