# ARM Architecture

## ARM Architecture Versions

- ARM means Advanced RISC Machines. ARM machines have a 32-bit Reduced Instruction Set Computer (RISC) Load Store Architecture. It is first RISC microprocessor for commercial use and market-leader for low- power and cost-sensitive embedded applications.
- The processor originated in England in 1984. At its inception ARM stood for Acorn RISC Machine. The first ARM reliant systems include the Acorn

  : BBC Micro, Masters and the Archimedes. During this early period they were used mostly for British educational systems and therefore, were not widely available or known outside England. However in 1987 the ARM became the first commercial RISC processor.
- The ARM is a Von Neumann, load/store architecture i.e. only 32-bit data bus for both instruction and data. Also for the load/store instruction accessmemory.
- Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers. ARM does not fabricate silicon itself.
- First models had only a 26-bit program counter, limiting the memory space to 64 MB.
- In 1990, the research section of Acorn separated from the parent companyand formed : Advanced RISC Machines Limited.
- The ARM is a 32-bit architecture. When used in relation to the ARM :
  1. Byte means 8 bits.
- Most ARM's implement two instruction sets

1. 32-bit ARM Instruction Set
2. 16-bit Thumb Instruction Set

- Memory is addressed as a 32 bit address space. Data type can be 8 bit bytes, 16 bit half-words or 32 bit words and may be seen as a byte line folded into 4-byte words.
- ARM1 processor was the first commercialized RiSC processor and it contained 25,000 transistors.

| ARM processor | Features |
|---|---|
| ARM1 | • First version of ARM processor.<br>• 26-bit addressing, no multiply / coprocessor. |
| ARM2 | • ARM2, First commercial chip.<br>• Included 32-bit result multiply instructions/coprocessor support. |
| ARM2a | • ARM3 chip with on-chip cache.<br>• Added load and store.<br>• Cache management. |
| ARM3 | • ARM6, 32 bit addressing, virtual.<br>• Memory support. |
| ARM7 | • Most popular used today.<br>• Suitable for DSP work. |
| ARM8 | • It Includes a five stage pipeline.<br>• Speculative instruction fetcher.<br>• Processor to allow a higher clock speed. |
| ARM9 | • Uses five stage pipeline.<br>• Support Harvard Architecture chip. |



**ARM Features**

1. Thumb Set designed for 16-bit word lengths and instructions, which internally executes by same 32-bit core.
2. ARM views memory as a linear collection of bytes numbered upwards from zero. it contains memory management unit and memory protection unit.

3. Most operations are executed over registers.
4. All instructions can be conditional.
5. it uses Big-endian and Little-endian method.
6. The ARM processor supports 25 different instruction.
7. ARM provides no explicit return instruction.
8. The Software Interrupt (SWI) instruction is the only way an ARMprocessor can access resources controlled by the operating system.
9. Many Thumb data processing instructions use a 2-address format.
10. Jazelle Instruction Set : Introduces technological infrastructure forrunning Java code.
11. The ARM architecture has a large variety of addressing modes

