MAILAM ENGINEERING COLLEGE

Mailam, Villupuram (Dt), Pin – 604304
(Approved by AICTE, New Delhi, Affiliated to Anna University Chennai,
Accredited by NBA, NewDelhi & Accredited by TCS)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SUB CODE/SUB NAME : EC8791 EMBEDDED AND REAL TIME SYSTEMS

YEAR/SEM : IV ECE / VII

TOTAL NUMBER OF PAGES :

TOTAL NUMBER OF COPIES :

UNIT II ARM PROCESSOR AND PERIPHERALS

ARM Architecture Versions – ARM Architecture – Instruction Set – Stacks and Subroutines – Features of the LPC 214X Family – Peripherals – The Timer Unit – Pulse Width Modulation Unit – UART – Block Diagram of ARM9 and ARM Cortex M3 MCU.

TEXT BOOKS:

1. Marilyn Wolf, —Computers as Components - Principles of Embedded Computing System Design‖, Third Edition —Morgan Kaufmann Publisher (An imprint from Elsevier), 2012. (UNIT I, II, III, V)
2. Jane W.S.Liu,‖ Real Time Systems‖, Pearson Education, Third Indian Reprint, 2003.(UNIT IV)

REFERENCES:

1. Lyla B.Das, —Embedded Systems : An Integrated Approach‖ Pearson Education, 2013.
2. Jonathan W.Valvano, —Embedded Microcomputer Systems Real Time Interfacing‖, Third Edition Cengage Learning,
2012.
3. David. E. Simon, —An Embedded Software Primer‖, 1st Edition, Fifth Impression, AddisonWesley Professional, 2007.

Prepared by

1. **RAMATHILAGAM.B, AP/ECE**

2. **MOHANA DEVI.C, AP/ECE**

VERIFIED BY BATCH COORDINATOR HOD/ECE PRINCIPAL

PART A

1. List the functions of ARM processor in supervisor mode.(OR)List the major functions of CPU in supervisor mode.(or) Enumerate functions of ARM processor in supervisor mode (APRIL 2014,MAY 2015)(Apr/May 16,18)(Nov/Dec 18)

❖ The supervisor mode has privileges that user modes do not which can be entered by reset.

❖ ARM instruction that puts the CPU in supervisor mode is called SWI:

❖ **SWI CODE_1**

❖ SWI causes the CPU to go into supervisor mode .

❖ In supervisor mode, the bottom 5 bits of the CPSR are all set to 1 to indicate that the CPU is in supervisor mode. The old value of the CPSR just before the SWI is stored in a register called the *saved program status register (SPSR)*.

❖ It is protected mode for operating system.

2. What are the instruction set features useful for embedded programming.[MAY/JUNE 2013]

✓ Many early computer architectures were what is known today as *complex instruction set computers (CISC)*. These machines provided a variety of instructions that may perform very complextasks, such as string searching; *they also generally used a number of different instruction formats of varying lengths.*

✓ One of the advances in the development of high-performance microprocessors was the concept of *reduced instruction set computers (RISC)*. *These computers tended to provide somewhat fewer and simpler instructions.*

3. **Types of operations supported in the ARM processor**.

❖ Arithmetic data operations which includes ADD,SUB,MUL,etc

❖ Logical data operations which includes AND,ORR,EOR,etc

❖ Shift/rotate data operations which includes LSL,LSR,ROR,etc

❖ Comparison instructions which includes CMP,TST,etc

❖ Move instructions which includes MOV, MVN etc

❖ Load/store instructions which includes LDR,STR etc..

4. What are the parameters used to evaluate the CPU performance?[MAY/JUNE 2013]

The parameters used to evaluate the CPU performance*Pipelining and caching.*

Pipelining:

The ARM 7 has a three-stage pipeline:

- *Fetch* the instruction is fetched from memory
- *Decode* the instruction's opcode and operands are decoded to determine what function to perform
- *Execute* the decoded instruction is executed.

Caching

❖ A cache is a small, fast memory that holds *copies of the contents of main memory.* Because the cache is fast, it provides high speed access for the CPU.

❖ CPU is using only small set of memory locations at any one time, the set of active locations is often called the *working set*.

5. Enumerate various issues in real time computing.[NOV/DEC 2013]

❖ Real-time programs are required to *finish their work within a certain amount of time;* if they run too long, they can create much unexpected behavior.

❖ The exact results of missing real-time deadlines depend on the detailed *characteristic of the I/O devices* and the nature of the timing violation.This makes debugging real-time problems especially difficult.

6. Write short notes on ARM processor. .[NOV/DEC 2013]

❖ ARM is actually a *family of RISC architectures* that have been developed over many years.

❖ ARM does not manufacture its own VLSI devices; rather, *it licenses its architecture* to companies who either manufacture the CPU itself or integrate theARM processor into a larger system.

❖ *ARM instructions* are written one perline, starting after the first column. *Comments* begin with a semicolon and continueto the end of the line.

❖ *A label*, which gives a name to a memory location,comes at the beginning of the line, starting in the first column.

Example:

```
LDR r0,[r8]; a comment

label ADD r4,r0,r1
```

7. What is function of exceptions? .[NOV/DEC 2012] (Nov/Dec 16)

❖ *An **exception** is an internally detected error. A simple example is division by zero.*

❖ *The exception mechanism provides a way for the program to react to such unexpected events. Exceptions are generally implemented as a variation of an interrupt.*

❖ *Exceptions in general require both prioritization and vectoring.*

❖ Exceptions must **be prioritized** because a single operation may generate more than one exception— for example, an illegal operand and an illegal memory access. The priority of exceptions is usually fixed by the CPU architecture.

❖ **Vectoring** provides a way for the user to specify the handler for the exception condition. The vector number for an exception is usually predefined by the architecture;it is used to index into a table of exception handlers.

8. How is ARM processor different from other processor?(or) Differentiate CISC and RISC processor [NOV/DEC 2012](or) Give two features that differ a general purpose microcontroller from an embedded systems.(NOV/DEC 2015)

ARM processor	Other processor
Arm is designed on RISC Architecture. RISC(Reduced Instruction set computer): Emphasis on software Single-clock, reduced instruction only. Register to register: "LOAD" and "STORE" are independent instructions. Low cycles per second, large code sizes. Spends more transistors on memory registers.	Intel and Amd are designed on x86 CISC architecture. X86/CISC(Complex Instruction set computer): Is more hardware based. It uses little-endian byte order. Where the least significant bytes get stored in lower address spaces

9. Define embedded computer system and what are the challenges in embedded computing system design.(Or)Why embedded computing is more suitable for real time systems?
(Nov /Dec 2015) (Apr/May 16)
Embedded computer system

❖ It is any device that includes a **programmable computer** but is not itself intended to be a general-purpose computer.

❖ Thus, a PC is not itself an embedded computing system, although PCs are often used to build embedded computing systems.

❖ But a **fax machine or a clock** built from a microprocessor is an embedded computing system.

Challenges in embedded computing system design

- Hardware constraints
- Meeting a deadline
- Minimizing power consumption
- Design for upgradability
- Really working or not.

10.What are all the application areas of the embedded systems?

- Telecom Smart Cards,
- Missiles and Satellites,

- Computer Networking, Automotive
- Digital Consumer Electronics, and

11. Define compiler, linker and loader?

Compiler

❖ A **compiler** is a computer program (or set of programs) that **transforms source code** written in a programming language (the source language) **into another computer language** (the target language, often having a binary form known as object code).

❖ The most common reason for converting a source code is to create an executable program.

Linker

❖ A **linker** or link editor is a computer program that takes one or more **object files** generated by a compiler

❖ It combines them into a single executable file, library file, or another object file.

Loader

❖ In computing, a **loader** is the part of an operating system that is responsible for **loading programs and libraries**. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

12. What are the types of power management features in CPU?

There are two types of power management features provided by CPUs.

- **Static power management**
- **Dynamic power management**

❖ A ***static power management*** mechanism is invoked by the user but does not otherwise depend on CPU activities. An example of a static mechanism is a ***power down mode*** intended to save energy. This mode provides a high-level way to reduceunnecessary power consumption.

❖ A ***dynamic power management*** mechanismtakes actions to control power based upon the dynamic activity in the CPU. Forexample, the CPU may turn off certain sections of the CPU when the instructions being executed do not need them.

13. Define Co processor ,sequence diagram and pipelining

Coprocessor

❖ ***co-processors***, *which are attached to the CPU and implement some ofthe instructions*.

❖ When the CPU receives a co-processor instruction, the CPU must activate the co-processor and pass it the relevant instruction.

❖ Co-processor instructions **can load and store** co-processor registers or can perform

internal operations. The CPU can suspend execution to wait for the co-processor instruction to finish.

❖ TheARMarchitecture provides support for up to **16 co-processors**.

❖ The unit occupies twoco-processor units in the ARM architecture, numbered 1 and 2, but it appears as asingle unit to the programmer. It provides eight 80-bit floating-point data registers, floating-point status registers, and an optional floating-point status register.

Sequence diagram

❖ A ***sequence diagram*** is somewhat similar to a hardware timing diagram, although the time flows verticallyin a sequence diagram, whereas time typically flows horizontally in a timingdiagram.*The sequence diagram is designed to show a particular scenario or choiceof events*—it is not convenient for showing a number of mutually exclusive possibilities.

❖ Processing includes three objects shown at the top of the diagram.

❖ Extending below each object is its *lifeline*, a dashed line that shows how long the Object is alive.

