Government of Karnataka

Department of Technical Education

Government Polytechnic RAICHUR

DEPARTEMENT OF ELECTRONICS AND COMMUNICATIONS

ARM Unit 1 -- Lecture Notes

For

5th Sem: ARM Controller

(Course code: 15EC52T)

Prepared By:

VITHALANI PARESH K.

Senior Lecturer, E&C DEPT.

GOVT. POLYTECHNIC Raichur (117)

Dist: Raichur-584101

Email id: pareshkv2015@gmail.com

Course Content

Unit1: ARM Embedded Systems and ARM Processor Fundamentals. Duration:10 Hrs.

The RISC design philosophy, ARM design philosophy, embedded system hardware- AMBA Bus protocol, embedded system software- applications. ARM core data flow model, Registers, CPSR-Processor modes, Banked registers. Pipeline - Characteristics.

Unit 2: ARM Instruction Set. Duration: 10 Hrs.

Fundamentals of ARM instructions, Barrel shifter, Classification and explanation of instructions with examples- Data processing, Branch, Load-store, SWI and Program Status Register instruction.

Unit 3: Introduction to THUMB and ARM Programming Duration: 08 Hrs.

Introduction to THUMB, Differences between ARM and THUMB, Register usage in Thumb, ARM Thumb Interworking. General Structure of ARM assembly module, Assembler directives- AREA, ENTRY, END, SPACE, DCD, DCB, DCW, DCI, DCQ, EQU, EXPORT, ALIGN, CODE16, CODE32, DATA. Simple ALP programs on Arithmetic & logical operations, Factorial, string operation, sorting, searching, and Scan.

Unit 4: Exception and Interrupt handling schemes Duration: 07 Hrs.

Exception handling- ARM processor exceptions and modes, vector table, exception priorities, link register offsets. Interrupts- assigning interrupts, interrupt latency, IRQ and FIQ exceptions with example- code for enabling and disabling IRQ and FIQ exceptions, Comparison between exception and interrupts. Interrupt handling schemes- nested interrupt handler, non-nested interrupt handler. Basic interrupt stack design.

Unit5: LPC2148 ARM CPU Duration: 06 Hrs.

LPC 2148 - Salient features, applications, block diagram, memory mapping. Functional features of Interrupt controller, RTC, USB, UART, I2C, SPI, SSP controllers, watch dog timers and other system control units.

Unit6: LPC 2148 – Peripherals Duration: 11 Hrs.

Pin Connect Block- Features, Register description with example. GPIO-Features, Applications, Pin description, Register description with examples PLL-Features, block diagram, bit structure of PLLCON, PLLCFG, & PLLSTAT, and PLLFEED. PLL frequency Calculation- procedure for determining PLL settings, examples for PLL Configuration Timers-Features, applications, Architecture of timer module, register description, Simple C programs for application using -GPIO, PLL, Timer.

Course content and pattern of marks for SEE

Unit No	Unit Name		Question to be set for SEE						Marks Wieghtage	Weightage(%)
		Hour	R	U	A	AN	E	C		
1	ARM Embedded Systems and ARM Processor Fundamentals.	10	5	10	5	5	0	0	25	16
2	ARM Instruction Set	10		5	5	5	5	5	25	20
3	Introduction to THUMB and ARM Programming.	8	5	5	5	0	0	5	20	14
4	Exception and Interrupt handling schemes.	6	5	5	5	5	0	0	20	13
5	LPC2148 ARM CPU	6	5	5	10	0	0	0	20	12
6	LPC 2148 – Peripherals	12	5	10	15	0	5	0	35	25
	Total	**52**	**25**	**40**	**45**	**15**	**10**	**10**	**145**	**100**

Course Assessment and Evaluation Scheme

Assessment Method	What		To Whom	Assessment mode	Max. Marks	Evidence Collected	Course Outcomes
Direct Assessment	CIE	IA	Students	3 Tests	20	Blue Books	1 to 6
	SEE	End Exam		Activity	5	Activity Sheets	1 to 6
				End of the course	100	Answer Scripts at BTE	1 to 6
				Total	**125**		
Indirect Assessment	Student feedback on course		Students	Middle of the course	Nil	Feedback Forms	1 to 3 & Delivery of course
	End of course survey			End of the course	Nil	Question-naires	1 to 6, Effectiveness of delivery instructions & assessment methods.

Unit: 1

ARM Embedded Systems and ARM Processor Fundamentals.

Prerequisites:

1. Knowledge of digital electronics.
2. Knowledge of microcontroller architecture and applications.

Learning Objectives:

1. Understand the features of embedded systems, architecture of ARM7 and applications.

Unit Outcomes:

Upon successful completion of the unit student is able to

- Understand the importance of RISC system design.
- State the special features of ARM processor design.
- Understand the AMBA protocol.
- Explain the role of software components in an embedded system.
- Understand the ATM core data flow model.
- Explain the register file of ARM core.
- Understand the operating modes of ARM core.
- Understand the role of banked registers.
- Understand the process of pipelining.

Breakage of Unit and Lecture schedule:

Si N.	Topic	Date	Duration of Lecture
1	The RISC design philosophy		1 Hour
2	ARM design philosophy		1 Hour
3	Embedded system hardware – AMBA protocol		1 Hour
4	Embedded system software– AMBA core applications		1 Hour
5	ARM core data flow model		1 Hour
6	ARM Registers		1 Hour
7	CPSR register – Processor modes		1 Hour
8	Banked registers		1 Hour
9	Concept of Pipeline		1 Hour
10	Revision -Question and Answers		1 Hour
		Total	10 Hours

Unit delivery:

The unit will be delivered through lectures notes, presentations, videos and support of modern tools.

Introduction:

ARM stands for Advance RISC Machines. The ARM processor is a key component of many successful 32-bit embedded systems. These processors widely used in mobile phones, handled organizers and multitude of other everyday portable consumer devices.