**ARM Architecture**

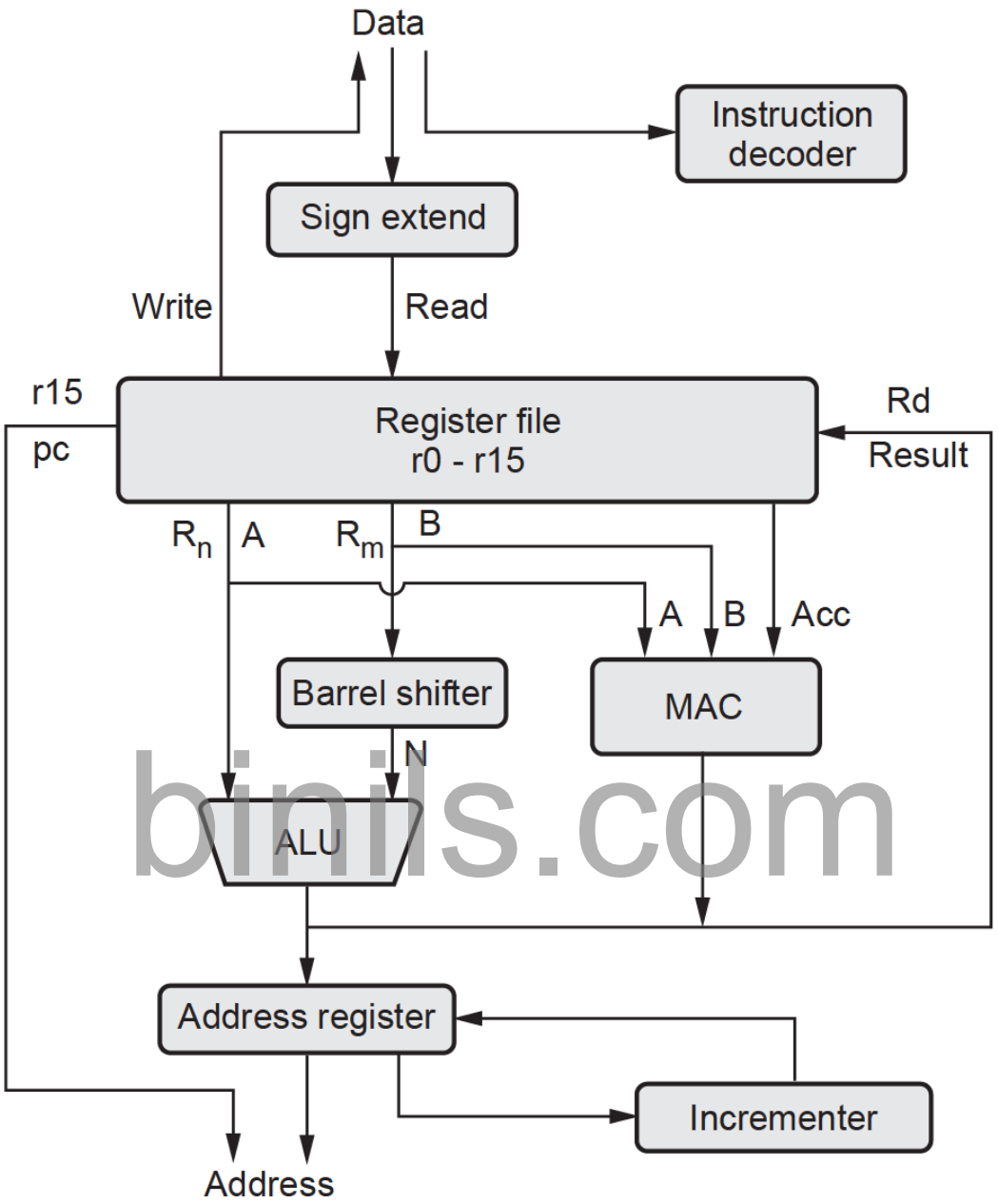
- The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32 bit RISC microcontroller.
- The ARM cortex is a complicated microcontroller within the ARM family that has ARMv7 design. There are 3 subfamilies within the ARM cortex family:

a) ARM Cortex Ax-series
b) ARM Cortex Rx-series
c) ARM Cortex Mx-series

- The ARM Architecture consists of following:

a) Arithmetic Logic Unit
b) Booth multiplier
c) Barrel shifter
d) Control unit
e) Register file

**ARM Architecture**




**ARM Architecture**

- The ARM processor conjointly has other components like the Program status register
- The modes bits conjointly exist within the program standing register, in addition to the interrupt and quick interrupt disable bits; Some special registers: Some registers are used like the instruction, memory data read and write registers and memory address register.

1. Priority encoder : The encoder is used in the multiple load and store instruction to point which register within the register file to be loaded or kept.
2. Multiplexers : Several multiplexers are accustomed to the management operation of the processor buses.
3. Arithmetic Logic Unit (ALU) : The ALU has two 32-bits inputs. The primary comes from the register file, whereas the other comes from the shifter. Status registers flags modified by the ALU outputs. The V-bit output goes to the V flag as well as the Count goes to the C flag. Whereas the foremost significant bit really represents the S flag, the ALU output operation is done by NORed to get the Z flag. The ALU has a 4-bit function bus that permits up to 16 opcode to be implemented.
4. Booth multiplier factor : The multiplier factor has 3 32-bit inputs and the inputs return from the register file. The multiplier output is barely 32-Least Significant Bits of the merchandise. The entity representation of the multiplier factor is shown in the above block diagram. The multiplication starts whenever the beginning 04 input goes active. Fin of the output goeshigh when finishing.
5. Barrel shifter : The barrel shifter features a 32-bit input to be shifted. This input is coming back from the register file or it might be immediate data. The shifter has different control inputs coming back from the instruction register. The Shift field within the instruction controls the operation of the barrel shifter. This field indicates the kind of shift to be performed.
6. Control unit : The control unit is sometimes a pure combinational circuit design. Here, the control unit is implemented by easy state machine. The processor timing is additionally included within the control unit. Signals

from the control unit are connected to each component within the processorto supervise its operation.

7. Incremented :

   For load and store instructions, the incremented updates the contents ofthe address register before the processor core reads or writes the next register value from or to the consecutive memory location.

   - The processor core continues the execution of instruction. Only when an exception or interrupt occurs, the normal execution flow is changed.

8. Address Register : This holds the address generated by the load and storeinstructions and places it on the address bus.
9. Instruction decoder : It decodes the instruction opcode read from thememory and then the instruction is executed.
10. Register file : This is a bank of 32-bit registers used for storing data items.

## Instruction Set

Instruction set defines the operations that can change the state. ARM instructions are all 32bit long are all 32-bit long (except for Thumb mode) Thereare 232 possible machine instructions.

