Paper ID #20286

Embedded Systems and Internet of Things (IoTs) - Challenges in Teaching the ARM Controller in the Classroom

Prof. Dhananjay V. Gadre, Netaji Subhas Institute of Technology

Dhananjay V. Gadre (New Delhi, India) completed his MSc (electronic science) from the University of Delhi and M.Engr (computer engineering) from the University of Idaho, USA. In his professional career of more than 27 years, he has taught at the SGTB Khalsa College, University of Delhi, worked as a scientific officer at the Inter University Centre for Astronomy and Astrophysics (IUCAA), Pune, and since 2001, has been with the Electronics and Communication Engineering Division, Netaji Subhas Institute of Technology (NSIT), New Delhi, currently as an associate professor. He directs two open access laboratories at NSIT, namely Centre for Electronics Design and Technology (CEDT) and TI Centre for Embedded Product Design (TI-CEPD). Professor Gadre is the author of several professional articles and five books. One of his books has been translated into Chinese and another one into Greek. His recent book ”TinyAVR Microcontroller Projects for the Evil Genius”, published by McGraw Hill International consists of more than 30 hands-on projects and has been translated into Chinese and Russian. He is a licensed radio amateur with a call sign VU2NOX and hopes to design and build an amateur radio satellite in the near future.

Dr. Ramesh S. Gaonkar, SUNY-PCC and IITGN

Ramesh Gaonkar, Ph.D., Professor Emeritus, SUNY OCC, (Syracuse, NY) was a professor in Electrical Technology, and presently, he is teaching as a Visiting Professor at Indian Institute of Technology (IIT), Gandhinagar, India. He teaches Embedded Systems at IIT either for a semester or a half-semester in an academic year. He has authored several books in the Microprocessor/Microcontroller area. He is a recipient of Excellence in Teaching awards from ASEE St. Lawrence Section and State University of New York (SUNY).

Ms. Sneha N. Ved, Indian Institute of Technology, Gandhinagar

Sneha N Ved completed her Undergraduate studies in Computer Science and Engineering from Nirma University Ahmedabad in 2011 followed by her Master’s degree from Indian Institute of Technology Madras in 2013. Sneha is currently working as a teaching assistant while pursuing her Ph.D. from Indian Institute of Technology Gandhinagar in Electrical Engineering. Sneha is an Intel India Ph.D. Fellow and her research interests include Computer Architecture and Embedded System Design.

Mr. Nikhilesh Prasannakumar, Netaji Subhas Institute of Technology

Nikhilesh Prasannakumar is an electronics and do-it-yourself enthusiast who completed his Masters and Undergraduate studies in Electrical and Electronics Engineering from National Institute of Technology Trichy (NITT) and National Institute of Technology Calicut (NITC) respectively. He is currently working as a Teaching cum Research Fellow at the Instrumentation and Control Engineering Division at Netaji Subhas Institute of Technology (NSIT), New Delhi while pursuing his Ph.D. Since 2014, he has been associated with the Centre for Electronics Design and Technology (CEDT), NSIT pursuing various projects in the field of Embedded Systems. While at NSIT he has mentored several student projects and has contributed to the design and development of several educational kits and pedagogy material based on a variety of microcontroller platforms. His research interests include embedded systems, intelligent control and internet of things.

Embedded Systems and Internet of Things (IoTs) and Challenges in Teaching the ARM Controller in the Classroom

Abstract

This presentation is concerned with issues related to teaching Embedded Systems and Internet of Things (IoTs), the frontier topics in engineering and technology curricula. ARM is the leading microcontroller used in designing Embedded Systems (such as smart phones, digital cameras, and smart appliances) and IoTs. Globally, ARM is the most widely used instruction set architecture in terms of quantity produced; over 50 billion ARM processors have been produced as of 2014, of which 10 billion were produced in 2013. It is the leading microcontroller in terms of the market share. Therefore, it is essential that our students should be familiar with applications of ARM in their undergraduate curricula. The presentation focuses on sharing teaching difficulties in ARM controller. ARM is a high performance, low power 32-bit microcontroller. To minimize power consumption ARM has used many techniques in its hardware design that increases the complexity in writing software. ARM has a very complicated architecture, and its instruction set includes various options for execution. In addition, they are many difficulties in teaching the ARM in the classroom setting because ARM Company licenses its core to many manufacturers with freedom to implement various features at their discretion.