❖ The boxes along the lifelines showthe *focus of control* in the sequence,that is,when the objectis actively processing. In this case, the mouse object is active only long enough tocreate the *mouse_click*event. The display object remains in play longer; it in turnuses call events to invoke the menu object twice: once to determine which menu

❖ Item was selected and again to actually execute the menu call. The find_region( )call is internal to the display object,so it does not appear as an event in the diagram.

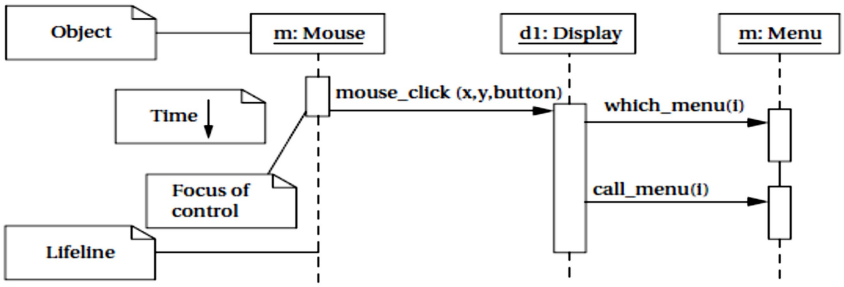


FIGURE 1.13
A sequence diagram in UML.

Pipelining

Modern CPUs are designed as ***pipelined*** machines in which several instructionsare executed in parallel. Pipelining greatly increases the efficiency of the CPU.
The ARM 7 has a three-stage pipeline:

- *Fetch* the instruction is fetched from memory
- *Decode* the instruction's opcode and operands are decoded to determine what function to perform
- *Execute* the decoded instruction is executed.

14. Differentiate Von Neumann Architecture and Super Harvard Architecture (SHARC)?

Von Neumann Architecture	Harvard Architecture
The memory holds both data and instructions, and can be read or written whengiven an address. A computer whose memory holds both data and instructions is known as a ***von Neumann*** machine	Harvard machine has **separate memories** for data and program.
The CPU has several internal ***registers*** that store values used internally. One of those registers is the ***program counter (PC)***,which holds the address in memory of an instruction.The CPU fetches the instruction frommemory,decodes the instruction,and executes it. The program counter does not directly determine what themachine does next, but only indirectly by pointing to an instruction in memory	The program counter points to program memory, not data memory. As a result, it isharder to write self-modifying programs (programs that write data values, then use those values as instructions) on Harvard machines

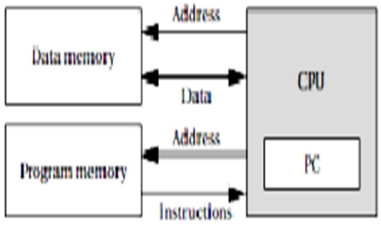


FIGURE 2.2
A Harvard architecture

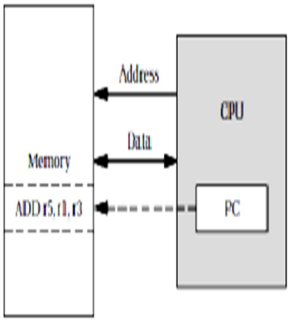


FIGURE 2.1
A von Neumann architecture computer.

15. Name the registers used in ARM processor.

❖ ARM has **16 general-purpose registers, r0 through r15**. Except for r15, they are identical—any operation that can be done on one of them can be done on the other one also. The r15 register has the same capabilities as the other registers, but it is also used as the program counter..

❖ The other important basic register in the programming model is the ***current program status register (CPSR)***. This register is set automatically during every arithmetic, logical, or shifting operation. The top four bits of the CPSR hold the following useful information about the results of that arithmetic/logical operation:

- The negative (N) bit is set when the result is negative in two's-complement arithmetic.
- The zero (Z) bit is set when every bit of the result is zero.
- The carry (C) bit is set when there is a carry out of the operation.
- The overflow(V) bit is set when an arithmetic operation results in an overflow.

16. What is an embedded system?

An embedded system is one that has **computer-hardware with software embedded in it as one of its most important component**. It is a computing device that does a specific focused job.

17. What is watch dog timer?

❖ A timer with timeout from which resets the processor in case the program gets stuck for an unexpected time.

❖ An important timing device in a system that **resets the system after a pre-defined timeout**. This time may be definable within the first few clock cycles after reset.

18. What is kernel?

A program with functions for memory allocation and deallocation,tasks scheduling, inter-Process communication, effective management of shared memory access by using theSignals ,exception handling signals, semaphores,queues,mailboxes,pipes,i/oManagement, interrupt controls, device drivers and device management.

19. **Eneumerate some embedded computers that are exists from origin of embedded systems (Nov/Dec 16)**

- Apollo Guidance Computer,

- Autonetics D-17 guidance computer

20. **Mention the various methods for reading from or writing to an I/O port (Apr/May 17)**

Microprocessors can provide programming support for input and output in two ways: ***I/O instructions***and ***memory-mapped I/O***.

21. **What are the role of microprocessor in Embedded system? (Nov 17)**

- First, microprocessors execute programs very efficiently.
- Second, microprocessor manufacturers spend a great deal of money to make their CPUs run very fast.

22.**How traps are handled in ARM processor (Nov 17)**

- A ***trap***,also known as a ***software interrupt***, is an instruction that explicitly generates an exception condition. The most common use of a trap is to enter supervisor mode.

23. Compare the functions of CPU and Co-processor (Apr/May 19)

CPU	CO-PROCESSOR
CPU is the main processing unit of the computer that performs arithmetic, logic and control operations according to the instructions	**Co-processors**are attached to the CPU and implement some of the instructions.
Maintains proper functioning of the entire computer	It helps the processor to increase the system performance

24. **Define assembler(Apr/May 19)**

- ***Assembler*** splits the program line by line.
- Assemblers must also provide some ***pseudo-ops*** to help programmers create complete assembly language programs.

25. Define RISC and CISC.

- **Many early computer architectures were what is known today as *complex instruction set computers (CISC).***
- **These machines provided a variety of instructions that may perform very complex tasks, such as string searching; they also generally used a number of different instruction formats of varying lengths**
- ***RISC***

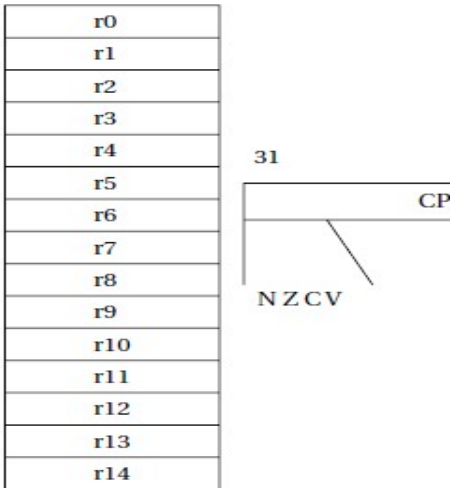
- **One of the advances in the development of high-performance microprocessors was the concept of *reduced instruction set computers (RISC).These computers tended to provide somewhat fewer and simpler* instructions. The instructions were also chosen so that they could be efficiently executed in *pipelined processors.***

Early RISC designs substantially outperformed **CISC designs of the period. As it turns out, we can use RISC techniques to efficiently execute at least a common subset of CISC instruction sets, so the performance gap between RISC-like and CISC-like instruction sets has narrowed somewhat.**

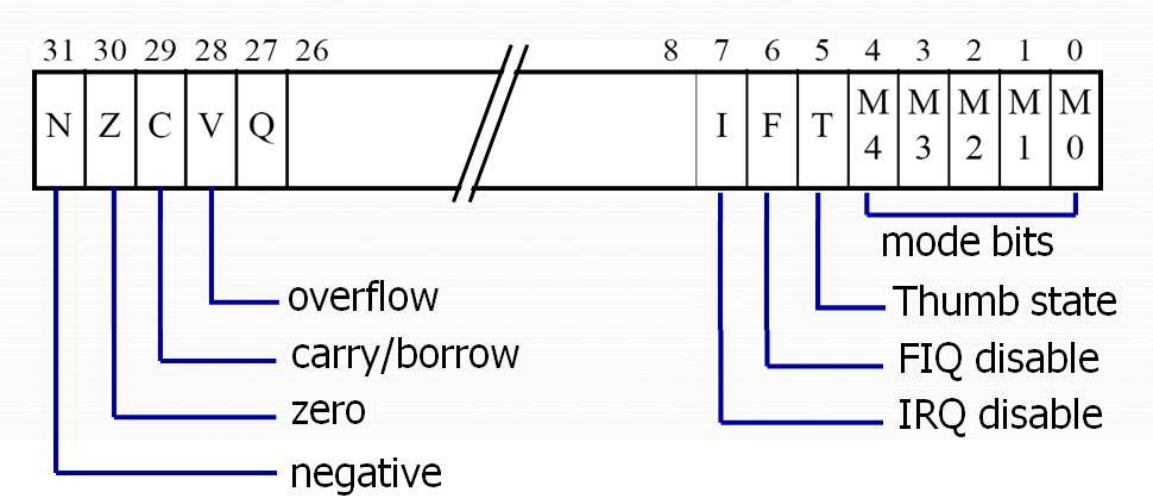
26. Definition of ARM.