**Features of ARM instruction set**

1. Load-store architecture
2. 3-address instructions
3. Conditional execution of every instruction
4. Possible to load/store multiple registers at once
5. Possible to combine shift and ALU operations in a single instruction

**Data Processing Instruction**

Data processing instructions are move, arithmetic, logical, comparison and multiply instructions. The load / store instruction only work on registers, NOT memory. They each perform a specific operation on one or two operands.



| Sr. No. | Instruction type | Instructions |
|---|---|---|
| 1 | Arithmetic | ADD, ADC, SUB, SBC, RSB, RSC |
| 2 | Logical | AND, ORR, EOR, BIC |
| 3 | Comparisons | CMP, CMN, TST, TEQ |
| 4 | Data movement | MOV, MVN |

**MOV and MVN Instruction**

MOV instruction move the data from one register to another register.

Format is as follows :

**MOV destination_reg , source_reg**

The "MVN" stands for "MOVE Negated". Toggling is done by using MVN instruction. Toggling means switching between 0 and 1. The easiest type of

toggling is just switching all the bits in a register. This is easily done with a MVNinstruction.

**The syntax of MVN instruction :**

**MVN (first_register),(second_register)**

| | |
|---|---|
| r2 : | 0101 0011 1010 1111 1101 1010 0110 1011 |
| r0 : | 1010 1100 0101 0000 0010 0101 1001 0100 |

**Example : MVN rO, r2**

Move the complemented value of r2 to rO

All the bits in the second_register will switch (1 to 0 or 0 to 1) and then the resultwill be stuck in the first_register.

**Example :**

```
MOV R5, R7      ; copy content of R7 to
R5MOV R9, R3    ; copy content of R3 to
R9
MVN Rl, R4      ; move the complemented value of R4 to R1
```



**Shift and Rotate**

Logical and arithmetic are the two shift types. There are six mnemonics forthe different shift types :

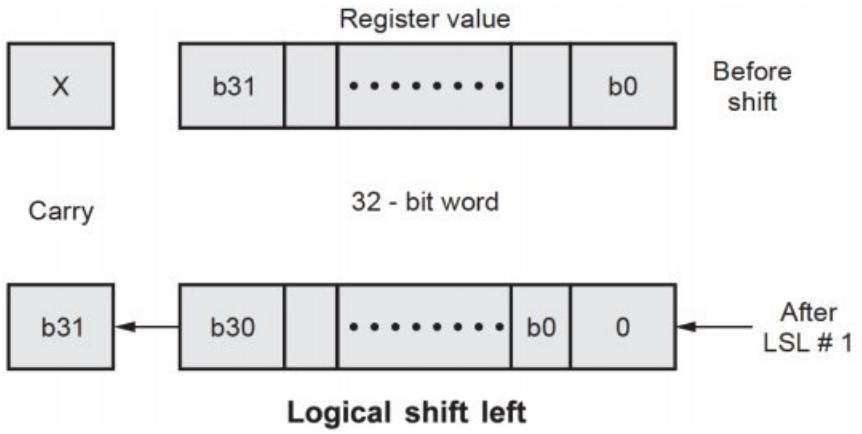
a) Logical Shift Left (LSL)
b) Arithmetic Shift Left (ASL)
c) Logical Shift Right (LSR)
d) Arithmetic Shift Right (ASR)
e) Rotate Right (ROR)
f) Rotate Right with Extend (RRX)

**Logical Shift Left**

Shifts left by the specified amount. LSL means "logical shift left by thespecified number of bits." This instruction is executed in a single clock cycle.

Example : LSL # n

Here n is the number of bit positions by which the value is shifted. Shifting left by n-bit on a signed or unsigned binary number has the effect of multiplyingit by 2n.



Logical shift left

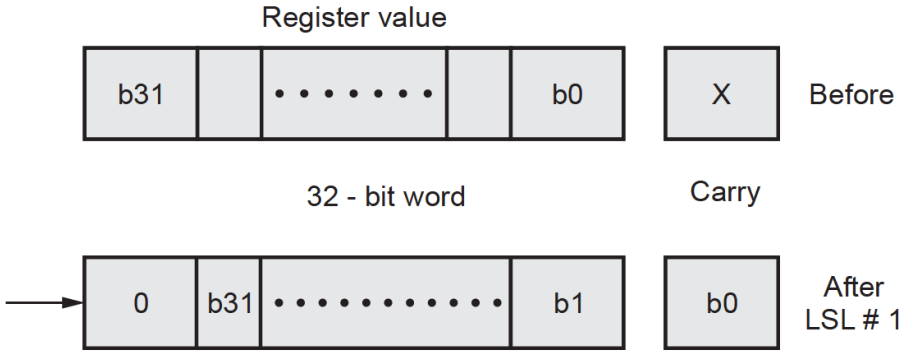
**Logical Shift Right (LSR)**

LSR by 0 to 32 places, fill the vacated bits at the most significant end ofthe word with zeros.