ARM designers have came a long way from the first ARM1 prototype in 1985. Over one billion ARM processors had been shipped worldwide by the end of 2001. The ARM Company bases their success on a simple and powerful original design which continues to improve today through constant technical innovations. In fact the ARM core is not a single core, but a whole family of designs sharing similar design principles and a common instruction set.

The ARM7TDMI is ARM's most successful core processor.

1.1: The RISC Design Philosophy:

The ARM core uses RISC architecture. The design philosophy aimed at delivering the following.

- Simple but powerful instructions
- Single cycle execution at a high clock speed
- Intelligence in software rather than hardware
- Provide greater flexibility on reducing the complexity of instructions.

The RISC philosophy is implemented with four major design rules:

1. Instructions.
2. Pipelines.
3. Register.
4. Load - Store architecture.

Instructions: – RISC processors have a reduced number of instruction classes. These classes provide simple operations that can each execute in a single cycle. The compiler or programmer synthesizes complicated operations (a divide operation) by combining several simple instructions. Each instruction is a fixed length to allow the pipeline to fetch future instructions before decoding the current instruction. In contrast, in CISC processors the instructions are often of variable size and take many cycles to execute.

Pipelines: —the processing of instructions is broken down into smaller units that can be executed in parallel by pipelines. Ideally the pipeline advances by one step on each cycle for maximum throughput. There is no need for an instruction to be executed by a mini program called microcode as on CISC processors.

Registers:—RISC machines have a large general-purpose register set. Any register can contain either data or an address. In contrast, CISC processors have dedicated registers for specific purposes.

Load-store architecture:—the processor operates on data held in registers. Separate load and store instructions transfer data between the register bank and external memory In contrast, with a CISC design the data processing operations can act on memory directly.

1.2: The ARM Design Philosophy.

There are a number of physical features that have driven the ARM processor design.

1. Small to reduce **power consumption** and extend battery operation.
2. **High code density** is essential for mass storage devices.
3. **Price** sensitive and use slow and low-cost memory devices.
4. Reduce the **area (size)** of the die taken up by the embedded processor.
5. **Hardware debug technology** within the processor which reduces the overall costs,
6. ARM core is not a pure RISC architecture.

1.3: Instruction Set for Embedded Systems.

The ARM instruction set differs from the pure RISC definition in several ways that make the ARM instruction set suitable for embedded applications:

Variable cycle execution for certain instructions— not every ARM instruction executes in a single cycle. For example, load-store-multiple instructions vary in the number of execution cycles depending upon the number of registers being transferred.

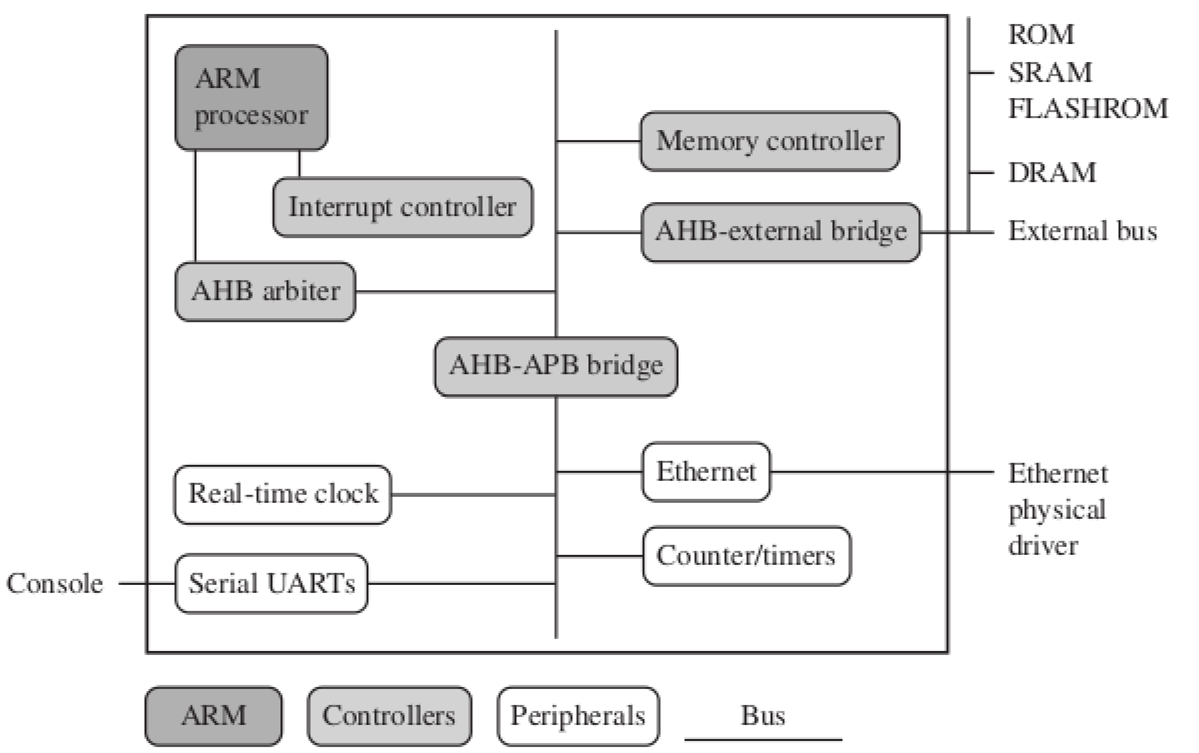
Inline barrel shifter leading to more complex instructions—the inline barrel shifter is a hardware component that preprocesses one of the input registers before it is used by an instruction.

Thumb 16-bit instruction set—ARM enhanced the processor core by adding a second 16-bit instruction set called Thumb that permits the ARM core to execute either 16- or 32-bit instructions.

Conditional execution—an instruction is only executed when a specific condition has been satisfied.

Enhanced instructions—the enhanced digital signal processor (DSP) instructions were added to the standard ARM instruction set to support fast 16×16-bit multiplier operations and saturation.

1.4: Embedded System Hardware



An example of an ARM-based embedded device, a microcontroller.