- **ARM is actually a family of RISC architectures that have been developed over many years.**
- **ARM does not manufacture its own VLSI devices; rather, it licenses its architecture to companies who either manufacture the CPU itself or integrate theARM processor into a larger system.**

27. Draw the Basic diagram of ARM Programming Model.



28. Write a neat sketch of Register CPSR



29. What are the operating Modes in ARM processor?

Processor mode		Description
User	usr	Normal program execution mode
FIQ	fiq	Supports a high-speed data transfer or channel process
IRQ	irq	Used for general-purpose interrupt handling
Supervisor	svc	A protected mode for the operating system
Abort	abt	Implements virtual memory and/or memory protection
Undefined	und	Supports software emulation of hardware coprocessors
System	sys	Runs privileged operating system tasks

30. Difference between CISC and RISC

CISC	RISC
Variable size instructions with many formats	Fixed size instructions (32 bit)with few formats
Multi clock complex instructions.	Single clock reduced instructions.
Memory to memory load andstore instructions	**Register to register load and**store
Small code size, high cycles per second.	Large code size, Low cycles per second.
Emphasis on hardware	Emphasis on software
Increased hardware cost.	Reduced hardware cost.

PART-B

1. EXPLAIN COMPLEX SYSTEMS AND MICROPROCESSORS

Sub-Topics:

A. Embedding Computers

B. Characteristics of Embedded Computing Applications

C. Why Use Microprocessors?
D. The Physics of Software
E. Challenges in Embedded Computing System Design
F. Performance in Embedded Computing
G. Different levels of abstraction

A. Embedding Computers

An Embedded system

An embedded system is one that has **computer-hardware with software embedded in it as one of its most important component.** It is a computing device that does a specific focused job.

B) Characteristics of Embedded Computing Applications

Complex algorithms:

- The operations performed by the microprocessor may be very sophisticated.

User interface:

- Microprocessors are frequently used to control complex user interfaces **that may include multiple menus and many options.**
- The moving maps in Global Positioning System (GPS) navigation are good examples of sophisticated user interfaces.

Real time:

- Many embedded computing systems have to performing real time— **if the data is not ready by a certain deadline, the system breaks.**
- In some cases, failure to meet a deadline is unsafe and can even endanger lives.

Multirate:

- Not only must operations be completed by deadlines, but many embedded computing systems have **several real-time activities going on at the same time.**

Manufacturing cost:

- The total cost of building the system is very important in many cases.
- Manufacturing cost is determined by many factors, including the type of microprocessor used, the amount of memory required, and the types of I/O devices.

Power and energy:

- Power consumption directly affects the cost of the hardware, since a larger power supply may be necessary.
- **Energy consumption affects battery life, which is important in many applications**, as well as heat consumption, which can be important even in desktop applications.

C) Why Use Microprocessors?

■ Microprocessors are a very efficient way to implement digital systems.

■ Microprocessors make it easier to design families of products that can be built to provide various feature sets at different price points and can be extended to provide new features to keep up with rapidly changing markets.

There are two factors that work together to make microprocessor-based designs fast.

- First, microprocessors **execute programs very efficiently.**
- Second, microprocessor manufacturers spend a great deal of money to make their **CPUs run very fast**.

D) The Physics of Software

- **Computing** is a physical act.
- **Software performance** and energy consumption are very important properties in connecting our embedded computers to the real world.
- **Understand the sources** of performance and power consumption.
- As much as possible, we want to make computing abstractions work for us as we work on the physics of our software systems.

E) Challenges in Embedded Computing System Design

- Hardware constraints
- Meeting a deadline
- Minimizing power consumption
- Design for upgradability
- Really working or not

F) Performance in Embedded Computing

- Embedded system designers, in contrast, have a very clear performance goal in mind—their program must meet its ***deadline***.
- At the heart of embedded computing is ***real-time computing***.The program receives its input data; the deadline is the time at which a computation must be finished.
- If the program does not produce the required output by the deadline, then the program does not work, even if the output that it eventually produces is functionally correct.

G.Different levels of abstraction

i.CPU

ii.Platform

iii.Program.

iv.Task

v.Multiprocessor

2. Write in detail about the operation of ARM processor [MAY/JUNE 2014] [APR 2016] Apr 17, Nov 17 Draw and explain ARM architecture in detail. Nov/Dec 18 Analyse the performance of ARM processor instruction set over CISC processor.

Subtopics

- ✓ ***Definition- ARM processor***
- ✓ ***Processor and Memory organization***
- ✓ ***Data operations***
- ✓ ***Instruction set***
  - ***i. Arithmetic instructions***
  - ***ii. Logical instructions***
  - ***iii. Shift/rotate instructions***
  - ***iv. Compare instructions***
  - ***v. Move instructions***

vi. Load/Store instructions and pseudo instructions

- ✓ ***Flow control***

ARM Definition

- ❖ ARM is actually a **family of RISC architectures** that have been developed over many years.
- ❖ **ARM does not manufacture its own VLSI devices; rather, it licenses its architecture** to companies who either manufacture the CPU itself or integrate the ARM processor into a larger system.

ARM instructions

The textual description of instructions, as opposed to their binary representation, is called an assembly language.

- ❖ **ARM instructions are written one per line**, starting after the first column.
- ❖ **Comments begin with a semicolon** and continue to the end of the line.
- ❖ **A label,** which gives a name to a memory location, comes at the beginning of the line, **starting in the first column**.
- ❖ **Example:**

LDR r0,[r8]; a comment
label ADD r4,r0,r1

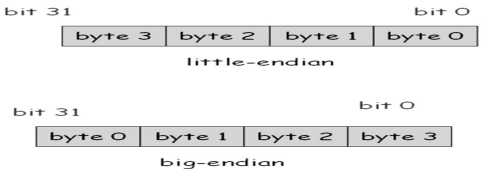
A)Processor and Memory Organization

Different versions of theARMarchitecture are identified by different numbers. **ARM7 is a von Neumann architecture machine, while ARM9 uses a Harvard architecture.**
TheARM architecture supports two basic types of data:
■ ThestandardARM word is **32 bits long.**
■ The word may be divided into four 8-bit bytes.

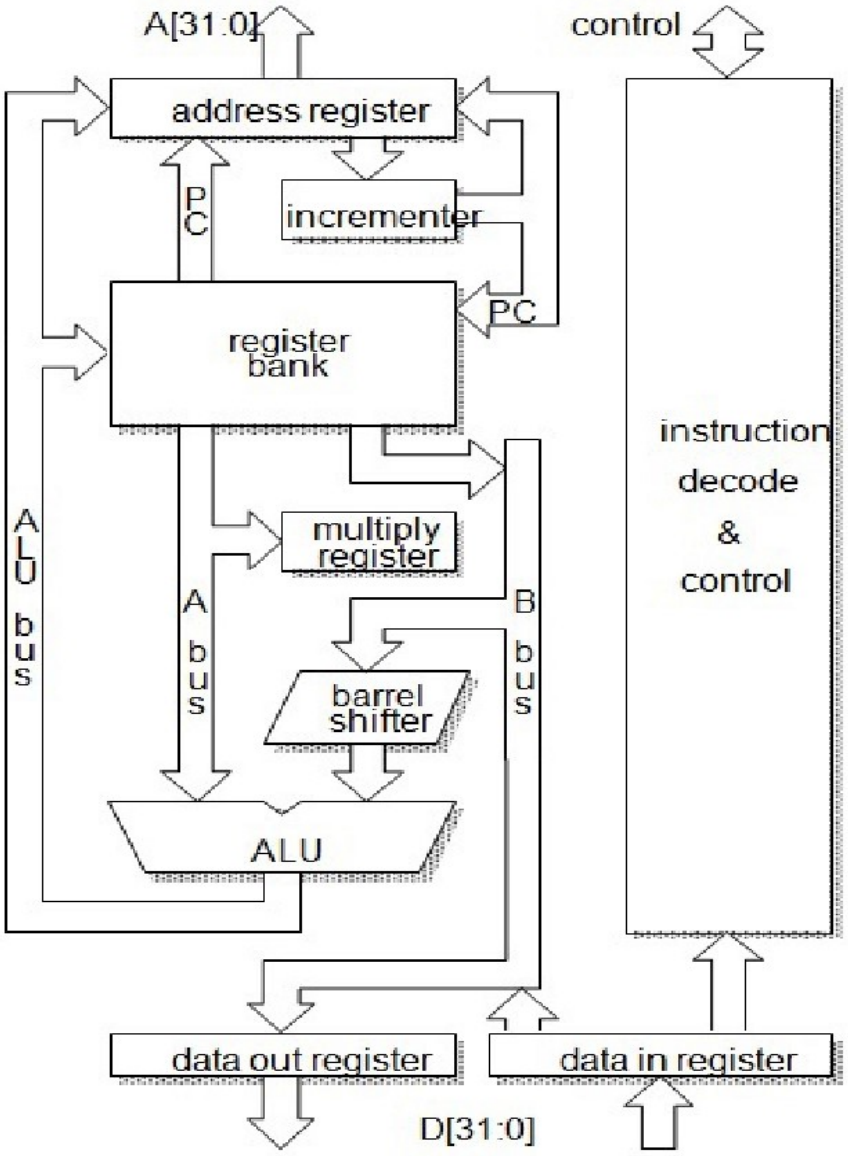
- ❖ ARM7 allows **addresses up to 32 bits long**. An address refers to a byte, not a word. Therefore, the word 0 in the ARM address space is at location 0, the word 1 is at 4, the word 2 is at 8,and so on.
- ❖ The ARM processor can be configured at power-up to address the bytes in a word in either ***little-endian*** mode (with the lowest-order byte residing in the low-order bits of the word) or ***big-endian*** mode (the lowest-order byte stored in the highest bits of the word), as illustrated in Figure .