Logical shifts can be useful as efficient ways of performing multiplicationor division of unsigned integers by powers of two.



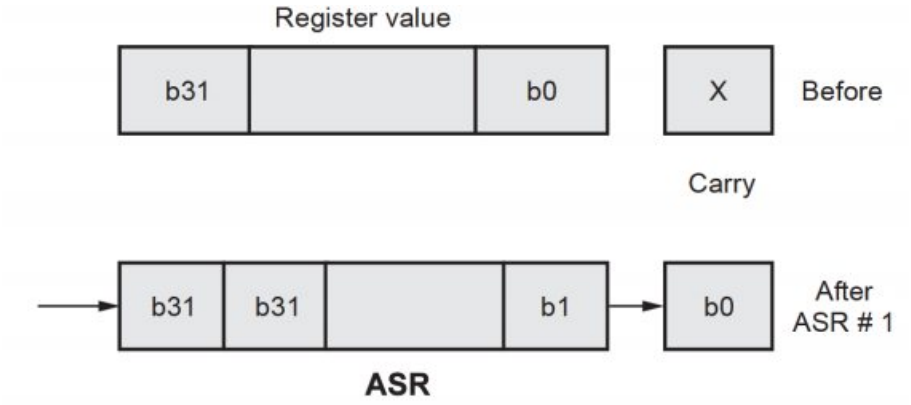
Shifting right by n-bit on an unsigned binary number has the effect ofdividing it by $2^n$.




**Arithmetic Shift Left and Right**

Arithmetic Shift Left (ASL) is same as logical shift left.

Arithmetic Shift Right (ASR) by 0 to 32 places, fill the vacated bits at the most significant end of the word with zeros if the source operand was positive and with ones it is negative.

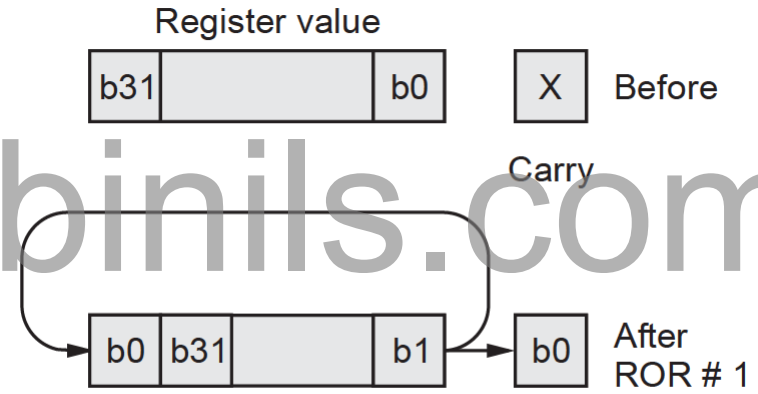



**ASR**

**Rotate Right (ROR)**

Rotate right by 0 to 32 places. The last bit rotated out is available in thecarry flag. Rotate left instruction is not used in ARM processor.

Rotate left by "n" positions is the same as a rotate right by (32 - n).

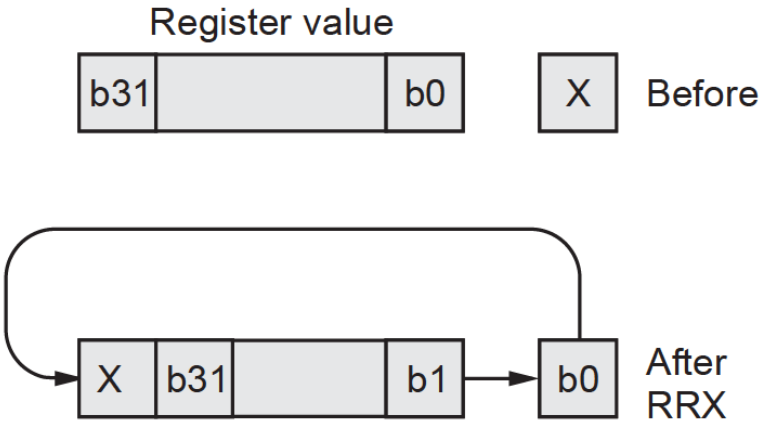





**ROR**

**Rotate Right Extended (RRX)**

It is always rotate by one bit only.




**RRX**

The vacated bit is filled with the old value of the C flag and the operand isshifted one place to the right. This operation uses the CSPR C flag as a 33rd bit. **Conditional Code Instruction**

ARM processor supports for conditional execution. The instruction is executed only when condition is true. Any data processing instruction is used forthis purpose.

Most instruction sets only allow branches to be executed conditionally. However by reusing the condition evaluation hardware, ARM effectively increase number of instruction. All instructions contain a condition field which determines whether the CPU will execute them.