The above figure shows a typical embedded device based on an ARM core. The embedded system can control many different devices, all these devices use a combination of software and hardware components.

Each block does a specific function. The lines connecting the blocks are the buses used to carry the data. The embedded system hardware is divided into four main hardware components, they are

1. ARM processor.
2. Controllers.
3. Peripherals.
4. Bus.

ARM processor: The ARM processor controls the embedded device. An ARM processor comprises a core plus the surrounding components that interface it with a bus. These components can include memory management and caches.

Controllers: Controllers coordinate important functional blocks of the system. Two commonly found controllers are interrupt controller and memory controller.

Peripherals: The peripherals provide all the input-output capability external to the chip and are responsible for the uniqueness of the embedded device.

Bus: A bus is used to communicate between different parts of the device.

1.5 ARM Bus Technology

Embedded systems use different bus technologies. The most common PCI bus technology, the Peripheral Component Interconnect (PCI) bus, this type of technology is external or off-chip.

Embedded devices use an on-chip bus that is internal to the chip and that allows different peripheral devices to be interconnected with an ARM core.

There are two different classes of devices attached to the bus.

Bus master (ARM processor core)—a logical device capable of initiating a data transfer with another device across the same bus.

Bus slaves (Peripherals)—logical devices capable only of responding to a transfer request from a bus master device.

A bus has two architecture levels.

Physical level — that covers the electrical characteristics and bus width (16, 32, or 64 bits).

Protocol level—the logical rules that govern the communication between the processor and a peripheral.

1.6: AMBA Bus Protocol

The Advanced Microcontroller Bus Architecture (AMBA) has been widely adopted as the on-chip bus architecture used for ARM processors.

It was introduced in 1996 and has been widely adopted as the on-chip architecture used for ARM processors.

The first AMBA buses introduced were.

1. ARM System Bus (ASB)
2. ARM Peripheral Bus (APB).

Later ARM introduced another bus design,

3. ARM High Performance Bus (AHB).

Plug-and-play interface for hardware developers — Using AMBA, peripheral designers can reuse the same design on multiple projects. A peripheral can simply be bolted onto the on-chip bus without having to redesign an interface for different processor architecture.

More about AHB bus:

- Centralized multiplexed provides higher data throughput rather than the ASB bidirectional bus design.
- AHB runs at higher clock speed.
- It supports widths of 64 and 128 bits.
- ARM has introduced two variations on the AHB bus:

  a. Multi-layer AHB: It's a bus allow multiple active bus masters.

  b. AHB-Lite: It's a subset of the AHB bus and it is limited to a single bus master.

The AHB and Multi-layer AHB support the same protocol for master and slave but have different interconnects. The new interconnects in multi-layer AHB are good for system with multiple processors. They permit operations to occur in parallel and allow for higher throughput rates.

1.7: Memory Controllers and Interrupt Controllers

Memory Controllers:

- Connect different types of memory to the processor bus.
- On power-up a memory controller is configured in hardware to allow certain memory devices to be active.
- Some memory devices must be set up by software.

Interrupt Controllers:

An interrupt controller provides a programmable that allows software to determine which peripheral or device can interrupt the processor at any specific time by setting the appropriate bits in the interrupt controller registers.

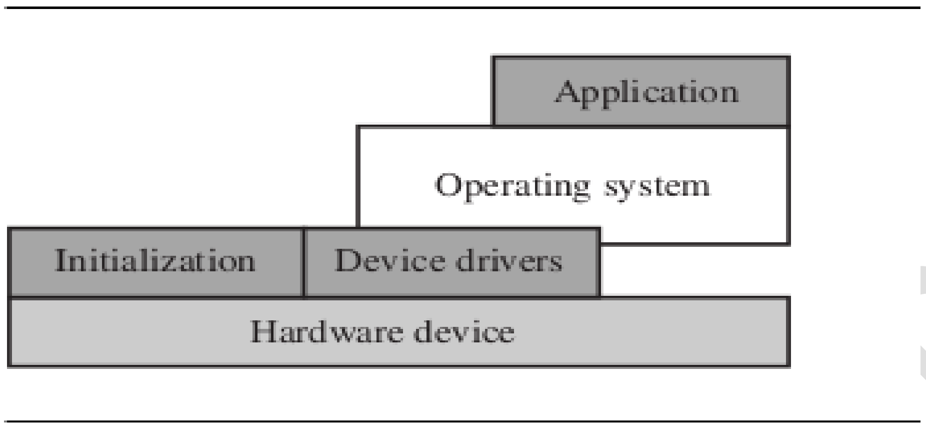
There are two types of interrupt controller available for the ARM processor:

Standard interrupt controller (SIC): The standard interrupt controller sends an interrupt signal to the processor core when an external device requests servicing. It can be programmed to ignore or mask an individual device or set of devices. The interrupt handler determines which device requires servicing by reading a device bitmap register in the interrupt controller.

Vector interrupt controller (VIC): The VIC is more powerful than the standard interrupt controller because it prioritizes interrupts. After associating a priority and a handler address with each interrupt, the VIC only asserts an interrupt signal to the core if the priority of a new interrupt is higher than the currently executing interrupt handler.

1.8: Embedded System Software

An embedded system needs software to drive it.



Software abstraction layers executing on hardware.

