Dayananda Sagar College of Engineering
Department of Electronics and Communi Engineering
ShavigeMalleshwara Hills, Kumaraswamy Layout, Bangalore – 560 078.
(An Autonomous Institute affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)
Accredited by National Assessment and Accreditation Council (NAAC) with 'A' grade

OPEN ENDED EXPRIMENT

MICRO CONTROLLER & EMBEDDED SYSTEM

Program: BE | Branch: ECE

Course: Micro controller and Embedded System | Semester: IV

Course Code: 22ECL45 | Date:

A Report On:

Fundamentals of ARM and its application.

Write an ALP to implement ARM core processor.

Submitted by:

Sl no	USN	NAME	MARKS
1.	1DS22EC054	Chaitra K Gouda	

Faculty In-charge:

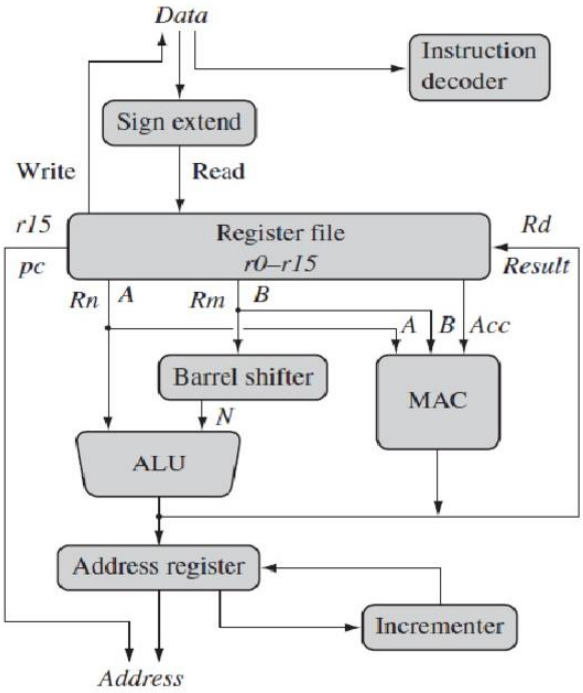
Dr. Suma M R

Assistant Professor, ECE Dept., DSCE, Bengaluru

INTRODUCTION-

The RISC architecture, a widely used computer arrangement, is the foundation of the ARM, which is an acronym for Advanced RISC Machine. It is a 32-bit module that was developed in 1987 by Acron. This board is made by ST Microelectronics and Motorola, two separate MCU manufacturers. This module is divided into various categories, such as ARMv1 and ARMv2, and each category offers a unique set of features. An ARM core is functional units connected by data buses. where, the arrows represent the flow of data, the lines represent the buses, and the boxes represent either an operation unit or a storage area. The figure shows not only the flow of data but also the abstract components that make up an ARM core.

An ARM core is functional units connected by data buses, as shown in the figure, where, the arrows represent the flow of data, the lines represent the buses, and the boxes represent either an operation unit or a storage area. The figure shows not only the flow of data but also the abstract components that make up an ARM core.



ARM Core Data Flow Model

1. Data enters the processor core through the Data bus. The ARM processor uses a load store architecture. This means it has two instruction types for transferring data in and out of the processor: load instructions copy data from memory to registers in the core, and conversely the store instructions copy data from registers to memory.
2. The instruction decoder translates instructions before they are executed. Each instruction executed belongs to a particular instruction set.
3. Data items are placed in the register file—a storage bank made up of 32-bit registers.
4. ARM instructions typically have two source registers, Rn and Rm, and a single result or destination register, Rd. Source operands are read from the register file using the internal buses A and B, respectively.
5. The ALU (arithmetic logic unit) or MAC (multiply-accumulate unit) takes the register values Rn and Rm from the A and B buses and computes a result. Data processing instructions write the result in Rd directly to the register file. Load and store instructions use the ALU to generate an address to be held in the address register and broadcast on the address bus.
6. Rm alternatively can be pre-processed in the barrel shifter before it enters the ALU. Together the barrel shifter and ALU can calculate a wide range of expressions and addresses. After passing through the functional units, the result in Rd is written back to the register file using the result bus.
7. For load and store instructions, the incremental updates the address register before the core reads or writes the next register value from or to the next sequential memory location. The processor continues executing instructions until an exception or interrupt changes the normal execution flow.

APPLICATION OF ARM-

1. It is a reduced instruction set computing Controller

- 32-bit high performance central processing unit
- 3-stage pipeline and compact one

2. It has THUMB-2 technology

- Merges optimally with 16/32 bit instructions
- High performance

3. It supports tools and RTOS and its core Sight debug and trace

- JTAG or 2-pin serial wire debugs connection
- Support for multiple processors

4. Low power Modes

- It supports sleep modes
- Control the software package
- Multiple power domains

5. Nested vectored interrupt controller (NVIC)

- Low latency, low noise interrupts response
- No need for assembly programming

TOOLS-

1. Keil (MDK-ARM) – This is the software used for programming in ARM-based Microcontrollers.

2. ARM Cortex M4 microcontroller (STM32F407xx) – This is the hardware that we will be programming.

Steps followed-

1. Connect PC to the microcontroller using a Type A to Type B USB cable. This connection enables the Keil software to communicate with the microcontroller for various functionality.