Bits 28 to 31 of each ARM instruction provide a condition field that defines whether the current instruction is to be executed.

| Suffix | Description | Flags tested |
|---|---|---|
| EQ | Equal | Z = 1 |
| NE | Not equal | Z = 0 |
| CS/HS | Unsigned higher or same | C = 1 |
| CC/LO | Unsigned lower | C = 0 |
| MI | Minus | N = 1 |
| PL | Positive or zero | 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 | Greater or equal | N = V |
| LT | Less than | N! = V |
| GT | Greater than | Z = 0 and N = V |
| LE | Less than or equal | Z = 1 or N = !V |
| AL | Always | |



Condition codes are simply a way of testing the ALU status flags.

Example :

| C language code | ARM code | |
|---|---|---|
| | Unconditional code | Conditional code |
| if (rO == 0) | CMP rO, #0 | CMP rO, #0 |
| { | BNE else | ADDEQ rl, rl, #1 |
| rl = rl + 1; | ADD rl, rl, #1 | ADDNE r2, r2, #1 |
| } | B end | |
| else | else | |
| { | ADD r2, r2, #1 | |
| r2 = r2 + 1; | end | |
| } | | |

If the conditional sequence is three instructions or less, it is better to use conditional execution than a branch.

| **Condition** | if ((RO==R1) && (R2==R3)) R4++ |
|---|---|
| **Unconditional** | CMP RO, R1 BNE loopCMP R2, R3 BNE loop ADD R4, R4, #1 loop :... |
| **Conditional** | CMP RO, R1 CMPEQ R2, R3 ADDEQ R4, R4, #1 |



If corresponding condition is true, the instruction is executed. If the condition is false, the instruction is turned into a nop. The condition is specified by suffixing the instruction with a condition code mnemonic.

**Compare Instruction**

These four instructions set the status bits/flags (N, Z, C, V) in the PSR according to the results of their operations.

**CMP :** Compare, using subtraction

**CMN :** Compare negated, using addition

**TEQ :** Test for equality, using XOR - does not affect V flag

**TST :** Test bit(s), using AND - does not affect V flag

Comparison is done by using subtraction operation. Source and destinationvalue is not changed only conditional flags are affected.

Certain operations (TST, TEQ, CMP, CMN) do not write the result to Rd. They are used only to perform tests and to set the condition codes on the result and always have the S bit set.

**Logic :**

**AND Rd, Rn, Rm**

Performs the bit-wise logical AND of the operands in registers Rn and Rmand writes the result into register Rd.

The Bit Clear Instruction ( BIC) is closely related to the AND instruction.

The bits of Rm are complemented before they are ANDed with the bits of Rn.

If RO =

02FA62CARl =

OOOOFFFF

**BIC RO, RO, Rl**

results In 02FA0000 being written Into ROTest instruction syntax :



**TST Rn, Rm or #value**

It performs bit-wise logical AND of the two operands, then sets condition codeflags.

Example : TST R2, #1

sets Z □1 if low-order bit of R2 is

0 sets Z □ 0 if low-order bit of R2

is 1

TEQ instruction :

**TEQ Rn, Rm or #value**

It performs bit-wise logical XOR of the two operands, then sets condition codeflags.

Example : TEQ R2, #5

sets Z □ 1 if R2 contains

5sets Z □ 0 otherwise

TST and TEQ instruction does not affect V flag. The TST instruction is useful to determine if one or more bits are set and it is often used with a constantcalled a "MASK".

TEQ instruction is useful for determining if the content of two registers contains identical values.

**Multiplication Instruction**

Multiplication instruction takes more than one cycle. it also requires hardware toperform operation.

**Multiply**

Syntax : **MUL Rd, Rm, Rs**

where Rd = Destination registerRm, Rs = Source register

Example : **MUL RO, R1, R2 @ RO = R1 x R2**

Features :

1. Second operand cannot be immediate.
2. The result register must be different from the first operand.
3. if S bit is set, C flag is meaningless.

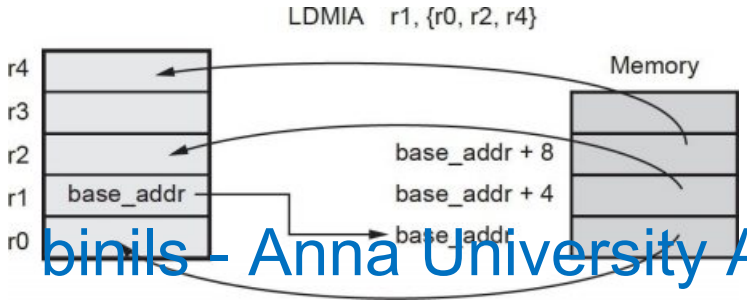


**Multiple Register Load and Store**

These instructions transfer large quantities of data more efficiently. It is used for procedure entry and exit for saving and restoring workspace registers and the return address.

**LDM Instruction**

LDR and STR instructions only load/store a single 32-bit word. ARM processor can load/store ANY subset of the 16 registers in a single instruction.



**STM Instruction**

Any registers can be specified. However, beware that if you include rl5(PC), you are effectively forcing a branch in the program flow.