Embedded Systems were used to be primarily stand-alone systems until the advent of IoT systems. Now Embedded Systems are getting smarter and connected with the internet that raises issues of security. This presentation focuses on how to simplify the teaching in the classroom, various options, and choices between assembly, C, and ARM's mbed platform.

Embedded Systems

Electrical Engineering and Electrical Engineering Technology curricula include a sequence of two or three courses that includes Introduction to Digital Logic Fundamentals followed by a Microprocessor/Microcontroller, and Embedded Systems. Some programs skip the second level course and go to the third level course in Embedded Systems. Many of these higher level courses have a lab or project component and include a microcontroller. In the 1990s and early 2000, these courses were based on an 8-bit microcontroller such as Intel 8051, PIC18, or AVR.

During this period, the following three technological changes made a significant impact on the field of Embedded System. 1) The embedded devices such as cell phone, smart-phone, and tablets became household devices, and these are battery-operated devices. Thus the issue of power consumption became very critical. 2) The embedded systems (such as a digital camera) started becoming more complex and 8-bit microcontrollers were falling short on the demands of these systems, 3) Because of the advancement in the fabrication technology, the price difference in fabricating 32-bit vs. 8-bit chips became relatively insignificant. The ARM controller met the challenges of these technological changes and became the leading controller in embedded systems.

8-bit Microprocessor to 32-bit Microcontroller

Historically, engineering programs at the undergraduate level have included coverage of a microprocessor related course in their syllabus, typically an 8-bit microprocessor such as the 8085 (or 6800). The chip manufacturer (Intel or Motorola) helped effective delivery of such courses by creating and sharing information related to their microprocessor development kit (MCS85, MEK6800), which served to provide practical hands-on experience to the students. The stability and professional acceptance of a microprocessor family often decided the popularity of their coverage in educational courses. Educational institutes were divided between Intel and Motorola camps.

Introductory 8-bit microprocessor courses were often supplemented with 16/32-bit 8086/68000 elective courses often titled 'Advanced Microprocessors'. With the advent of 8051, microcontroller based design courses also became popular. The salient features of these courses were that they taught professionally relevant subject as these microprocessors and microcontrollers were still used in the industry. No significant investment in laboratory equipment and/or teacher training to offer these courses was needed. However, in the middle of the 90s when newer microcontroller architectures such as the PIC and AVR challenged the supremacy of the 8051 (although it is also a surprising fact that 8085/8051 are still being taught in many undergraduate courses around the world). From the late 90s to middle of the first decade of the new century, a vast majority of educational courses aligned themselves to teaching 8085 as the introductory microprocessor course followed by AVR or PIC. The recurring theme observed in the dispensation of these courses was the relative architectural stability, continued professional popularity as well as 'good return on investments' in terms of laboratory equipment and teacher training.

Since the middle of the last decade when the Cortex-M ARM architecture became available, a debate on the suitability as well as the viability of running theory and laboratory course around the ARM architecture has been raging. Given that ARM architectures of various capabilities rule the roost in professional design, the suitability aspect has been settled. However, there are doubts on the viability aspects, and these stem from the fact that ARM does not manufacture their own silicon and instead one has to depend on traditional chip manufacturers for a learning platform based on a suitable ARM SoC (System on Chip). The chip vendor on the other hand can only offer a certain SoC based on the professional acceptance and popularity, which leads to the SoC being marked as NRND (Not Recommended for New Design) if the market seems to take an unfavorable turn. This, in turn, leads to difficulties in educational programs in sustaining the laboratory learning material as well as relearning on the part of the teaching faculty.

