Unit-2

ARM MICROCONTROLLERS

ARCHITECTURE AND PROGRAMMING

Syllabus

- Architecture, Pin Diagram, Register Set & Modes, Memory Organization, Instruction set, Programming ports, Timer/Counter, Serial communication, I/O System, Development Tools, interrupts in C, Introduction ARM mBed platform.

ARM HISTORY

- 1983 developed by Acorn computers
  - To replace 6502 in BBC computers
  - 4-man VLSI design team
  - Its simplicity comes from the inexperience team
  - Match the needs for generalized SoC for reasonable power, performance and die size
  - The first commercial RISC implemenation
- 1990 ARM (Advanced RISC Machine), owned by Acorn, Apple and VLSI

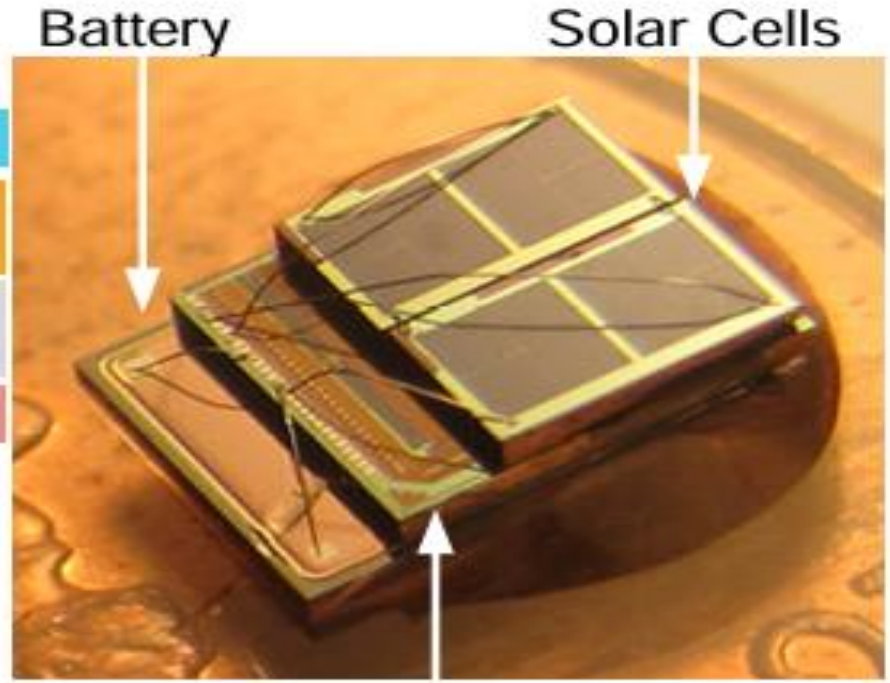
Design and license ARM core design but not fabricate

Huge Range of Applications

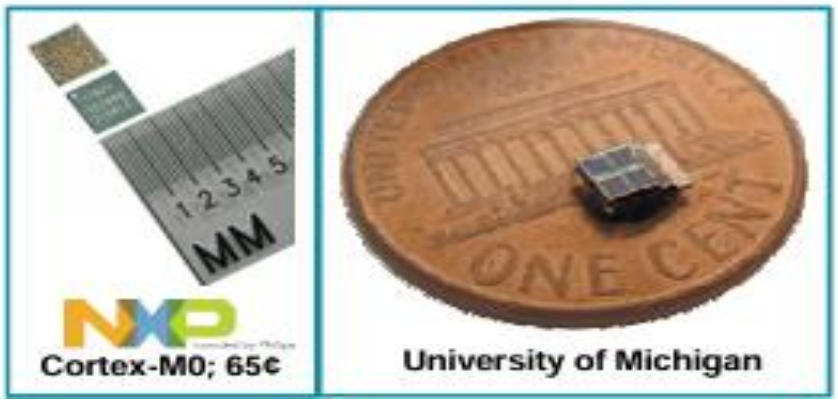
World's Smallest ARM Computer?