Endianness

- Relationship between bit and byte/word ordering defines endianness:



3. **Explain in detail about ARM Architecture**



B) Data Operations

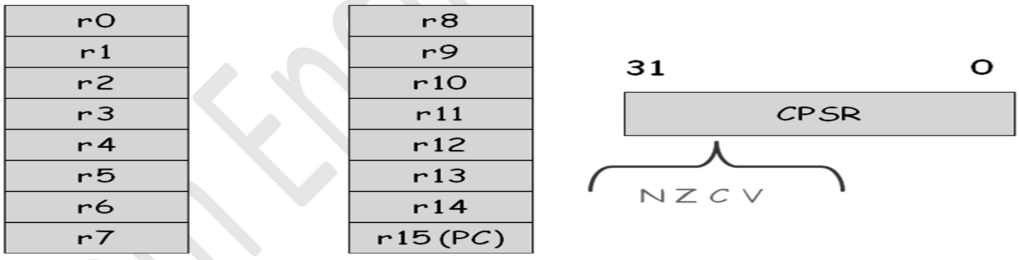
❖ In the ARM processor, arithmetic and logical operations cannot be performed

directly on memory locations. While some processors allow such operations to directly reference main memory, ARM is a ***load-store architecture***—data operands must first be loaded into the CPU and then stored back to main memory to save the results. Figure shows the registers in the basic ARM programming model.

- **ARM has 16 general-purpose registers, r0 through r15**. Except for r15, they are identical—any operation that can be done on one of them can be done on the other one also. The r15 register has the same capabilities as the other registers, but it is also used as the program counter.
- The other important basic register in the programming model is the ***currentprogram status register (CPSR)***. **This register is set automatically during every arithmetic, logical, or shifting operation**. The top four bits of the CPSR hold the following useful information about the results of that arithmetic/logical operation:

■ **The negative (N) bit** is set when the result is negative in two's-complement arithmetic.
■ **The zero (Z) bit** is set when every bit of the result is zero.
■ **The carry (C) bit** is set when there is a carry out of the operation.
■ **The overflow(V) bit** is set when an arithmetic operation results in an overflow.

ARM programming model



- The basic form of a data instruction is simple:
  - ADD r0,r1,r2
- This instruction sets register r0 to the sum of the values stored in r1 and r2.
- In addition to specifying registers as sources for operands, instructions may also provide ***immediate operands***, which encode a constant value directly in the instruction. For example,
  - ADD r0,r1,#2

Instruction sets

ARM data instructions

ADD	Add
ADC	Add with carry
SUB	Subtract
SBC	Subtract with carry
RSB	Reverse subtract
RSC	Reverse subtract with carry
MUL	Multiply
MLA	Multiply and accumulate

Arithmetic

AND	Bit-wise and
ORR	Bit-wise or
EOR	Bit-wise exclusive-or
BIC	Bit clear

Logical

LSL	Logical shift left (zero fill)
LSR	Logical shift right (zero fill)
ASL	Arithmetic shift left
ASR	Arithmetic shift right
ROR	Rotate right
RRX	Rotate right extended with C

Shift/rotate

ARM Compare Instructions

CMP	Compare
CMN	Negated compare
TST	Bit-wise test
TEQ	Bit-wise negated test

ARM Move Instructions

MOV	Move
MVN	Move negated

ARM load-store instructions and pseudo-operations.

LDR	Load
STR	Store
LDRH	Load half-word
STRH	Store half-word
LDRSH	Load half-word signed
LDRB	Load byte
STRB	Store byte
ADR	Set register to address

ARM uses ***register-indirect addressing,*** In register-indirect addressing, the value stored in the register is used as the address to be fetched from memory; **the result of that fetch is the desired operand value**. Thus, as illustrated in Figure 2.13, if we set r1 =0 x 100, the instruction

LDR r0,[r1]

sets r0 to the value of memory location 0x100.

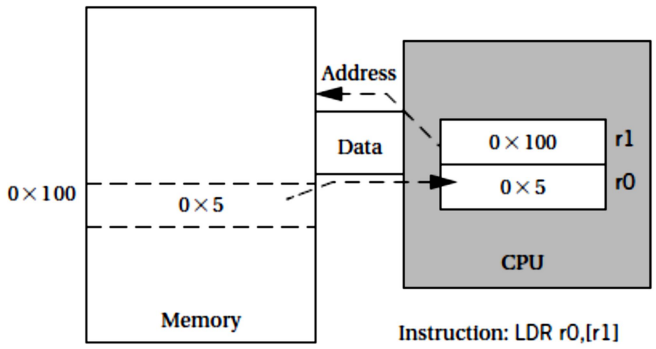


FIGURE 2.13
Register-indirect addressing in the ARM.

Flow of Control

The B (branch) instruction is the basic mechanism inARM for changing the flow of control.The address that is the destination of the branch is often called the *branchtarget.*

EQ	Equals zero	Z = 1
NE	Not equal to zero	Z = 0
CS	Carry set	C = 1
CC	Carry clear	C = 0
MI	Minus	N = 1
PL	Nonnegative (plus)	N = 0
VS	Overflow	V = 1
VC	No overflow	V = 0
HI	Unsigned higher	C = 1 and Z = 0
LS	Unsigned lower or same	C = 0 or Z = 1
GE	Signed greater than or equal	N = V
LT	Signed less than	N V
GT	Signed greater than	Z = 0 and N = V
LE	Signed less than or equal	Z = 1 or N V

FIGURE 2.15

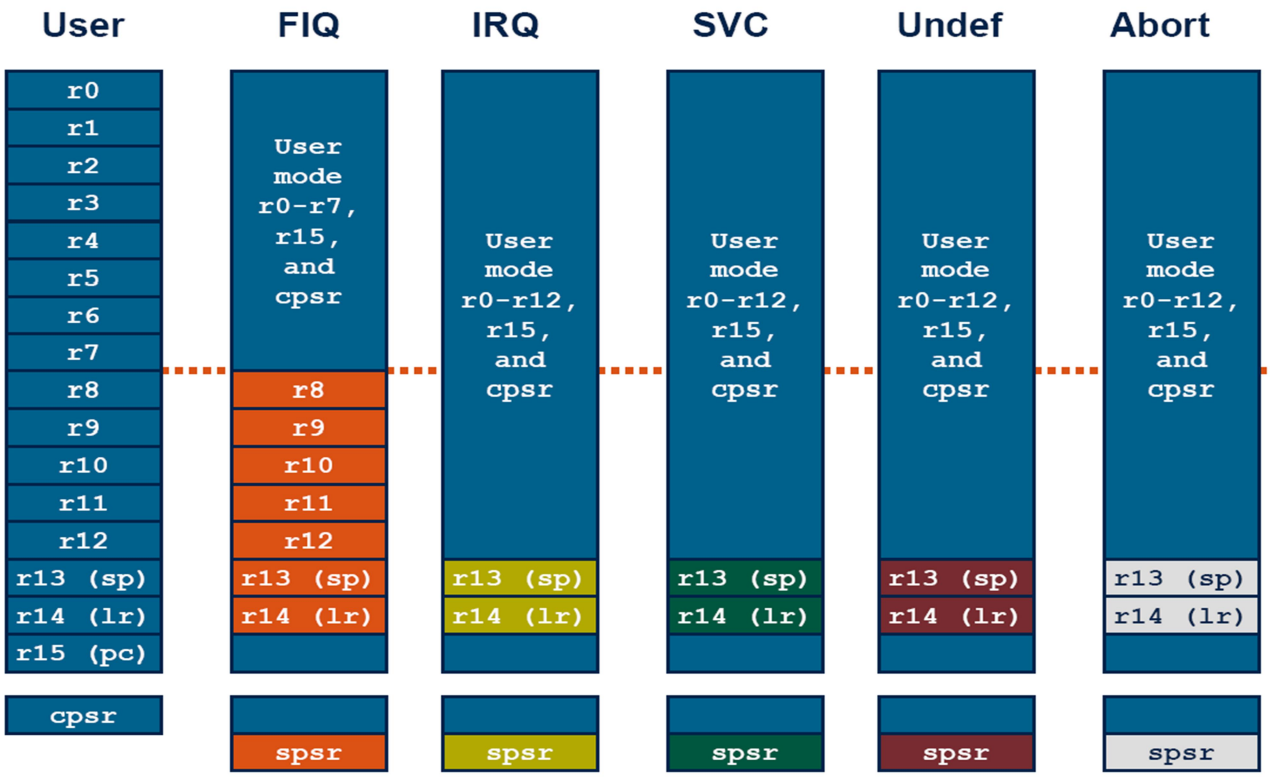
	ARM ($cpsr\ T = 0$)	Thumb ($cpsr\ T = 1$)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution[a]	most	only branch instructions
Data processing instructions	access to barrel shifter and ALU	separate barrel shifter and ALU instructions
Program status register	read-write in privileged mode	no direct access
Register usage	15 general-purpose registers +*pc*	8 general-purpose registers +7 high registers +*pc*

	Jazelle ($cpsr\ T = 0, J = 1$)
Instruction size	8-bit
Core instructions	Over 60% of the Java bytecodes are implemented in hardware; the rest of the codes are implemented in software.

4. Discuss the operation of CO-PROCESSORS with ARM processor. Nov 17

- **CPU** architects often want to provide flexibility in what features are implemented in the CPU. One way to provide such flexibility at the instruction set level is to allow ***co-processors***, **which are attached to the CPU and implement some of the instructions.**
- To support co-processors, certain opcodes must be reserved in the instruction set for co-processor operations. Because it executes instructions, a co-processor must be tightly coupled to the CPU. When the CPU receives a co-processor instruction, the CPU must activate the co-processor and pass it the relevant instruction.
- **Co-processor instructions can load and store co-processor registers or can perform internal operations**. The CPU can suspend execution to wait for the co-processor instruction to finish; it can also take a more superscalar approach and continue executing instructions while waiting for the co-processor to finish.
- A CPU may, of course, receive co-processor instructions even when there is no coprocessor attached. Most architectures use illegal instruction traps to handle these situations. The trap handler can detect the co-processor instruction and, for example, execute it in software on the main CPU. Emulating co-processor instructions in software is slower but provides compatibility.