The complementary instruction to LDMIA is the STMIA instruction :

**STMIA rl, {rO, r2, r4} ; mem32[rl] : = rO**
**; mem32[rl + 4] : = r2**
**; mem32[rl + 8] : = r4**

The Load and Store Multiple Instructions (LDM/STM) allow between 1 and 16 registers to be transferred to or from memory. The order of register transfer cannot be specified, order in the list is insignificant.

The lowest register number is always transferred to/from lowest memory location accessed. The transferred registers can be either :

1. Any subset of the current bank of registers (default)
2. Any subset of the user mode bank of registers when in a privileged mode (postfix instruction with a "A")



Base register used to determine where memory access should occur at four different addressing modes and base register can optionally update following thetransfer.

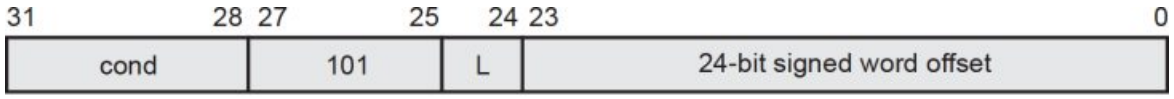
**Branch Instruction**

The branch instructions cause the processor to execute instructions from adifferent address

A simple branch or branch with link instruction :

**B{condition}**
**<address>**
**BL{condition}**
**<address>**

Bits [27:25] identify this as a B or BL instruction, they have values 101 only for these instructions.




**Branch instructions**

The top 4 bits [31:28] are used to specify the conditions under which the instruction is executed, this is common with all other instructions.

The L-bit (bit 24) is set if it is a branch with link instruction and clear if itis a plain branch. BL is jump to subroutine instruction.

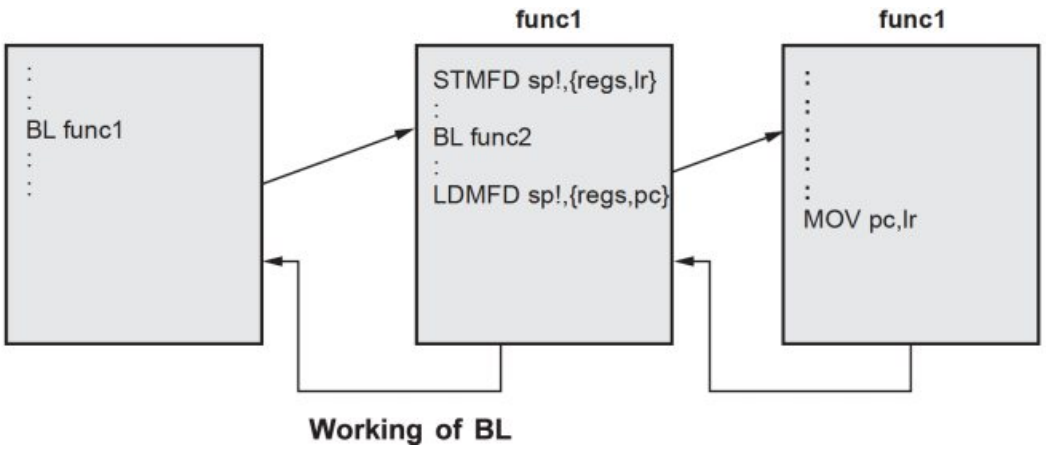
24-bit signed offset specifies destination of branch in 2's complement form. The word offset is shifted left by 2 bits to form a byte offset. This offset is addedto the PC by the processor.

The instruction can therefore specify a branch of +/- 32 Mbytes. The branch offset must take account of the prefetch operation, which causes the PC to be 2 words (8 bytes) ahead of the current instruction.



Branches beyond +/- 32 Mbytes must use an offset or absolute destinationwhich has been previously loaded into a register.

**Branch with Link (BL)** writes the old PC into the link register (R14) of the current bank. The PC value written into R14 is adjusted to allow for the prefetch, and contains the address of the instruction following the branch and link instruction. Note that the CPSR is not saved with the PC.




**Working of BL**

## Pulse Width Modulation Unit and UART

### Pulse Width Modulation Unit

Pulse Width Modulation (PWM) is a simple method of using a rectangulardigital waveform to control an analog variable

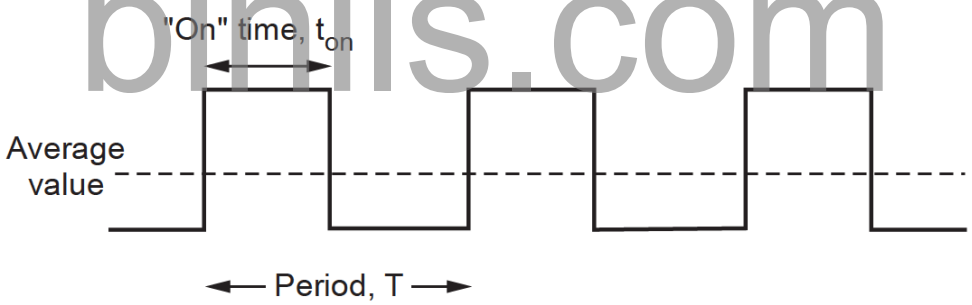
PWM control is used in a variety of applications, ranging from communications to automatic control.