ARM Controller

The ARM is a 32-bit controller based on the RISC architecture with 'low power consumption' as one of the salient features. In addition, ARM has included many features in designing its buses and the instruction set to improve performance and minimize power consumption. Therefore, ARM-based controllers have become leading controllers in embedded systems.

Because of its leading place in the industry, we have started introducing ARM in our engineering curricula either in our projects or in a course. Although ARM architecture is complex, and we are making an assumption that if our students are exposed to ARM controller, they will be able to handle and incorporate any other microcontroller in designing an embedded system on their job. Now to teach the ARM controller in the classroom, we need to examine the following issues related to teaching:

- Searching for ARM documentation
- Handling complex architecture
- Bus architecture
- Initializing general purpose I/O Ports
- Managing interrupts
- Compatibility in the instruction sets and various features

Searching for ARM Documentation

ARM is a licensing company. They design the basic core of the controller, assign a block of address ranges to memory and I/Os, and leave all other architectural options to the silicon manufacturers of the chips. As a consequence, chips from different manufacturers of the same architecture differ significantly in memory address ranges, I/O addresses, and features of the peripheral modules. As an example, the chip based on Cortex-M4 from TI and NXP differ significantly regarding memory and I/O addresses and available peripherals. Therefore, to perform an experiment in the lab, faculty/students need to search for websites, data manuals, and reference manuals from three different sources: the silicon chip manufacturer, ARM, and simulation software vendor such as Keil.

Handling Complex Architecture

The architecture of a microcontroller includes the description of various hardware components, their relationship with each other, and their communication links (buses). In an 8-bit microcontroller such as Intel 8051, the architecture includes three major components: processor, registers, and three buses: address, data, and control. On the other hand, ARM architecture includes many more additional features such as expanded buses, interrupt controller, and debug unit (Figure 1). To teach an 8-bit microcontroller, it is a conceptual quantum jump from discrete gates and flip-flops. It is even more difficult to teach with additional architectural components such as the interrupt controller, and the debug unit.

Since the early 1990s, the architecture of ARM controllers has gone through many changes – from ARMv3 to ARMv8 to meet the demands of changing technology environment. Each architecture design has its own family of processors. The recent architectural designs are known as ARMv7, and the Cortex-M processor series (using ARMv6 and ARMv7 architectures) is designed for embedded systems where low power, minimum silicone area, and low cost are very important features.

Bus Architecture

To improve the processor performance, ARM designed a new system of buses called AMBA (Advanced Microcontroller Bus Architecture) that separates the buses as AHB (Advanced High-Performance Bus) and APB (Advanced Peripheral Bus) that are connected by a bridge (See Figure 1). The AHB is a high bandwidth – high-speed bus that is used to transfer data between the processor and on-chip memory. The APB is used with slow external peripherals such as LEDs and keyboards.