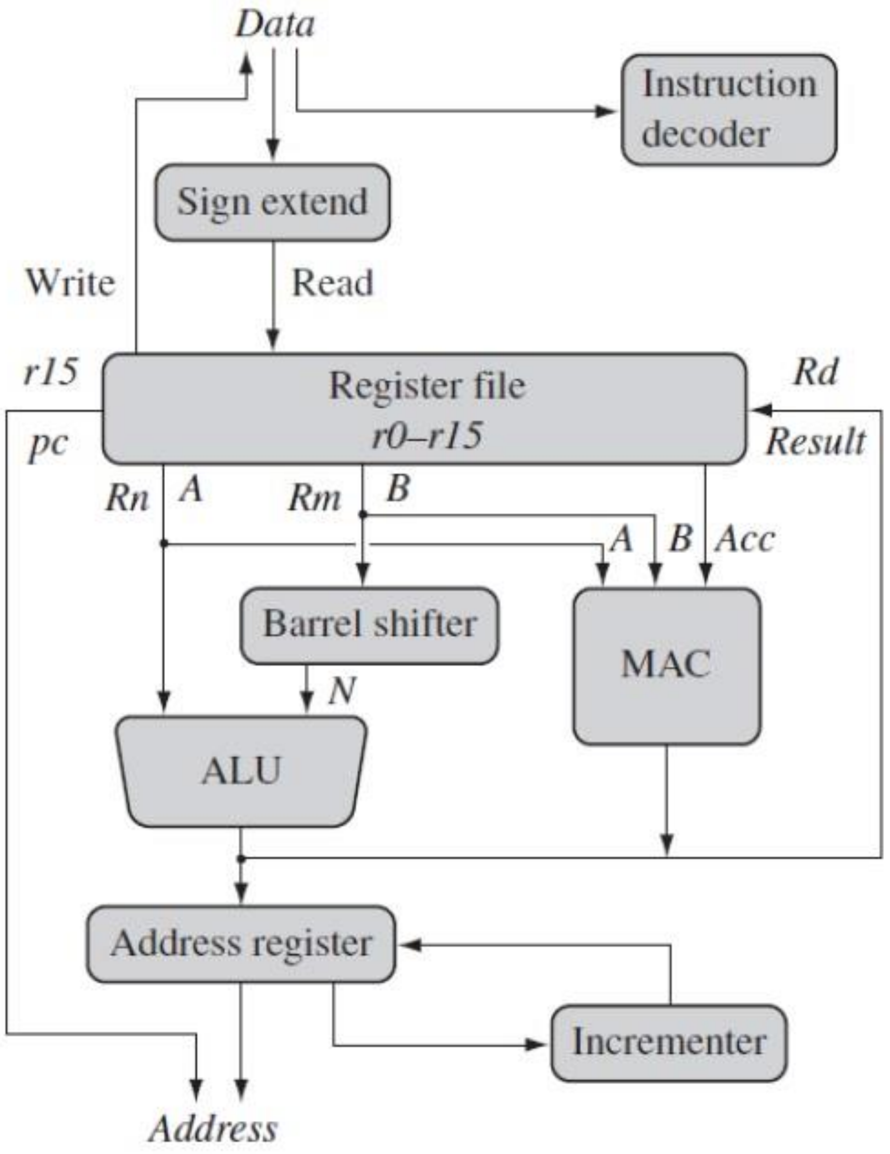
- Initial hardware configuration involves setting up the target platform so it can boot an image. The target platform itself comes up in a standard configuration. Diagnostic code is fault identification and isolation. Booting involves loading an image and handing control over to that image. The boot process itself can be complicated if the system must boot different operating systems or different versions of the same operating system.
- Initialization code is the first code executed on the board and its sets up the minimum parts of the board before handing control over to the operating system.
- Device drivers provide a consistent software interface to the peripherals on the hardware device.
- Operating system provides an infrastructure to control applications and manage hardware system resources. Many embedded systems do not require a full operating system.
- Application performs one of the tasks required for a device.

1.9: Applications of ARM Processor

ARM processor found in numerous market segments including the following

- Networking applications: Home gateways, DSL modems etc
- Mobile phone devices.
- Mass storage devices: Hard drives.
- Imaging products: Inkjet printers.
- Digital TV, Net books, etc.
- Portable Games Consoles: Nintendo DS, PlayStation Portable, Game Boy Advance, GP2x
- Portable Media Players, Camcorders: Apple iPod
- GPS Navigation Systems: Tom Tom 300
- Set Top Boxes, TVs, Hard Discs, Routers,

1.10: ARM core data flow model



The block diagram of an ARM core has following functional units.

- ALU (Arithmetic Logic Unit)
- Address register.
- Register File
- Barrel shifter
- MAC (Multiply Accumulate Unit)
- Instruction decoder.

All the above functional blocks are connected by data buses as shown in above figure. The arrows represents flow of data, the lines represents buses and boxes represents either an operation unit or storage area.

The above figure shows a Von-Neumann implementation of ARM in which the data items and instruction share a same bus. Data enters the processor core through data bus.

Instruction decoder translates the instructions into binary form before they are executed. Each instruction executed belongs to a particular instruction set.

The ARM processor uses Load-Store architecture ie it has two instruction types for transferring data in and out of the processor. They are

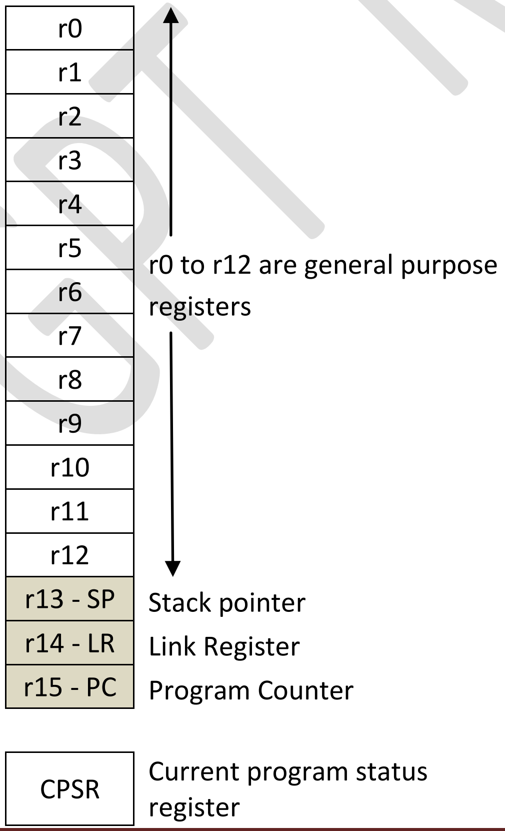
Load: This instruction copy data from memory to register in the core.