Wireless Sensor Network

- Sensors, timers
- Cortex-M0 +16KB RAM 65nm
  UWB Radio antenna
- 10 kB Storage memory
  ~3fW/bit
- 12µAh Li-ion Battery



Processor, SRAM and PMU



Wirelessly networked into large scale sensor arrays

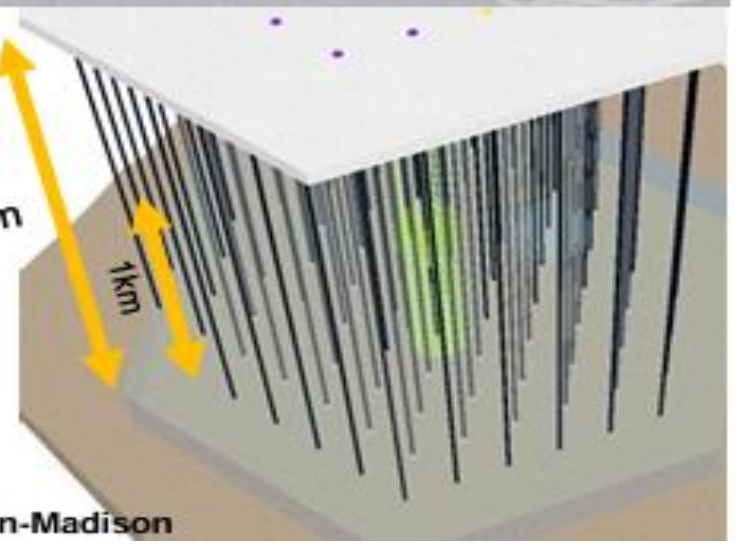
World's Largest ARM Computer?

4200 ARM powered
Neutrino Detectors

70 bore holes 2.5km deep

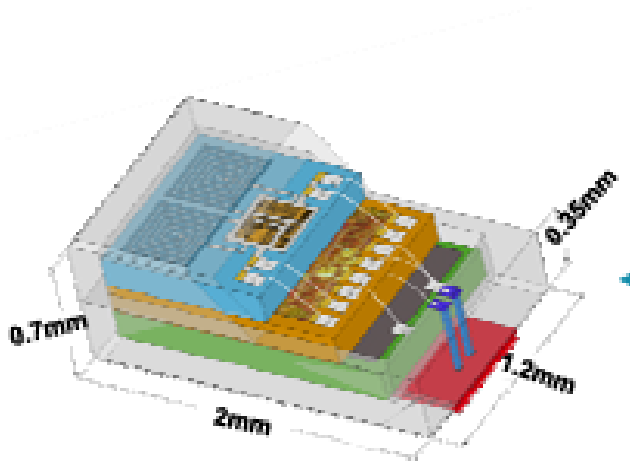
60 detectors per string
starting 1.5km down

1km^3 of active telescope

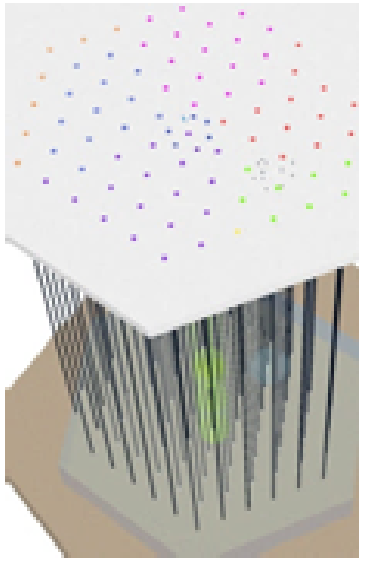


Work supported by the National Science Foundation and University of Wisconsin-Madison

From 1mm³ to 1km³



1mm³ ⟷ 1km³



10¢ ⟷ $1000

Mobile Home Mobile Computing Server

Embedded Consumer Enterprise PC HPC

ARM Architecture

- 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 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 access memory.

ARM Architecture

- In 1990, the research section of Acorn separated from the parent company and formed : Advanced RISC Machines Limited.
- The ARM is a 32-bit architecture. When used in relation to the ARM :
  - 1. Byte means 8 bits
  - 2. Halfword means 16 bits.
  - 3. Word means 32 bits.
- Most ARM's implement two instruction sets
  - 1. 32-bit ARM Instruction Set
  - 2. 16-bit Thumb Instruction Set