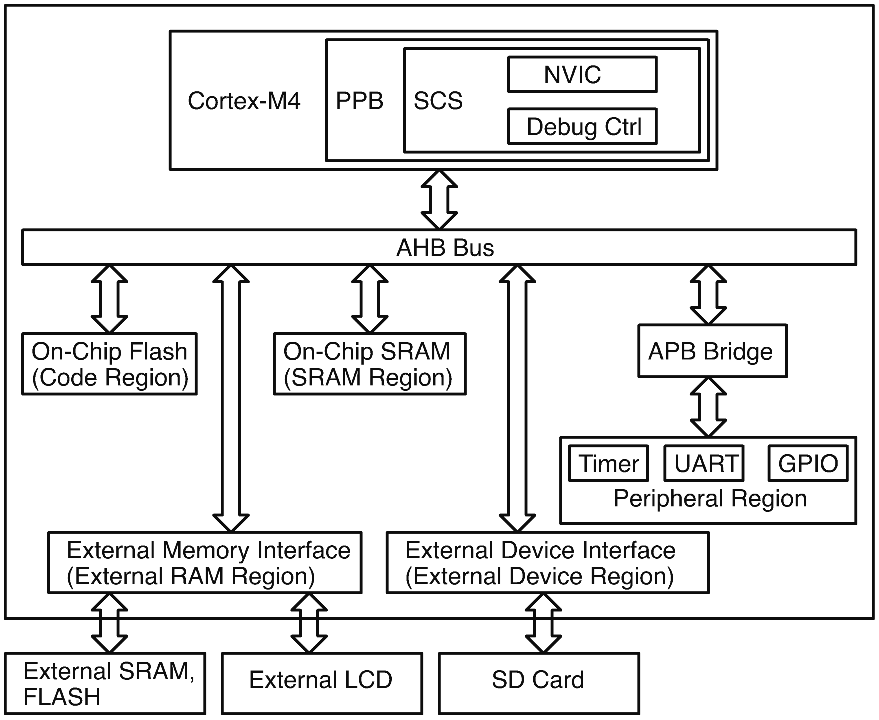
Cortex-M4
PPB
SCS
NVIC
Debug Ctrl
AHB Bus
On-Chip Flash (Code Region)
On-Chip SRAM (SRAM Region)
APB Bridge
Timer
UART
GPIO
Peripheral Region
External Memory Interface (External RAM Region)
External Device Interface (External Device Region)
External SRAM, FLASH
External LCD
SD Card

Figure 1: Architecture of ARM M-Cortex Controller

In the design of the ARM controller, there are many enhancements to the basic RISC architecture that provide a good balance of high performance, small code size, low power consumption and small silicon area. However, these enhancements add more challenges in teaching the architecture of the ARM controller.

Initializing General Purpose I/O (GPIO) Ports

The ARM architecture includes eight I/O Ports labeled as A, B, C, D, E, F, G and H with each port having possible 32 pins (B31-B0). However, it is left to the chip manufacturer to design the number of ports, the number of pins per port, and which pins to use, based on its intended applications and design. The ARM architecture has assigned the address range of 5 MB from 4000_0000H to 5FFF_FFFFH to GPIO ports. However, it is left to the manufacturer to assign specific addresses to GPIO Ports A to H. If we select Cortex-M4 for our classroom project, we need to refer to the silicon manufacturer's data-sheets for its specifications.

In most 8-bit microcontrollers, the initialization of an I/O port requires only two instructions: one for the data direction register and the other for the I/O register either to read or write data. However, to initialize I/O ports in ARM controller requires multiple (five to seven) instructions because ARM has included many features in its design to minimize the power consumption and improve the performance as follows:

1. The system clock to all I/O pins is turned off to minimize unnecessary power consumption. Therefore, the clock should be turned on before we write any other I/O instructions.
2. The I/O pins are multiplexed, meaning each pin can be set up to perform different functions; therefore, the GPIO function for a given pin must be enabled.

3. The data direction register enables the input or output function.
4. If a pin is set up for the output function, there are registers to set, reset, or toggle the pin.
5. Each I/O pin can be internally connected to a pull-up or pull down resister, or the pin can be disconnected from the resistor.
6. The drive strength of a pin- high or low – can be set up.
7. The slew rate – high or low – can be set up.

By examining the above requirements and the options available to initialize an I/O port, it is evident that even GPIO programming on an ARM microcontroller is not an easy task.

Managing Interrupts

The interrupt process in 8-bit microcontrollers generally limited to a few interrupts. In the ARM controller design, the process is set up to handle 240 interrupts and multiple exceptions. Furthermore, the interrupt process varies among different architectures. Specifically, in Cortex-M processors, it is handled by a hardware module called Nested Interrupt Controller (shown in Figure 2). The interrupts are classified in two groups: interrupts initiated by external devices and exceptions initiated by the processor such as illegal instruction or illegal process such as divide by zero.