The microcontrollers do everything with ones and zeros. That means microcontroller works with 3.3 V and 0 V as digital 1 & 0.

It can't produce for example 1 V or 2.5 V or any other value different than 0 V and 3.3 V. Here PWM feature allows us to generate any voltage level between0 V and 3.3 V.

The period is normally kept constant, and the pulse width, or "on" time is varied. The duty cycle is the proportion of time that the pulse is 'on' or 'high', andis expressed as a percentage.

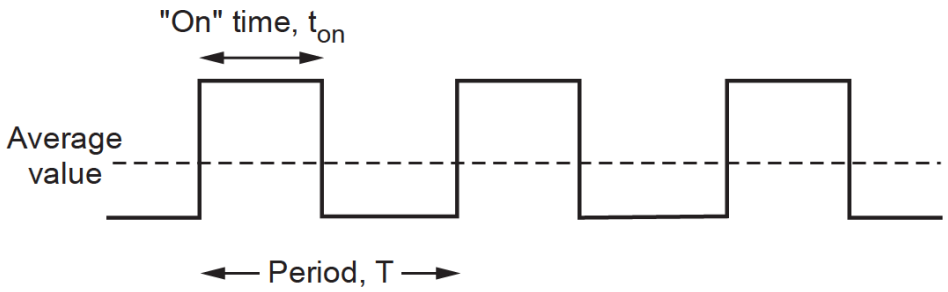





**Duty cycle = 100 % X Pulse on time / Pulse period**

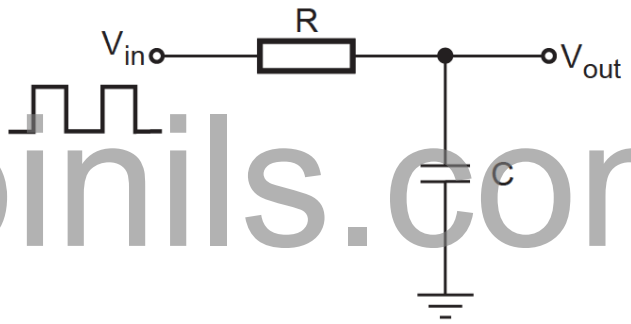
Whatever duty cycle a PWM stream has, there is an average value, as indicated by the dotted line.

If the on time is small, the average value is low; if the on time is large, the average value is high. By controlling the duty cycle, we control this average value.




The average value can be extracted from the PWM stream with a low-pass filter. In this case, and as long as PWM frequency and values of R and C are appropriately choosen, Vout becomes an analog output.

In practice, this sort of filtering is not always required; many physical systems have response characteristics which, in reality, act like low pass filter.






The PWM in LPC2148 is capable of producing six channels of single edge controlled PWM or three channel of dual edge controlled PWM.

## UART

**UART (Universal Asynchronous Receiver/Transmitter)** is the microchip with programming that controls a computer's interface to its attached serial devices.

One of the most common interfaces used in embedded systems is the Universal Asynchronous Receiver/Transmitter (UART).

It provides the computer with RS-232C Data Terminal Equipment (DTE) interface so that it can "talk" to and "exchange" data with modems and other devices.

UART peripherals typically have several configurable parameters required to support different standards. There are five parameters which must be configured correctly to establish a basic serial connection :

1. **Baud rate** : Baud rate is the number of symbols or modulations per second.
2. Number of data bits : The number of data bits transmitted is typically between 5 and 8, with 7 and 8 being the most common since an ASCII character is 7 bits for the standard set and 8 bits for the extended.
3. **Parity** : The parity can be even, odd, mark or space.
4. **Stop bits** : The number of stop bits is most commonly configurable to either one or two.
5. **Endianess** : Some UART peripherals offer the option to send the data in either LSB (least significant bit) or MSB (most significant bit). Serial communication of ASCII characters is almost always LSB.



**RS-232**

RS-232 is a point-to-point signalling standard, meaning only two devices can be connected to each other.

The minimum connection required for bidirectional communication is three signals : Transmit (TX), receive (RX) and ground.

The separate RX and TX lines mean that data can flow in both directions at the same time. This is called full-duplex and it is the standard means for communicating over serial.

- **RS-232 Error Conditions**

  Errors in the RS-232 are : Framing error, over run errors and parity errors.

**Framing error** : This error arises when the receiver does not receive the stop bit. Stop bits may change its level because of noise or signal degradation.

**Over run error** : This error occurs when new character appears at the receiver before the previous character has been handled and disposed off completely.

**Parity error** : If the parity of the character is changed, then parity error will occur. It indicates that one or more data bits are in error.