Store: This instruction copy data from register to memory in the core.

Operation:

- Data items placed in register file – a storage bank made up of 32-bit registers. Since ARM core is a 32-bit processor, most of the instruction treat the register as holding signed or unsigned 32-bit values.
- The sign extend hardware converts signed 8-bit or 16-bit numbers to 32-bit values as they are read from memory and placed in a register.
- ARM instruction typically have two source register R_n and R_m , and Single result or destination register R_d. Source operands are read from register file using internal buses A and B respectively.
- The ALU or MAC unit takes the register values R_n and R_m from the A and B buses and computes a result.
- Data processing instruction directly write the result to Rd register. Load and Store instructions use the ALU to generate an address to be held in the address register and broadcast on the address bus.
- One important feature of the ARM is that register Rm alternatively can be preprocessed in the barrel shifter before it enters the ALU. Together barrel shifter and ALU is used to calculate wide range of expression and addresses.
- After passing through the functional units, the result in the Rd is written back to the register file using the result bus.
- For Load and Store instruction the incrementer updates the address register before the core reads or writes next register value from or to the next sequential memory location. The processor continues executing instruction until an exception or interrupt changes normal execution flow.

1.11: ARM Registers



SPSR	Saved program status register

General purpose registers hold either data or an address. They are identified with the letter 'r' prefixed to the register number. For example, register 4 is given the label r4. The above figure shows the active registers available in the user mode.

● All registers are 32-bit in size.

● 18 active registers

● 16 data registers – r0 to r15

● 2 process status registers

– CPSR – Current Program Status Register

– SPSR – Saved Program Status Register

● r13, r14, r15 have special functions

● r13 – stack pointer (sp) and stores the head of the stack in the current processor mode

● r14 – link register (lr) where the core puts the return address whenever it calls a subroutine.

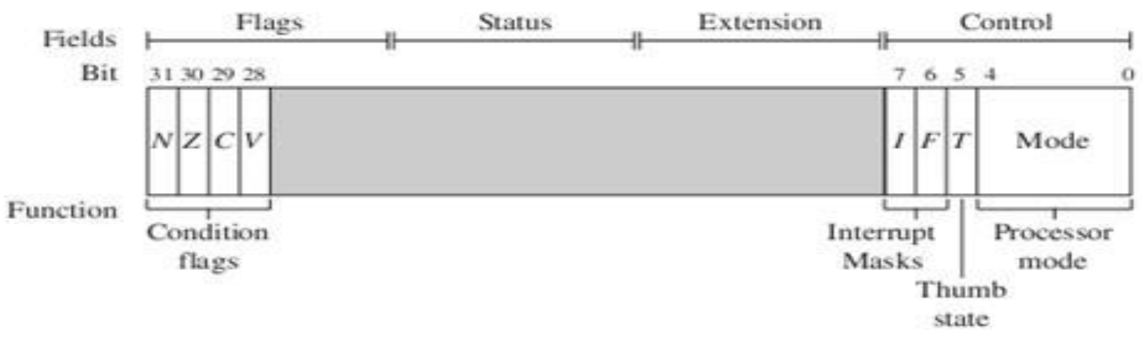
● r15 – program counter (pc) and contains the address of the next instruction to be fetched by the processor.

● r13 and r14 can also be used as general purpose register as these registers are banked during a processor mode change.

● the registers r0 to r13 are orthogonal. Any instruction that you can apply to r0, you can equally apply to other registers.

● there are instructions that treat r14 and r15 in a special way.

1.12: CPSR (Current Program Status Register)



- The ARM core uses the CPSR to monitor and control internal operations. The CPSR is dedicated 32-bit register and resides in register file.
- The above figure shows basic layout of current program status register and it is divided into four fields each 8-bit wide as shown above.
  1. Flags 2. Status 3. Extension 4. Control
- In the current design, Extension and status fields are reserved for future use.

- The flag field contains the conditional flags such as negative flag, zero flag, carry flag and overflow flag.
- The control field contains processor mode bits, state bit and interrupts mask bits.

Conditional Flags:

Bit 31 – N – Negative flag: This flag bit indicates the result of arithmetic and logical operation is negative.

N = 0, Result is positive.

N = 1, Result is negative.

Bit 30 – Z – Zero flag: This flag bit indicates the result of arithmetic and logical operation is ZERO.

Z = 0, Result is positive (Unsigned).

Z = 1, Result is negative (Signed).

Bit 29 – C – Carry flag: This flag bit indicates the overflow occurred or not after the addition.

C = 0, Carry has not occurred after addition.

C = 1, Carry occurred after addition.

Bit 28 – V – Overflow flag: This flag bit indicates the signed overflow has occurred or not ie result has exceed the limit or not.

V = 0, Overflow not occurred.

V = 1, Overflow occurred.

Control Bits:

Bit 7 – I – Interrupt Request (IRQ)

I = 0, Enables the IRQ Interrupts.

I = 1, Disables the IRQ Interrupts.

Bit 6 – F – Fast Interrupt Request (FIQ)

F = 0, Enables the FIQ Interrupts.

F = 1, Disables the FIQ Interrupts.

Bit 5 – T – Thumb

T = 0, Indicates the ARM state.

T = 1, Indicates the THUMB state.

Bit 4 to 0 – Processor mode bits.

These bits are used for selection of operating modes is done as follows.