2. See the inbuilt LED on STM Microcontroller which will glow/blink with the delay according to the program written by the user.

3. Program and configure the STM32F407xx microcontroller with the help of Keil Software.

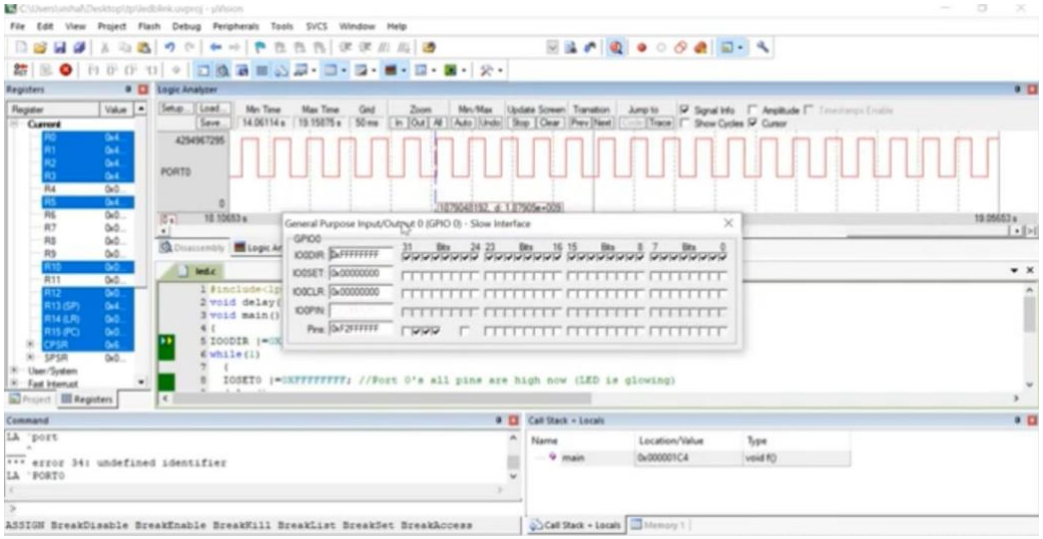
4. Flash the code to the microcontroller after compiling the code successfully and doing necessary configuration settings.

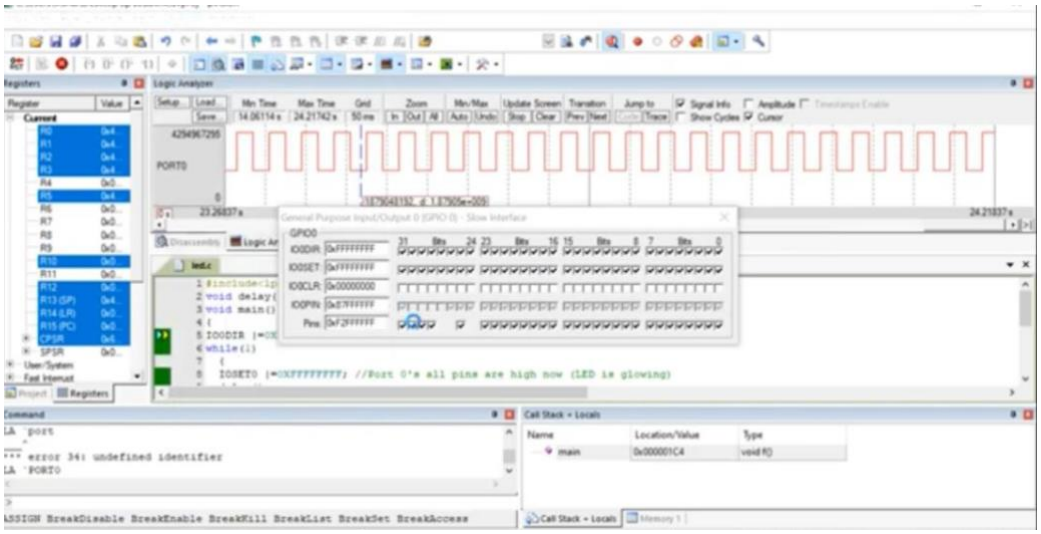
5. The inbuilt LED on the microcontroller should start blinking according to the program flashed into it. This shows that the setup is good and working correctly.

SOFTWARE-

```
#include "stm32f4xx.h"

void delay (int n);

int main(void)

{

RCC->AHB1ENR |=8;                    //enable clock to GPIOD

GPIOD->MODER |=0X55000000;  //set pin to output mode

 while(1)

{

GPIOD->ODR ^=0X0000F000;       // turn on all LED's

delay(300);

 }

}

void delay(int n)

{

 int i,j;

for(j=0;j<n;j++)

 for(i=0;i<3195;i++);                    //16MHz System Clock

}
```

OUTPUT-





The output seen above is for zero timer and random delay. It can observed in the square wave and the blinking of the LED on the display.

CONCLUSION- Output is obtained for Blink LED in Arm Cortex Microcontroller using Keil software and is verified using the square wave.