5. Write a Neat Sketch of ARM Register Organisation



6. What are the interrupts of ARM?

Exception/interrupt	Shorthand	Address
Reset	RESET	0x00000000
Undefined instruction	UNDEF	0x00000004
Software interrupt	SWI	0x00000008
Prefetch abort	PABT	0x0000000c
Data abort	DABT	0x00000010
Reserved	—	0x00000014
Interrupt request	IRQ	0x00000018
Fast interrupt request	FIQ	0x0000001c

7.Explain in detail about UART.

- UART – Stands for Universal Asynchronous Receiver Transmitter
- USART – Stands for Universal Synchronous Asynchronous Receiver Transmitter
- **In RS-232 we implement serialport with UART**
- Actually UART receives/sends data to microprocessor/microcontroller through data bus. The remaining part of signal handling of RS-232 is done by UART i.e. start bit, stop bit, parity etc.

A UART may be usedwhen:

High speed is not required

An inexpensive communication link between two devices is required

UART communication is very cheap

Single wire for each direction (plus ground wire) Asynchronous because no clock signal is transmitted

Relatively simple hardware

Bit rate:

Number of bits sent every second (BPS)

Baud rate:

Number of symbols sent every second, where every symbol can represent more than one bit.

Ex. high-speed modems which use phase shifts to make every data transition period represent more than one bit.

For the PIC 16f877A's USART, with every clock tick one bit is sent, each symbol represents one bit.

So, we can consider bit rate and baud rate the same thing.

TRANSMISSION REQUIREMENT

- **Before transmission begins, transmitter and receiver must agree on :**

- Baud rate (75, 150, 300, 600, etc)

- **1, 1.5 or 2 stop bits**
- **5, 6, 7 or 8 data bits**
- **even, odd or no parity**
- **The LPC214x devices currently have two on-chip Universal Asynchronous Receiver Transmitter (UART).**
- **They are both identical to use, except UART1 has additional modem support. Both peripherals conform to the 550 industry standard specification.**
- **Both have a built-in baud rate generator and 16 byte transmit and receive FIFO.**

Features of UART

- **16 byte Receive and Transmit FIFO**
- **Register locations conform to 550 industry standard**
- **Receiver FIFO trigger points at 1, 4, 8, and 14 bytes**
- **Built-in fractional baud rate generator with auto-baud capabilities**
- **Mechanism that enables software and hardware flow control implementation**

BASIC BLOCK DIAGRAM OF UART

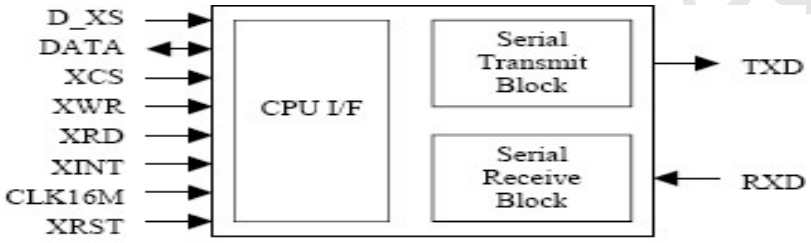
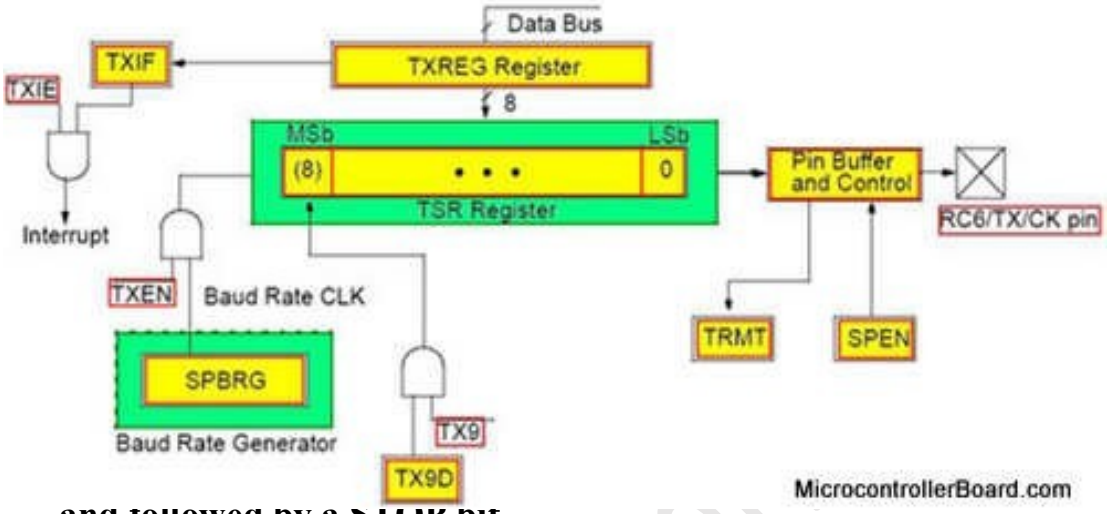


Figure 1. Basic UART block diagram.

SERIAL COMMUNICATION BLOCK DIAGRAM

8. Explain in detail about UART transmitter and receiver.

UART/USART TRANSMITTER



using 9-bit,

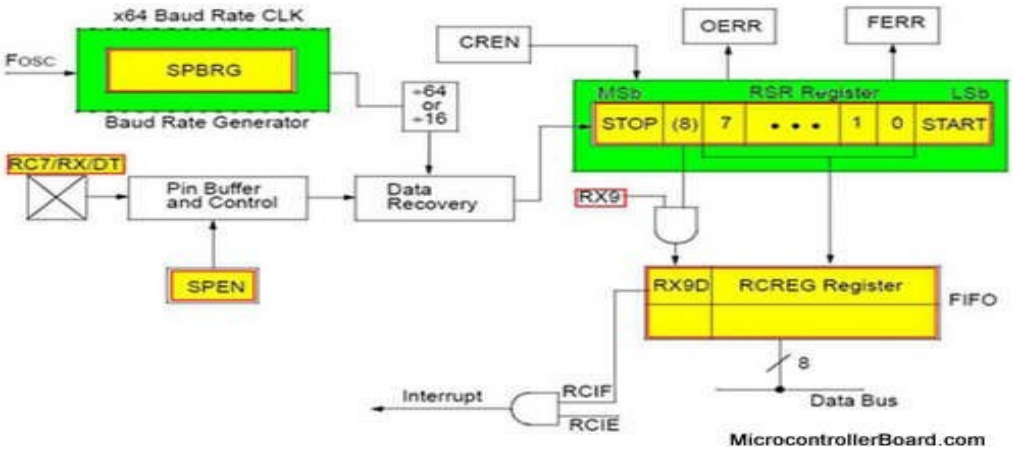
r the STOP

a START bit

and followed by a STOP bit.

- **TXIF bit : in the PIR1 register**
- **Indicates when data can be written to TXREG(when data is moved from TXREG into the Transmit Shift Register,**
- **It cannot be cleared in software. It will reset only when new data is loaded into the TXREG register.**
- **It doesn't indicate that the transmission has completed.**
- **TRMT bit:**
- **Once the data in the TSR register has been clocked out on the TX pin(at the beginning of the STOP bit), the TRMT bit in the TXSTA register will be set,**
- **Indicating that the transmission has been completed.**

UART/USART RECIEVER



- **The clock of the receiver is a multiple of the bit rate, in PIC 16f877A,it's x16 or x64. So, each bit is transmitted/received in 16 clock cycle.**
- **If the receiver detects a start bit for a period= bit period (16 clock cycles), then it waits for the period of half bit, and then sample the value on the R**
- **Every received bit is sampled at the middle of the bit's time period.**
- **The USART can be configured to receive eight or nine bits by the RX9 bit in the RCSTA register.**
- **After the detection of a START bit, eight or nine bits of serial data are shifted from the RX pin into the Receive Shift Register, one bit at a time.**
- **After the last bit has been shifted in, the STOP bit is checked and the data is moved into the FIFO buffer.**
- **RCREG is the output of the two element FIFO buffer. A next start bit can be sent immediately after the stop bit.**
- **RCIF: indicates when data is available in the RCREG.**

APPLICATIONS

- **Communication between distant computers**
- **Serializes data to be sent to modem**
- **De-serializes data received from mode**

9. Explain the features of ARM.

ARM7/ARM9 Architecture Feature Highlights

- **32/16-bit RISC architecture (ARM v4T)**
- **32-bit ARM instruction set for maximum performance and flexibility**
- **16-bit Thumb instruction set for increased code density**
- **Unified bus interface, 32-bit data bus carries both instructions and data**
- **8-, 16-, and 32-bit Data Types**
- **Three-stage pipeline**
- **4GBytes Linear Address Space**

- **32-bit ALU and high-performance multiplier**
- **37 piece of 32 bit register**
- **Very small die size and low power consumption**
- **Fully static operation**
- **Coprocessor interface**
- **Extensive debug facilities:**
  - **Embedded ICE-RT real-time debug unit.**
  - **On-chip JTAG interface unit.**

Interface for direct connection to Embedded Trace Macro cell (ETM).