CPSR Mode bits					Operating Mode
M4	M3	M2	M1	M0	
1	0	1	1	1	Abort
1	0	0	0	1	Fast Interrupt Request
1	0	0	1	0	Interrupt Request
1	0	0	1	1	Supervisor
1	1	1	1	1	System
1	1	0	1	1	Undefined
1	0	0	0	0	User

1.13: Processor Operating Modes:

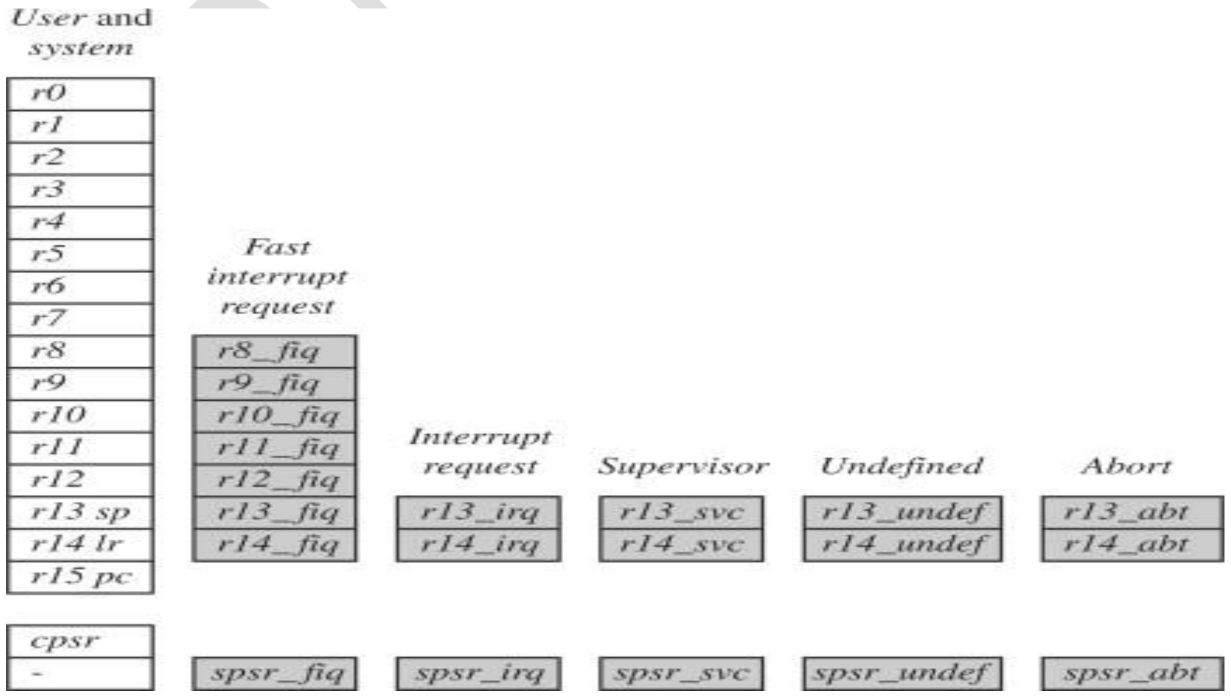
The processor mode determines which register are active and the access rights to the CPSR itself.

- Each processor mode is privileged or non- privileged.
  - A privileged mode allows full read-write access to CPSR and Non- privileged mode allows only read access to CPSR but still allows read-write access to conditional flags.
- ARM processor has seven processor modes. Out of this six privileged modes and one Non-privileged mode. They are
  - **Privileged modes:** Abort, Fast interrupt request, Interrupt request, Supervisor, System, Undefined.
  - **Non- Privileged mode:** User.
- The processor enters abort mode when there is a failed attempt to access memory.
- Fast interrupt request & Interrupt request modes correspond to the two interrupt levels available on the ARM processor.
- Supervisor mode is the mode that the processor is in after reset and is generally the mode that an operating system kernel operates in.
- System mode is a special version of user mode that allows full read-write access to the CPSR.
- Undefined mode the used when processor encounters an instruction that is undefined or not supported by the implementation.
- User mode is used for programs and applications.

Table: List of Various modes and its binary patterns in the cpsr register

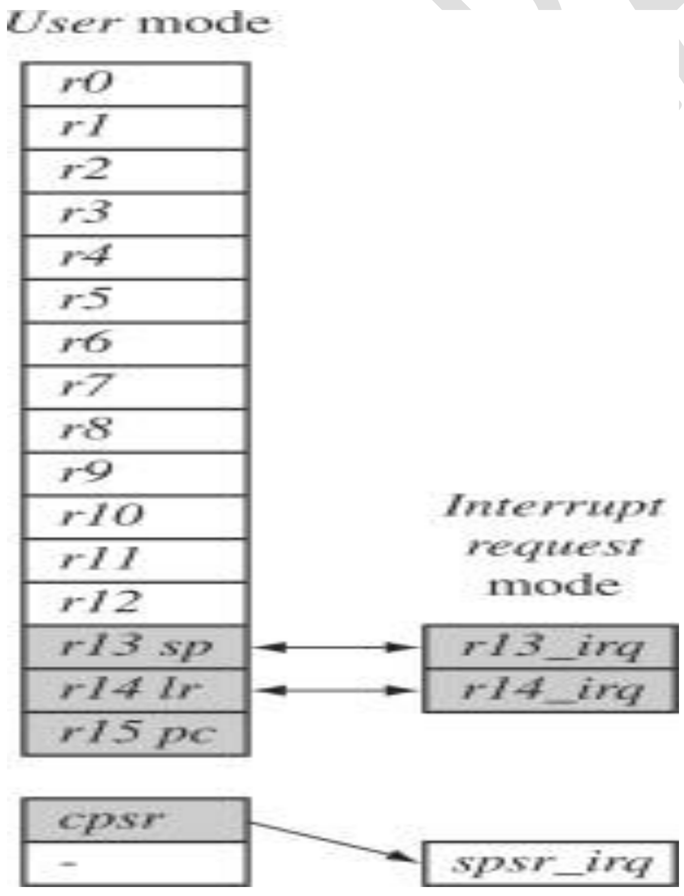
Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast Interrupt Request	fiq	yes	10001
Interrupt Request	irq	yes	10010
Supervisor	sve	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