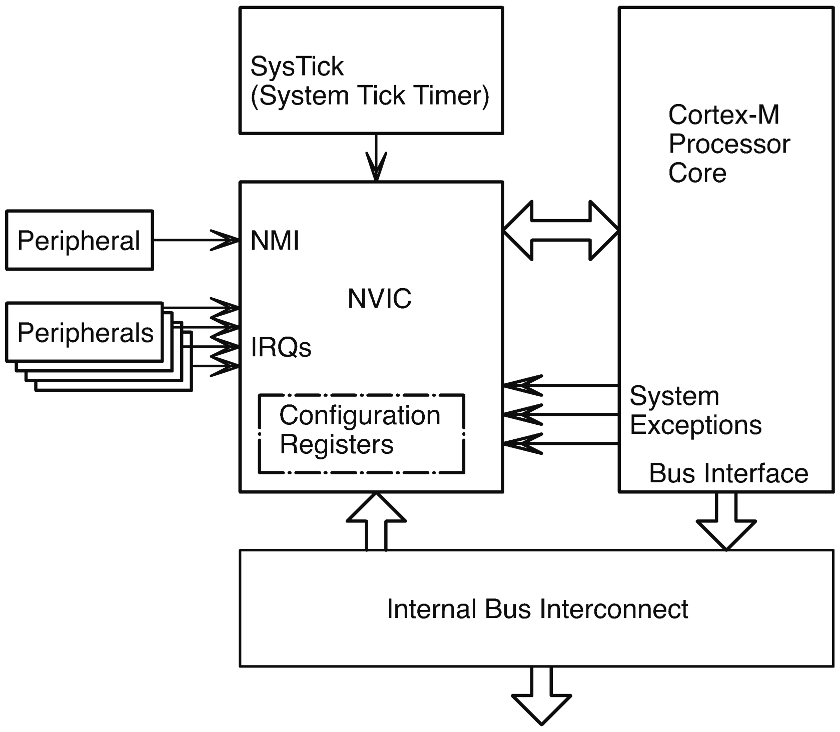

Figure 2 Nested Vector Interrupt Controller (NVIC)

Nested Interrupt Controller is a hardware module that is coupled tightly with the processor to minimize the time spent for handling interrupts. It receives inputs from external peripheral interrupts and NMI (non-maskable interrupt) as well as from the exceptions initiated by the processor and the System Tick Timer. The NVIC is capable of supporting up to 240 interrupts, each with up to 256 levels of priority that can be changed during the execution of a program. Also, the processor and NVIC can be put into a very low-power sleep mode, leaving the Wake Up Controller (WIC) to identify and prioritize interrupts.

Interrupt Vector Table: This table includes 256 interrupts (and exceptions) and their assigned memory locations called Interrupt Vector locations. Each interrupt is assigned four memory locations to store the 32-bit starting address of the associated interrupt service routine (ISR). The Interrupt Vector Table is divided into two groups. The first group -- includes Reset, NMI, reserved locations, and exceptions, and their addresses are assigned by ARM. The second group begins at the interrupt type 16 labeled as INT0 to INT240 is assigned to external peripherals, and chip manufacturers are free to choose peripherals and their vector addresses. Even among the Cortex-M processors, the available interrupts vary.

Instruction Set

For ARM Cortex M based microcontrollers, variety exists in two different forms – in terms of ARM Architecture and Instruction Set support as well as manufacturer specific variations in terms of on-chip peripherals and feature support (Table 1). In the early years of ARM development, all instructions were 32-bit wide, and they required significant memory space. To save memory space, ARM came up with 16-bit instruction set called Thumb1 which was operationally a compressed version of the original instruction set. The architecture was modified to switch between two modes. This was followed by Thumb2 instruction set to improve performance.