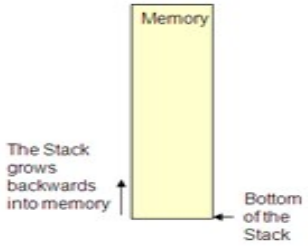
- **Pipelined (ARM7: 3 stages)**
- **Cached (depending on the implementation)**
- **Von Neuman-type bus structure (ARM7), Harvard (ARM9)**
- **7 modes of operation (usr, fiq, irq, svc, abt, sys, und)**
- **Simple structure -> reasonably good speed / power consumption ratio**
- **Very Low Power Consumption: Industry-leader in MIPS/Watt.**

10.Write a short notes on Stack and Subroutine.

Stacks & Subroutines

The Stack

- The stack is an area of memory identified by the programmer for **temporary storage** of information.
- The stack is a **LIFO (Last In First Out.)** structure.
- The stack normally **grows backwards** into memory.
- In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.



- **Given that the stack grows backwards into memory, it is customary to place the bottom of the stack at the end of memory to keep it as far away from user programs as possible.**
- **In the 8085, the stack is defined by setting the SP (Stack Pointer) register.**

LXI SP, FFFFH

- **This sets the Stack Pointer to location FFFFH (end of memory for the 8085).**

Saving Information on theStack

- **Information is saved on the stack by PUSHing it on.**
- **It is retrieved from the stack by POPing it off.**
- **The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back.**
- **Both PUSH and POP work with register pairs ONLY.**

The PUSH Instruction

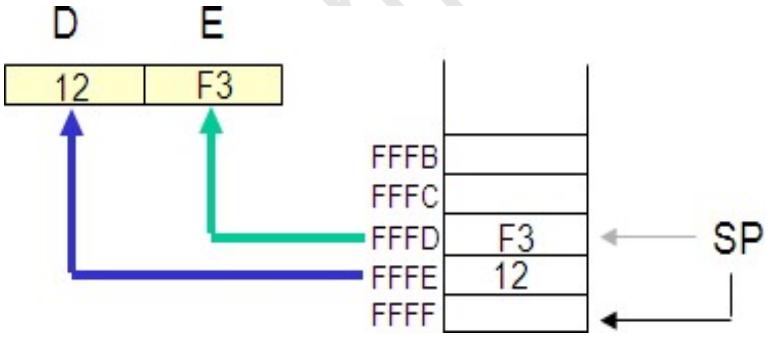
- **POP B/D/H/PSW**

Copy the contents of the memory location pointed to by the

SP to register E

Increment SP

Copy the contents of the memory location pointed to by the



Operation of the Stack

- During pushing, the stack operates in a "decrement then store" style.
- The stack pointer is decremented first, then the information is placed on the stack.
- During poping, the stack operates in a "use then increment" style.
- The information is retrieved from the top of the the stack and then the pointer is incremented.
- The SP pointer always points to "the top of the stack".

LIFO

- **The order of PUSHs and POPs must be opposite of each other in order to retrieve information back into its original location.**

PUSH B PUSH D

... POP D

POP B

- **Reversing the order of the POP instructions will result**

in the exchange of the contents of BC and DE.

Subroutines

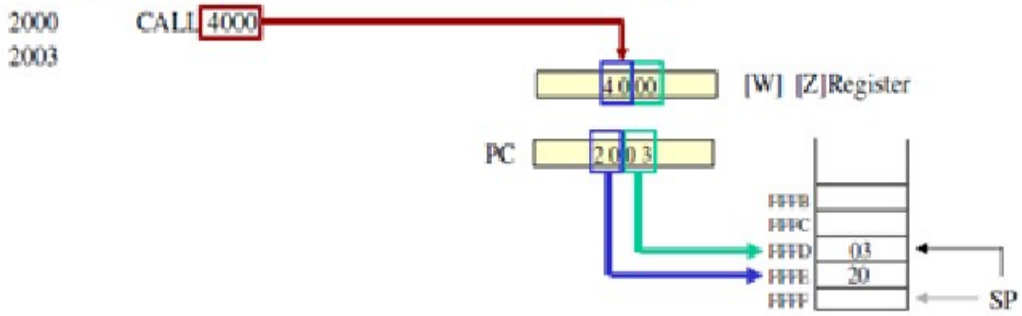
- A subroutine is a group of instructions that will be used repeatedly in different locations of the program.
- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.
- In Assembly language, a subroutine can exist anywhere in the code.
- However, it is customary to place subroutines separately from the main program.

Subroutines

- The 8085 has two instructions for dealing with subroutines.
- The **CALL** instruction is used to redirect program execution to the subroutine.
- The **RTE** instruction is used to return the execution to the calling routine.

The CALL Instruction

- **CALL 4000H**
  - 3-byte instruction.
  - Push the address of the instruction immediately following the CALL onto the stack and decrement the stack pointer register by two.
  - Load the program counter with the 16-bit address supplied with the CALL instruction.
  - Jump Unconditionally to memory location.

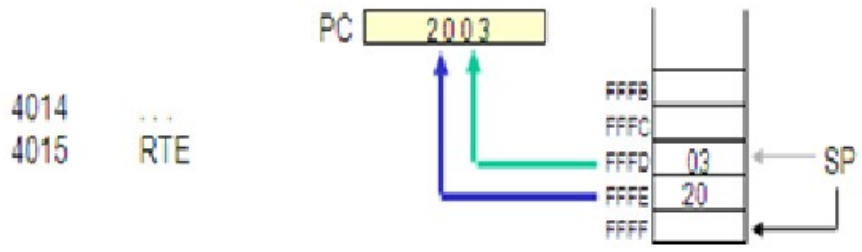


The CALL Instruction

- MP Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the W register and stores the lower order 8bit of the address in the Z register
- Push the address of the instruction immediately following the CALL onto the stack [Return address]
- Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register.

The RTE Instruction

- **RTE**
  - 1-byte instruction
  - Retrieve the return address from the top of the stack and increments stack pointer register by two.
  - Load the program counter with the return address.
  - Unconditionally returns from a subroutine.



11. Explain in detail about ARM CORTEX M3 MCU.

Definition of ARM CORTEX M3 MCU

LPC1751FBD80: 32-bit Microcontroller (MCU) based on Arm Cortex-M3 Core

- **The LPC1751 is a Cortex-M3 microcontroller for embedded applications featuring a high level of integration and low power consumption at frequencies of 100 MHz.**

Features include 32 kB of flash memory, 8 kB of data memory, USB Device, 8-channel DMA controller, 4 UARTs, 1 CAN channels, 2 SSP, 1 SPI, 2 I2C, 8-channel 12-bit ADC, motor control PWM, Quadrature Encoder interface, 4 general purpose timers, 6-output general purpose PWM, ultra-low power Real-Time Clock with separate battery supply, and up to 52 general purpose I/O pins

Features

- **Arm Cortex-M3 processor, running at frequencies of up to 100 MHz**
- **Arm Cortex-M3 built-in Nested Vectored Interrupt Controller (NVIC)**
- **Up to 32 kB on-chip flash programming memory**
- **Up to 8 kB of SRAM**
- **In-System Programming (ISP) and In-Application Programming (IAP)**
- **Eight channel General Purpose DMA controller (GPDMA)**
- **Multilayer AHB matrix interconnect provides a separate bus for each AHB master**
- **Split APB bus allows high throughput with few stalls between the CPU and DMA**
- **USB 2.0 full-speed device controller**
- **Four UARTs with fractional baud rate generation**
- **CAN 2.0B controller with one channel**
- **SPI controller with synchronous, serial, full duplex communication**
- **Two SSP controllers with FIFO and multi-protocol capabilities**
- **Two I2C-bus interfaces supporting fast mode with a data rate of 400 kbit/s**

- **Real-Time Clock (RTC)**
- **WatchDog Timer (WDT)**
- **Arm Cortex-M3 system tick timer, including an external clock input option**
- **Standard JTAG test/debug interface. Boundary scan Description Language (BSDL) is not available for this device**
- **Integrated PMU (Power Management Unit)**
- **Four reduced power modes**
- **Single 3.3 V power supply (2.4 V to 3.6 V)**
- **One external interrupt input configurable as edge/level sensitive**
- **Non-maskable Interrupt (NMI) input**
- **Wakeup Interrupt Controller (WIC)**
- **Processor wake-up from Power-down mode via any interrupt**
- **Brownout detect with separate threshold for interrupt and forced reset**
- **Power-On Reset (POR)**
- **Crystal oscillator with an operating range of 1 MHz to 25 MHz**
- **4 MHz internal RC oscillator trimmed to 1 % accuracy**
- **PLL allows CPU operation up to the maximum CPU rate**
- **USB PLL for added flexibility**
- **Code Read Protection (CRP) with different security levels**
- **Unique device serial number for identification purposes**

Applications

- **eMetering**
- **Lighting**
- **Industrial networking**
- **Alarm systems**
- **White goods and Motor control**

12. Write the features of LPC214X family.

- **The ARM is a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA) developed by ARM Holdings**
- **LPC2141/42/44/46/48 microcontrollers are based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB.**

- **The ARM is a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA) developed by ARM Holdings**
- **LPC2141/42/44/46/48 microcontrollers are based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB.**

FEATURES OF LPC2148

• ROM	• 512 KB
• RAM	• 32 KB
• IO PORTS	• 2(P0,P1)
• Timers	• 2(32 bit)
• Serial comm	• 2 UART, 2 I2C, 1 SSP ,1 SPI
• USB RAM	• 2 KB

• PWM modules	• 6
• ADC	• 2(14 channels)
• Interrupts	• 16
• Package	• 64 PIN(LQFP)
• DAC	• 1