1.14: Banked Registers



- The above figure shows all 37 registers in the register file. Of those, 20 registers are hidden from a program at different times. These registers are called banked registers and are identified by the shading in the diagram.
- Banked registers are available only when the processor is in a particular mode; for example, abort mode has banked registers r13_abt, r14_abt and spsr_abt.
- Banked registers of a particular mode are denoted by an underline character post-fixed to the mode mnemonic or _mode.
- Every processor mode except user mode can change mode by writing directly to the mode bits of the cpsr.
- All processor modes except system mode have a set of associated banked registers that are a subset of the main 16 registers.
- A banked register maps one-to-one onto a user mode register. If you change processor mode, a banked register from the new mode will replace an existing register.
    - For example, when the processor is in the interrupt request mode, the instructions you execute still access registers named r13 and r14. However, these registers are the banked registers r13_irq and r14_irq.
- The user mode registers r13_usr and r14_usr are not affected by the instruction referencing these registers. A program still has normal access to the other registers r0 to r12.
- The processor mode can be changed by a program that writes directly to the cpsr (the processor core has to be in privileged mode) or by hardware when the core responds to an exception or interrupt.
- The following exceptions and interrupts cause a mode change: reset, interrupt request, fast interrupt request, software interrupt, data abort, pre fetch abort, and undefined instruction. Exceptions and interrupts suspend the normal execution of sequential instructions and jump to a specific location.

Example: What happens when an interrupt forces a mode change?



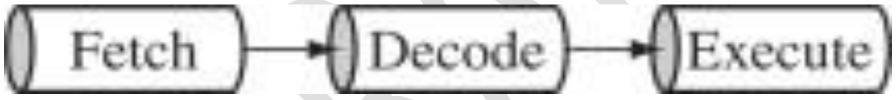
- The figure shows the core changing from user mode to interrupt request mode, which happens when an interrupt request occurs due to an external device raising an interrupt to the processor core. This change cause's user registers r13 and r14 to be banked.
- The user registers are replaced with registers r13_irq and r14_irq, respectively. Note r14_irq contains the return address and r13_irq contains the stack pointer for interrupt request mode.
- The figure also shows a new register appearing in interrupt request mode: the saved program status register (spsr), which stores the previous mode's cpsr. You can see in the diagram the cpsr being copied into spsr_irq.
- To return back to user mode, a special return instruction is used that instructs the core to restore the original cpsr from the spsr_irq and bank in the user registers r13 and r14.
- Note that the spsr can only be modified and read in a privileged mode. There is no spsr available in user mode.
  Another important feature to note is that the cpsr is not copied into the spsr when a mode change is forced due to a program writing directly to the cpsr. The saving of the cpsr only occurs when an exception or interrupt is raised.

1.15: Pipelining:

A pipeline is the mechanism a RISC processor uses to execute instructions. Using a pipeline speeds up execution by fetching the next instruction while other instructions are being decoded and executed.

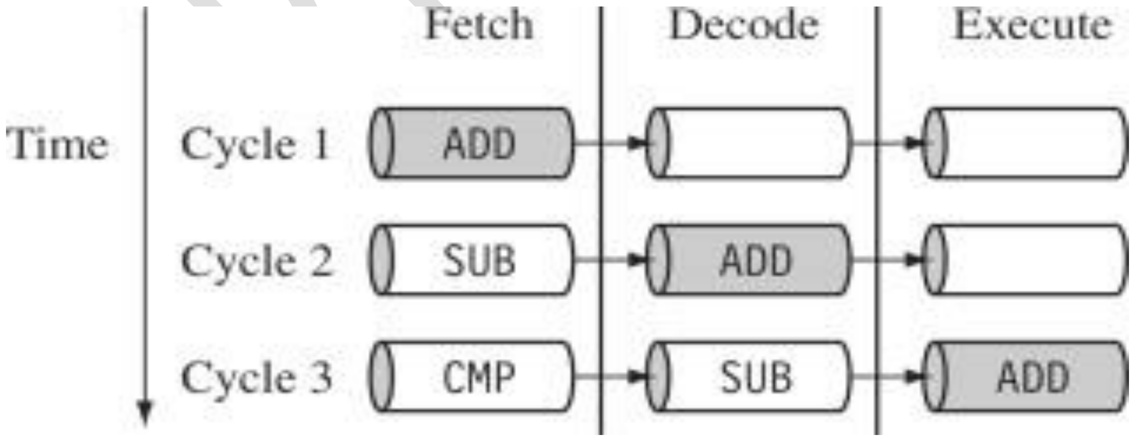
One way to view the pipeline is to think of it as an automobile assembly line, with each stage carrying out a particular task to manufacture the vehicle.

Three-stage pipeline:



- Fetch: It loads an instruction from memory.
- Decode: It identifies the instruction to be executed.
- Execute: It processes the instruction and writes the result back to a register.

Example: ARM7 three-stage pipelining



The above figure shows a sequence of three instructions being fetched, decoded, and executed by the processor. Each instruction takes a single cycle to complete after the pipeline is filled.

The three instructions are placed into the pipeline sequentially.

First cycle: the core fetches the ADD instruction from memory.

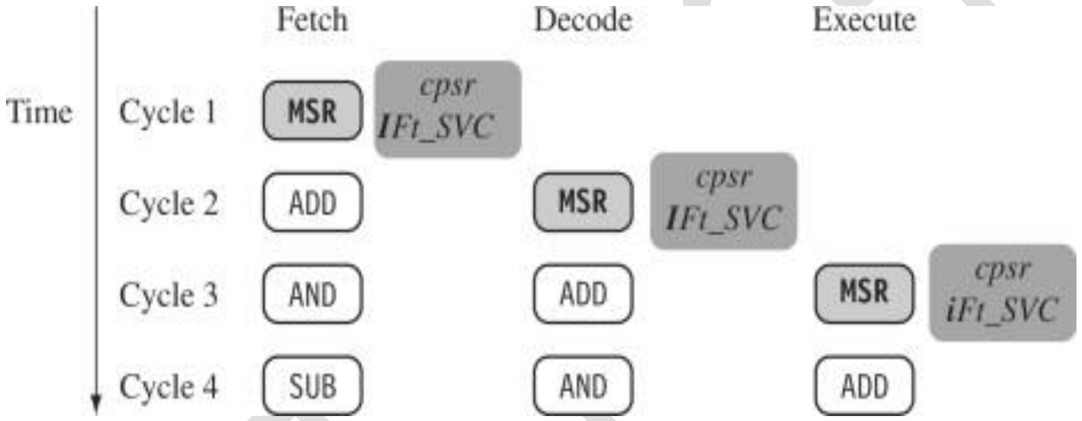
Second cycle: the core fetches the SUB instruction and decodes the ADD instruction.