ARM Core	Cortex M0	Cortex M0+	Cortex M1	Cortex M3	Cortex M4
Thumb-1 Instructions	Most	Most	Most	Entire	Entire
Thumb-2 Instructions	Some	Some	Some	Entire	Entire
Multiply Instructions	32-bit result	32-bit result	32-bit result	32/64-bit result	32/64-bit result
Divide Instructions	No	No	No	Yes	Yes
DSP Instructions	No	No	No	No	Yes
Floating-point Instructions	No	No	No	No	Optional: SP
Instruction pipeline stages	3	2	3	3	3
Architecture	Von Neumann	Von Neumann	Von Neumann	Harvard	Harvard
ARM architecture	ARMv6-M	ARMv6-M	ARMv6-M	ARMv7-M	ARMv7E-M

Table 1: Instruction Variations in Cortex-M Chips

In terms of ARM Architecture, the Cortex-M0/M0+/M1 implement the ARMv6-M, the Cortex-M3 implements the ARMv7-M, and the Cortex-M4 implements the ARMv7E-M. The architectures are binary instruction upward compatible from ARMv6-M to ARMv7-M to ARMv7E-M, i.e. binary instructions available for the Cortex-M0/M0+/M1 can execute without modification on the Cortex-M3/M4. Only Thumb-1 and Thumb-2 instruction sets are supported in Cortex-M architectures. All Cortex-M cores implement a common subset of instructions that consists of most Thumb-1, some Thumb-2, including a 32-bit result multiply.

The Cortex-M0/M0+/M1 include Thumb-1 instructions, except new instructions (CBZ, CBNZ, IT) which were added in ARMv7-M architecture along with a minor subset of Thumb-2 instructions (BL, DMB, DSB, ISB, MRS, MSR). The Cortex-M3 adds the three remaining Thumb-1 instructions, all Thumb-2 instructions, hardware integer divide, and saturation arithmetic instructions. The Cortex-M4 adds DSP instructions and an optional single-precision floating-point unit (VFPv4-SP).

From the teaching perspective, the instruction set became complex in terms of its compatibility, operations, and execution options.

Emergence of IoT

In recent years, embedded systems are being designed as smart devices meaning they are capable of receiving data, making decisions, and communicating with other devices through the Internet. An embedded system with the Internet connectivity is known as an IoT device, in general. Any device connected to the Internet must have an IP address, and with the IP address, the designer must confront a host of security issues such as data security, firmware security, and the system operations security.

In IoT systems security is a very critical issue. To resolve security issues, ARM has two solutions: hardware and software. For the hardware approach, ARM has two new controllers – M23 and M33 that provide trust zone security for data, firmware, and operations. Some memory segments and interrupts can be defined as secure segments. For the software approach, the encryption is the technique used to write software.

Teaching Issues In the Classroom

As mentioned earlier, the ARM microcontroller has a very dominant presence in Embedded Systems, which will grow exponentially because of the expansion of IoT. Therefore, it is an extremely important topic to teach in undergraduate curriculum for the preparation of our students to be industry-ready. After examining various aspects of the ARM controller, let us look at the issues we need to focus in the classroom in a given semester of 16 weeks. Most of our programs have an objective that our students should be prepared to analyze and troubleshoot the existing embedded system, and be able to design an embedded system to meet given specifications. Traditionally, the course is generally divided into three parts: architecture, programming in assembly and C languages, and I/O communication, and this is followed by the integration of these topics in designing a project.

However, pursuit of these topics does not leave much time and scope to cover enough I/O programming aspects. As discussed earlier, the architecture of the ARM is much more complex than that of 8-bit microcontrollers. Similarly, assembly language programming as well as C programming is time consuming and demanding when memory addresses, I/O peripherals and their addresses are manufacturer specific.