- **Advantages of RS-232 :**
  1. It is low cost interface.
  2. RS 232C can provide good performance at low cost.
  3. Control and hand shake signals ensure the communication success.
  4. Easy to use because the IC is available for RS-232C.
  5. Voltage level reduces the interference due to noise



- **Disadvantages of RS-232 :**
  1. Distance covered by RS-232C cable is only 50 ft, which is notenough in many applications.
  2. Data transfer rate is slow. Baud rate is 20 k baud for less than 50 ft.
  3. Multiple user can not share single RS 232C cable.
  4. RS-232C interface is less flexible, since many interconnection arrangements are not possible.
  5. Voltage levels of RS-232C interface are not compatible with many digital electric circuit and device. Additional power supply is required to convert voltage levels.
  6. RS-232C interface uses single common ground for all the signals. Hence effect of noise is maximum

## Stacks and Subroutines

Stacks are highly flexible in the ARM architecture. In the ARM processor, any one of the general purpose registers could be used as a stack pointer.

**Stack instructions** : The ARM instruction set does not contain any stack specific instructions like push and pop. The instruction set also does not enforce in anyway the use of a stack. Push and pop operations are performed by memory access instructions, with auto-increment addressing modes.

**Stack pointer** : The stack pointer is a register that points to the top of the stack. In the ARM processor, there are no dedicated stack pointer registers, and any one of the general purpose registers can be used as the stack pointer.

**Stack types** : Since it is left to the software to implement a stack, different implementation choices result different types of stacks. There are two types of stack depending on how the stack grows.



**Ascending stack** : In a push the stack pointer is incremented, i.e the stackgrows towards higher address.

**Descending stack** : In a push the stack pointer is decremented, i.e the stackgrows towards lower address.

There are two types of stack depending on what the stack pointer points to.

1. **Empty stack** : Stack pointer points to the location in which the next item will be stored. A push will store the value, and increment the stack pointer.
2. **Full stack** : Stack pointer points to the location in which the last item was stored. A push will increment the stack pointer and store the value.

A subroutine is a reusable program module. A main program can call or jump to the subroutine one or more times. The stack is used in several ways when subroutines are called.

## Features of LPC 214X Family and Timer Unit

### Features of LPC 214X Family

The LPC2148 microcontroller is designed by Philips (NXP Semiconductor) with several in-built features & peripherals

The main features of LPC2148 include the following.

1. The LPC2148 is a 16 bit or 42 bit ARM7 family based microcontroller andavailable in a small LQFP64 package.
2. On-chip static RAM is 8 kB - 40 kB, on-chip flash memory is 42 kB - 512 kB, the wide interface is 128 bit, or accelerator allows 60 MHz high-speedoperation.
3. It takes 400 milliseconds time for erasing the data in full chip and 1 millisecond time for 256 bytes of programming.
4. Embedded Trace interfaces and Embedded ICE RT offers real-time debugging with high-speed tracing of instruction execution and on-chip Real Monitor software.
5. It has 2 kB of endpoint RAM and USB 2.0 full speed device controller. Furthermore, this microcontroller offers 8kB on-chip RAM nearby to USBwith DMA.
6. One or two 10-bit ADCs offer 6 or 14 analogs Inputs with low conversiontime as 2.44 |s/ channel. Only 10 bit DAC offers changeable analog output.
7. External event counter/32 bit timers - 2, PWM unit, & watchdog.
8. Low power RTC (real time clock) & 32 kHz clock input.
9. Several serial interfaces like two 16C550 UARTs, two I2C-buses with 400kbit/s speed.
10. The 5 volts tolerant quick general purpose Input/output pins in a smallLQFP64 package.
11. The incorporated oscillator on the chip will work by an exterior crystal thatranges from 1 MHz-25 MHz

12. The modes for power-conserving mainly comprise idle & power down.
13. For extra power optimization, there are individual enable or disable ofperipheral functions and peripheral CLK scaling.

**Timer Unit**

The LPC2148 has two functionally identical general-purpose timers :

**Timer0 and Timer1.**

These both timers are 32-bit along with 32-bit prescaler. Timer allows us to generate precise time delay.

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.

Timer used in LPC 2147 are as follows :



1. **Prescale Counter (PC)** : The 32-bit PC is a counter which is incremented to the value stored in PR (Prescale Register) when value in PR is reached, The TC (Timer Counter) is incremented and PC is cleared. The PC is observable and controllable through bus interface.
2. **Prescale Register** : The 32-bit register which hold the maximum value ofprescale counter after which it reset.
3. **Timer Counter (TC)** : This is 32-bit Timer Counter which gets incremented whenever PC Prescale Counter value reaches to its maximumvalue as specified in PR.
4. **Timer Control Register** : Timer Control register used to control the timercontrol functions.
5. **Count Control Register (CTCR)** : This register selects Timer Counter Mode. In our example we have used timer mode. This can be done by setting CTCR to 0x0.