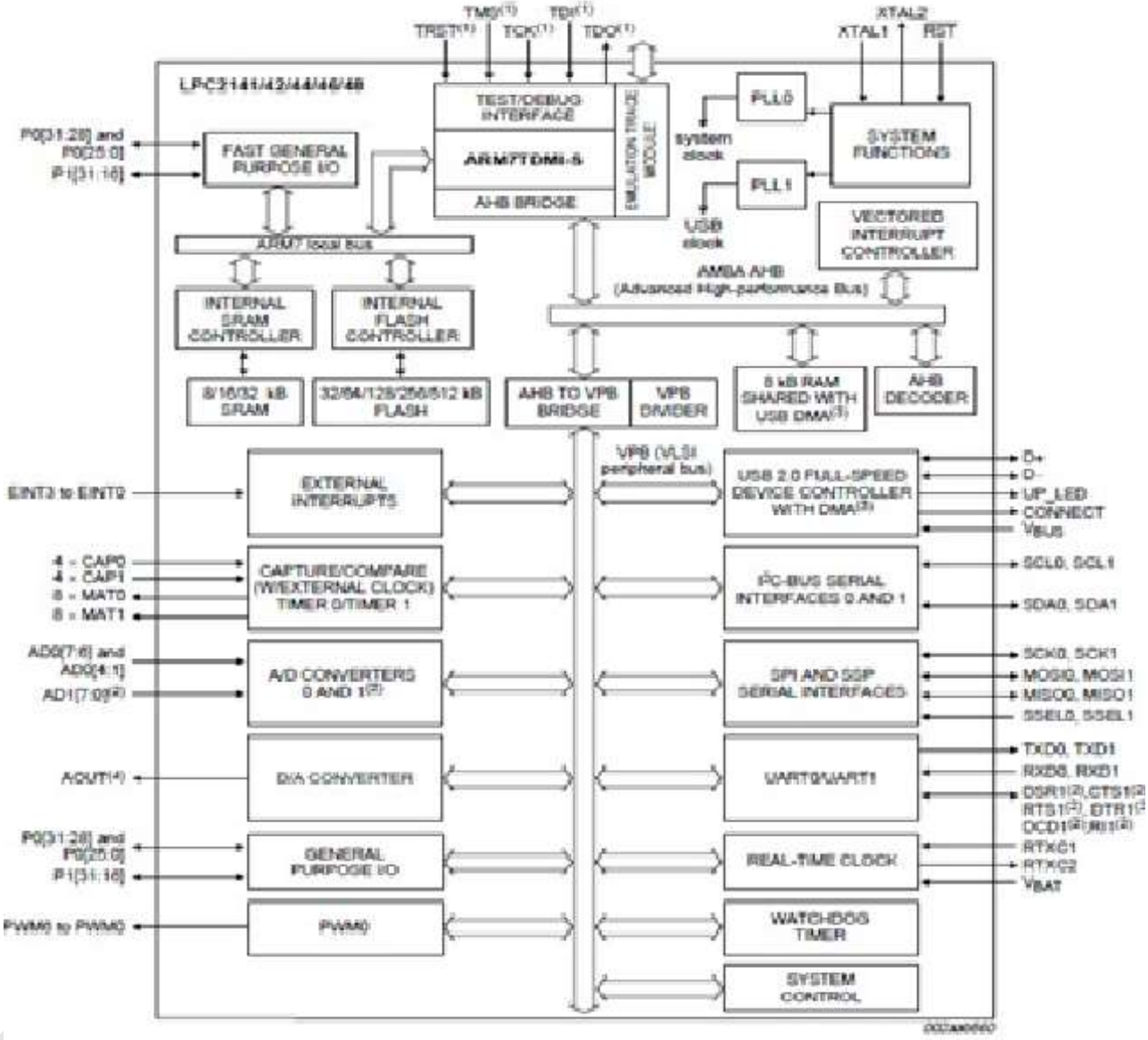
◉

APPLICATIONS

- ◉ iPod from Apple
- ◉ D-Link DSL-604+ Wireless ADSL Router
- ◉ Many automobiles embed ARM7 cores.
- ◉ Sirius Satellite Radio receivers
- ◉ Most of Nokia's mobile phone range.

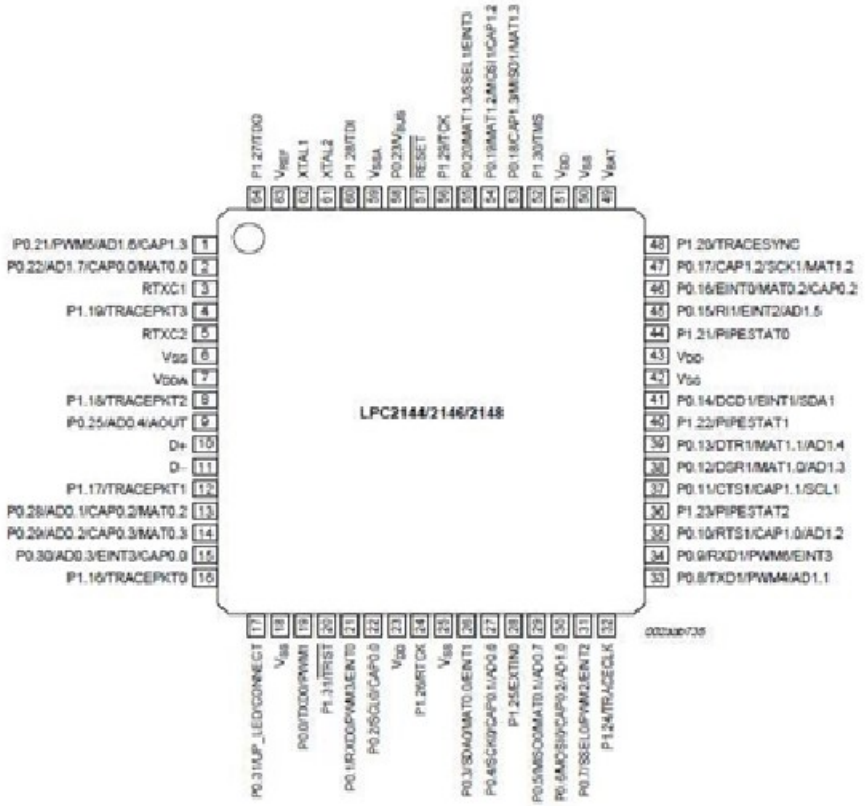
13. Explain in detail about LPC214X Family Architecture.

LPC2148 ARCHITECTURE



- The System Control Block includes several system features and control registers for a number of functions that are not related to specific peripheral devices. These include:
- Crystal Oscillator
- External Interrupt Inputs
- Miscellaneous System Controls and Status

- **Memory Mapping Control**
- **PLL**
- **Power Control**
- **Reset**
- **VPB Divider**
- **Wakeup Timer**
- **Each type of function has its own register(s) if any are required and unneeded bits are defined as reserved in order to allow future expansion.**



Lpc 2144/6//8 consists 45 GPIO functionality in is 2 port which as

1. Port0 (P0.0 to P0.31)- 24,26,27 are invisible pins, remaining 29 are visible i/o pins.
2. Port1 (P1.16 to P0.31)- 16 pins are visible and 16 pins are invisible(P1.0-P1.15)

- It consist of 19 different peripherals such as

FUNCTION	PIN	TYPE & DESCRIPTION
D+	10	INPUT/OUTPUT(USB bidirectional D+line)
D-	11	INPUT/OUTPUT(USB bidirectional D- line)
XTAL1	62	
XTAL2	61	
RTXC1	3	INPUT(Input to the RTC oscillator circuit)
RTXC2	5	OUTPUT(output to the RTC oscillator circuit)
VSS	6, 18,25,42,50	
VSSA	52	INPUT(Analog Ground: 0 V reference)
VDD	23, 43, 51	(power supply)

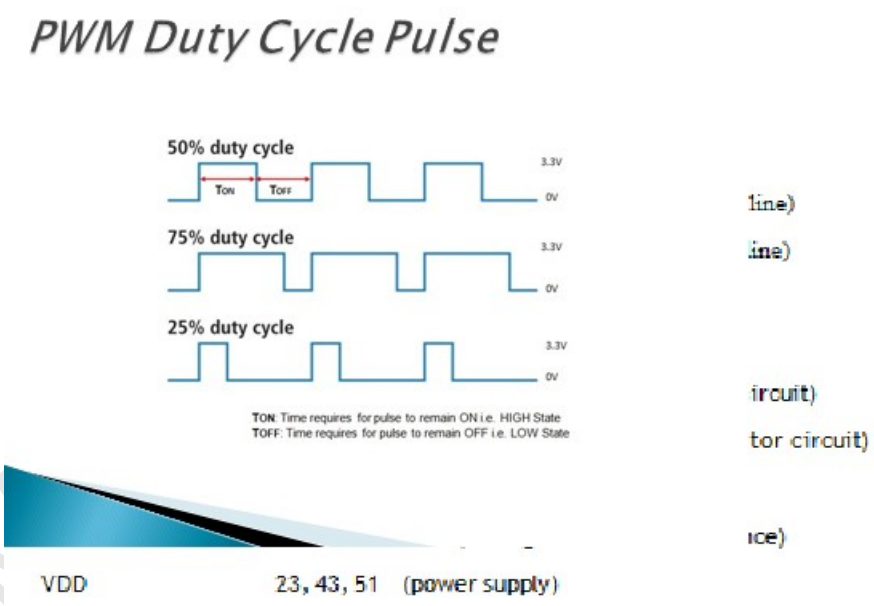
- VDDA 7 INPUT(analog power supply)
- VREF 63 INPUT(**A/D Converter Reference**)
- VBAT 49 INPUT(RTC power supply)

14. Write a Short notes on Pulse Width Modulation (PWM UNIT).

- **Pulse Width Modulation (PWM) is very useful technique for controlling analog circuits with processors digital outputs.**
- **PWM used in a wide variety of applications ranging from measurement and communications to power control.**
- **As PWM is famous technique to generate a signal of varying duty cycle.**
- **Here we will use it to control brightness of LED using LPC2148.**

Why we need PWM?