The solution provided by the chip manufacturers and also by ARM is the use of application program interface (**API**), which is a set of routines, protocols, and tools for building software applications. An **API** specifies the necessary information such as register definitions to write assembly and C programs. Manufacturers generally provide their own device specific driver libraries in an easy to use API format in order to ease code development for their platform, but these libraries tend to be manufacturer specific. However, for some ARM chips, an open source GNU C/C++ compiler with appropriate configuration can use these driver libraries. For a traditional course that teaches CPU architecture, the focus can be on developing application code and leave the handling of the on-chip peripherals to the driver libraries provided by the manufacturer.

This will require a mix of C/C++ function calls (for the driver libraries) as well as assembly language (for the application code) in a typical program. This can be easily achieved using the inline assembly functionality of almost all C/C++ compilers for the ARM platform. This allows the course to have more focus on the appropriate use of the CPU instruction set to perform a given operation and does not require any significant focus on the specifics of the operation of the various on-chip peripherals.

Another approach is to design a course with the system approach using an ARM microcontroller, and the focus can be shifted on to the capabilities of the various peripherals available on the chip. In this case, instead of using assembly language for the application code, the application development can be made simpler by using the C/C++ compiler for the entire program, as the focus is on the system level design and not the processor and its architecture. With the systems approach, it may be possible to include the IoT extension to Embedded Systems. But because of security issues of the Internet communication, the system must include software approaches of encryption to write programs. This type of approach demands software expertise, and such a topic may belong in a Computer Science course.

One of the major drawbacks in both of these approaches suggested above is that students will learn embedded systems without much understanding of hardware and the communication process between the processor, memory, and I/O peripherals.

Summary

ARM is the leading microcontroller used in designing Embedded Systems. Therefore, it is an extremely important processor to teach in Electrical Engineering and Technology undergraduate curriculum to prepare our students to be industry-ready. It is a high performance, low power 32-bit microcontroller. To minimize power consumption ARM has used many techniques in its hardware design that increase the complexity in writing software. ARM has a very complicated architecture, and its instruction set includes various options for execution. In addition, they are many difficulties in teaching the ARM in the classroom setting because ARM Company licensees its core to many manufacturers with freedom to implement various features at their discretion. Therefore, two different approaches are suggested in designing the embedded system course: one to include extensive use of API provided by manufacturers and the second approach is to teach the course with the systems approach without focusing on hardware, instruction set, and assembly language.

Embedded Systems were used to be primarily stand-alone systems until the advent of IoT systems. Now they are connected with the Internet that raises issues of security. This presentation focuses on how to simplify the teaching in the classroom, and choices between assembly, C, and ARM’s mbed platform.

References

1. Dhananjay V. Gadre, Rohit Dureja and Shanjit S. Jajmann. "*Getting Started with Stellaris ARM Cortex-M Embedded Processors: A Lab Manual for Stellaris Guru Evaluation Kit*", Universities Press, India, 2013
2. William Hohl and Christopher Hinds. *“ARM Assembly Language: Fundamentals and Techniques",* Second Edition, CRC Press, 2014
3. Fei Hu. *“Security and Privacy in Internet of Things (IoTs): Models, Algorithms, and Implementations*”, CRC Press, 2016
4. Trevor Martin. “*The Designer's Guide to the Cortex-M Processor Family: A Tutorial Approach" ,* Newnes, 2013
5. Muhammad Ali Mazidi, & Shujen Chen: *“TI ARM Peripherals, Programming, and Interfacing”*, Kindle Edition, Amazon, 2014
6. Mohammad Tahir and Kashif Javed. “*ARM Microprocessor Systems: Cortex-M Architecture, Programming, and Interfacing”*, CRC Press, 2017
7. Jonathan Valvano. *“Embedded Systems: Introduction to ARM Cortex-Microcontrollers*”, 5th Ed. Jonathan V. Valvano, 2012
8. Joseph Yiu. *"The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors",* Newnes, 2013
9. Yifeng Zhu. *“Embedded Systems With Cortex-M Microcontrollers in Assembly Language and C”,* E-Man Press LLC; 2nd edition, 2015