Third cycle: both the SUB and ADD instructions are moved along the pipeline.

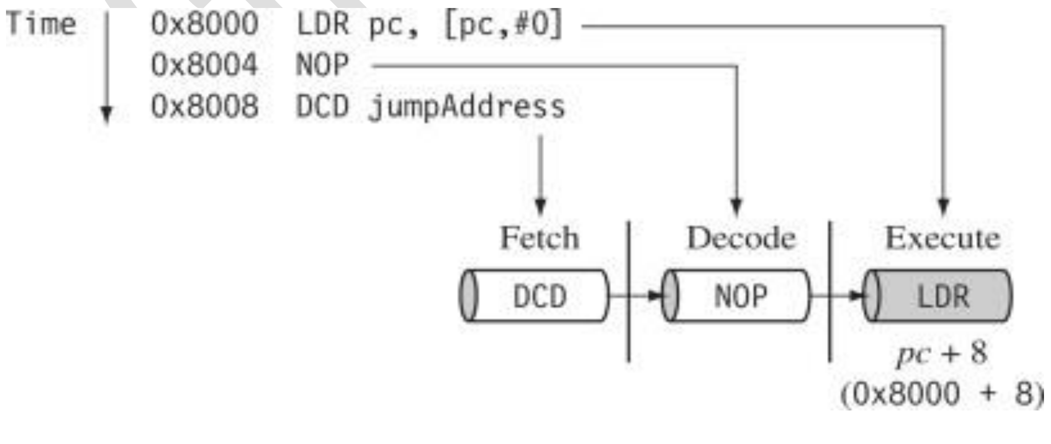
The ADD instruction is executed, the SUB instruction is decoded, and the CMP instruction is fetched. This procedure is called filling the pipeline.

- The pipeline allows the core to execute an instruction every cycle.
- As the pipeline length increases, the amount of work done at each stage is reduced, which allows the processor to attain a higher operating frequency. This in turn increases the performance.
- The system latency also increases because it takes more cycles to fill the pipeline before the core can execute an instruction.
- The increased pipeline length also means there can be data dependency between certain stages. You can write code to reduce this dependency by using instruction scheduling.

1.15.1: Pipeline Characteristics.



The above figure shows an instruction sequence on an ARM7 pipeline. The MSR instruction is used to enable IRQ interrupts, which only occurs once the MSR instruction completes the execute stage of the pipeline. It clears the I bit in the cpsr to enable the IRQ interrupts. Once the ADD instruction enters the execute stage of the pipeline, IRQ interrupts are enabled.


```
Time | 0x8000  LDR pc, [pc,#0]
     | 0x8004  NOP
     ↓ 0x8008  DCD jumpAddress
```

The above figure illustrates the use of the pipeline and the program counter pc. In the execute stage, the pc always points to the address of the instruction plus 8 bytes. In other words, the pc always points to the address of the instruction being executed plus two instructions ahead.

This is important when the pc is used for calculating a relative offset and is an architectural characteristic across all the pipelines. Note when the processor is in Thumb state the pc is the instruction address plus 4.

There are three other characteristics of the pipeline:

First: the execution of a branch instruction or branching by the direct modification of the pc causes the ARM core to flush its pipeline.

Second: ARM10 uses branch prediction, which reduces the effect of a pipeline flush by predicting possible branches and loading the new branch address prior to the execution of the instruction.

Third: an instruction in the execute stage will complete even though an interrupt has been raised. Other instructions in the pipeline will be abandoned, and the processor will start filling the pipeline from the appropriate entry in the vector table.

Review Questions:

Si No	Question	C.L	Marks
1.1	**The RISC design philosophy**		
1	Explain the major design rules of RISC system design.	U	5
2	Write the advantages and drwabacks of RISC system.	R	5
1.2	**ARM design philosophy**		
3	List the special features of ARM processor design.	R	5
4	Justify how ARM is suitable for mobile applications.	U	5
5	Justify the features that improve code density.	U	5
1.4	**Embedded system hardware**		
6	Explain the block diagram of ARM based embedded device.	U	10
7	Explain two architecture levels of bus.	U	3
8	Explain AMBA bus protocols.	U	7
1.8	**Embedded system software and Applications of ARM Processor.**		
9	Explain the role of software components in embedded system.	U	5
10	List the applications of ARM processor.	R	5
1.10	**ARM core data flow model**		
11	Sketch near ARM data flow model	R	5
12	Explain ARM core data flow model	U	10
13	Explain the function of sign extend hardware	R	5
1.11	**ARM registers**		
14	Explain the register file of ARM processor with a neat sketch	U	7
15	Explain active register in user mode	U	5
16	Explain special function register of ARM	U	3
1.12	**CPSR and Processor modes**		
17	Explain the bit structure of CPSR	U	10
18	Discuss the mode bits of CPSR register.	U	3

19	Describe the functions of flags of CPSR register.	R	5
20	Define processor mode. List different processor modes of ARM core	R	5
21	Explain processor modes.	U	5
1.14	**Banked registers.**		
22	Define Banked registers. Explain briefly.	R	5
23	Explain banked registers with neat sketch.	U	10
1.15	**Pipeline**		
24	Define pipelining.	R	3
25	Explan 3-stage pipelining.	U	5
26	Write the advantages and disadvantages of pipelining.	R	5
27	Discuss the characteristics of pipelining.	U	55

Sd/--

VITHALANI PAESH K.

Sr GR L/EC, GPT Raichur (117)

Email id: pareshkv2015@gmail.com