- **PWM feature allows us to generate any voltage level between 0V and 3.3V.**
- **PWM will control brightness of LED.**
- **This is proceeded to discussion about duty cycle.**



$$Duty\ Cycle = \frac{TON}{TON + TOFF}$$

$$Duty\ Cycle\ (\%) = \frac{TON}{TON + TOFF}\ X\ 100$$

TOFF: Time requires for pulse to remain OFF i.e. LOW State

tor circuit)

PWM in LPC2148 ARM7

- **PWM in LPC2148 ARM7 looks complicated than the general purpose timers.**
- **However it is really an extra general purpose timer with some additional hardware.**
- **The PWM in LPC2148 is capable of producing six channels of single edge controlled PWM or three channel of dual edge controlled PWM.**
- **The Phillips LPC2148 has 6 channels of pulse width modulation.**
- **There are 7 registers to accommodate the PWM with register 0 being used to set base frequency (f =1/T).**

Thus since there is only one base frequency register all 6 channels must have base frequency

Registers: PWM in LPC2148 ARM7

PWMTCR PWM Timer Control Register– PWMTCR is used to control the timer counter functions. The Timer Counter can be disable or reset through the PWMTCR.

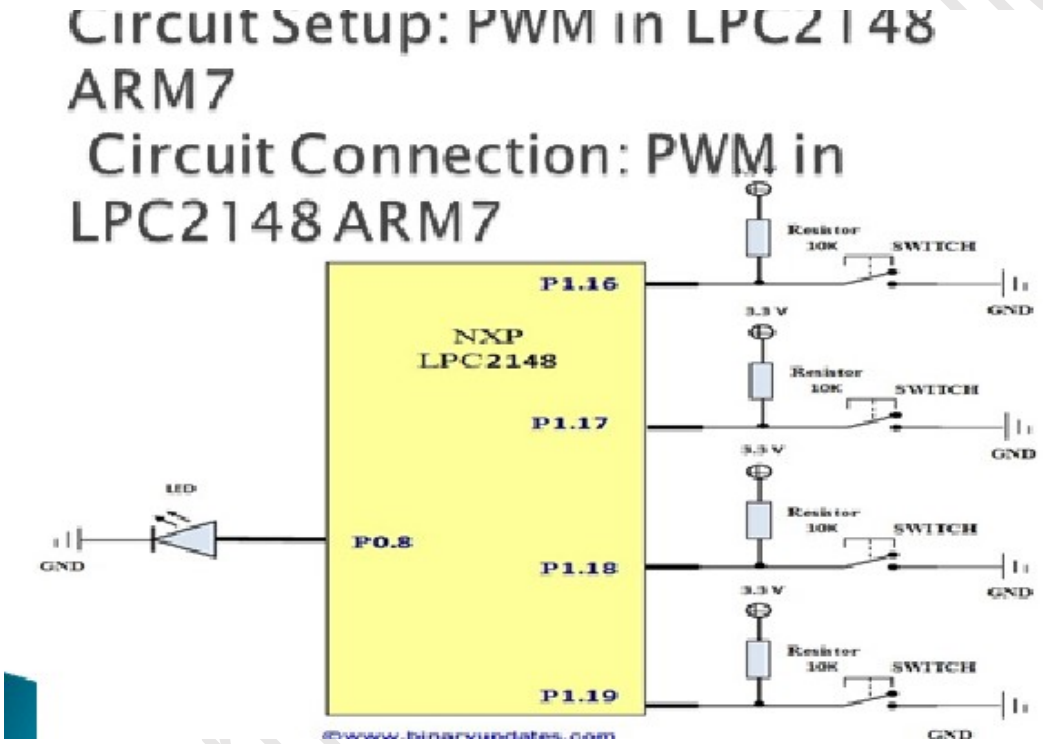
PWMPR PWM Prescale Register– The PWMTC (PWM Timer Counter) is incremented every PWMPR+1 cycles of PCLK.

PWMMR0-PWMMR6 PWM Match Register 0- PWM Match Register 6– PWMMR0-6 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC.

In addition, a match between PWMMR0-PWMMR6 and the PWMTC sets all PWM outputs that are single edge mode and sets PWM1 if it is in double-edge mode.

- **PWMMCR PWM Match Control Register– The PWMMCR is used to control if an interrupt is generated and if the PWMTC is reset when match occurs.**

PWMIR PWM Interrupt Register– The PWMIR can be written to clear interrupt. The PWMIR can be read to identify which of the possible interrupt sources are pending



15. Write a short notes on Timer Counter unit in ARM7(LPC2148)?

<u>What is Timer ?</u>

• Timer is fully depend upon the oscillator that attached externally to the microcontroller because it uses the frequency of oscillator to operate.

• When we trigger timer it start from initial value and run up to decided value stored by user in special function registers. When it reach its maximum value, it overflows and a certain bit is decided to show over flow in SFR(special function register) also called flag bit.

- **Some timers also used prescalar technique to enhance its limits.**
- **Suppose a timer is of 8 bit so it can achieved up to = 256 (2^8)**
- **for 16 bit so it can achieved up to = 65536 (2^16).**
- **For 32 bit so it can achieved up to = 2^32.**

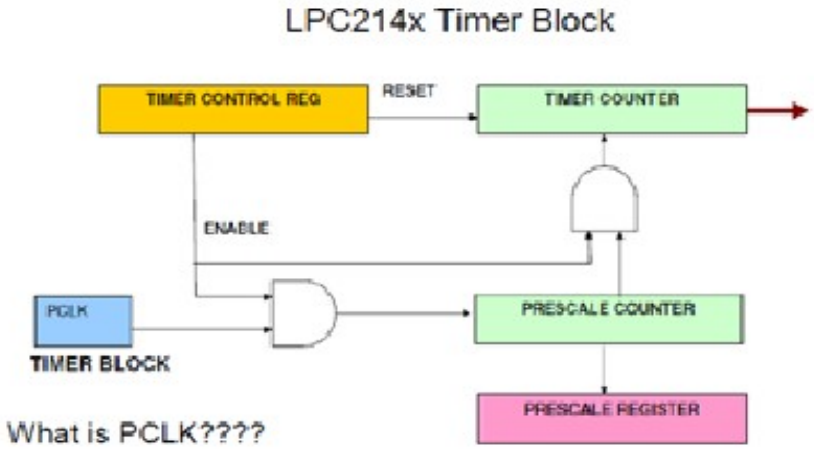
- **The LPC2148 has two functionally identical general purpose timers: Timer0 and Timer1.**
- **• Both Timer/Counter with a programmable 32-bit Prescaler.**
- **• Counter or Timer operation.**
- **• Up to four 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.**
- **• Four 32-bit match registers that allow:**
- **– Continuous operation with optional interrupt generation on match.**
- **– Stop timer on match with optional interrupt generation.**

  – Reset timer on match with optional interrupt
- **Up to four external outputs corresponding to match registers, with the following capabilities:**
- **– Set low on match.**
- **– Set high on match.**
- **– Toggle on match.**
- **– Do nothing on match.**
- **Capture Register: As the name suggests it is used to Capture Input signal.**
- **When a transition event occurs on a Capture pin , it can be used to copy the value of TC into any of the 4 Capture Register or to genreate an Interrupt.**
- **Hence these can be also used to demodulated PWM signals.**

How Timers in LPC2148 ARM7 Microcontroller Works?

- **The heart of timers of the LPC2148 Microcontroller is a 32-bit free running counter, which is designed to count cycles of the Peripheral Clock (PCLK) or an external clock, this counter is programmable with 32-bit prescaler.**

- **The tick rate of the Timer Counter (TC) is controlled by the 32-bit number written in the Prescaler Register (PR) in the following way.**
- **There is a Prescale Counter (PC) which increments on each tick of the PCLK.**
- **When it reaches the value in the Prescaler register, the timer count is incremented and the Prescaler Counter (PC) is reset, on the next PCLK.**
- **This cause the timer counters to increment on every PCLK when PR=0, every 2 PCLKs when PR=1, etc.**



- The Prescale Counter is incremented on every PCLK.
- When Prescale Counter reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK

The Peripherals

- **Peripherals connect to the computer input/output (I/O) ports**
- **They provide input or receiving its output**
- **They are not considered part of the computer:**
  - **They are only specialized gadgets that encode or decode information between the computer and the physical world**
- **The keyboard encodes our keystrokes into binary form for the computer**
- **The monitor decodes information from the computer's memory and displays it on a screen**
- **The peripherals handle the physical part of the operation**

Portable Memory & Hard Drives

- **Some peripherals are used by computers for both input and output:**
  - **USB memory**
  - **Hard disks/drives**
- **They are storage devices**
- **The hard disk is the *alpha-peripheral,* being the most tightly linked device to the computer**
- **Hard Disk**
- **Hard disk is essential**
- **Programs and their data *must* reside in the computer's memory when programs run**
- **The hard disk can be seen as an extension of the computer's memory**
- **Typically it is a hundred times larger and several thousand times slower**
- **Most peripheral devices are "dumb"**
  - **They provide only basic physical translation to or from binary signals.**
- **Additional information from the computer is needed to make it operate "intelligently"**

- **Added processing by software called a device driver gives the peripheral its standard meaning and behavior**
- **Every device needs a device driver**