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

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

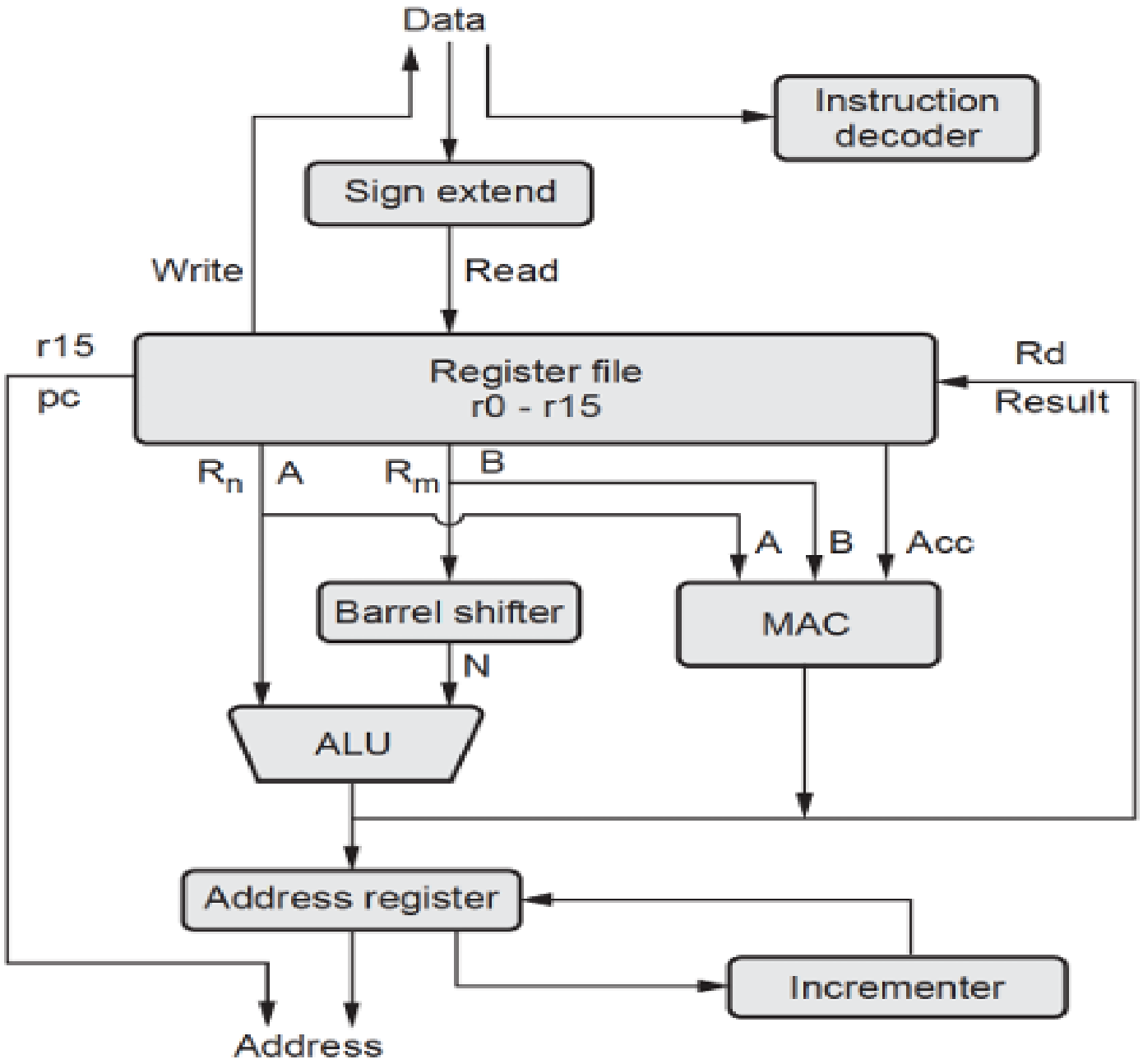
b) ARM Cortex Rx-series

c) ARM Cortex Mx-series

ARM Architecture

- 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

- 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.

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 goes high when finishing.

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.
- The quantity by which the register thought to be shifted is contained in an immediate field within the instruction or it might be the lower 6 bits of a register within the register file.

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 processor to supervise its operation.

Incremented :

• For load and store instructions, the incremented updates the contents of the 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.

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

ARM7 Based (LPC2148) Microcontroller Pin Configuration

- The LPC2148 microcontroller is designed by Philips (NXP Semiconductor) with several in-built features & peripherals.
- Due to these reasons, it will make more reliable as well as the efficient option for an application developer.
- LPC2148 is a 16-bit or 32-bit microcontroller based on ARM7 family.

Classification of ARM Pins

- Power supply pins
- Clock and reset pins
- GPIO pins
- Communication interface pins
- Analog pins
- PWM, Timer and Debug pins

Power Supply Pins

- VDD – 3.3V main supply
- VSS – Ground
- VBAT – Battery supply for RTC
- VREF – Reference voltage for ADC operation

Clock and Reset Pins

- XTAL1 and XTAL2 are used to connect external crystal
- PLL pins are used for frequency multiplication
- RESET pin initializes the system during power-up or fault

GPIO Pins

- Port 0: P0.0 to P0.31
- Port 1: P1.16 to P1.31
- Pins can be configured as input or output
- Each pin supports alternate peripheral functions

Communication Interface Pins

- UART pins are used for serial communication
- SPI pins support high-speed data transfer
- I2C pins are used for two-wire communication
- USB pins enable USB device communication

Analog Pins

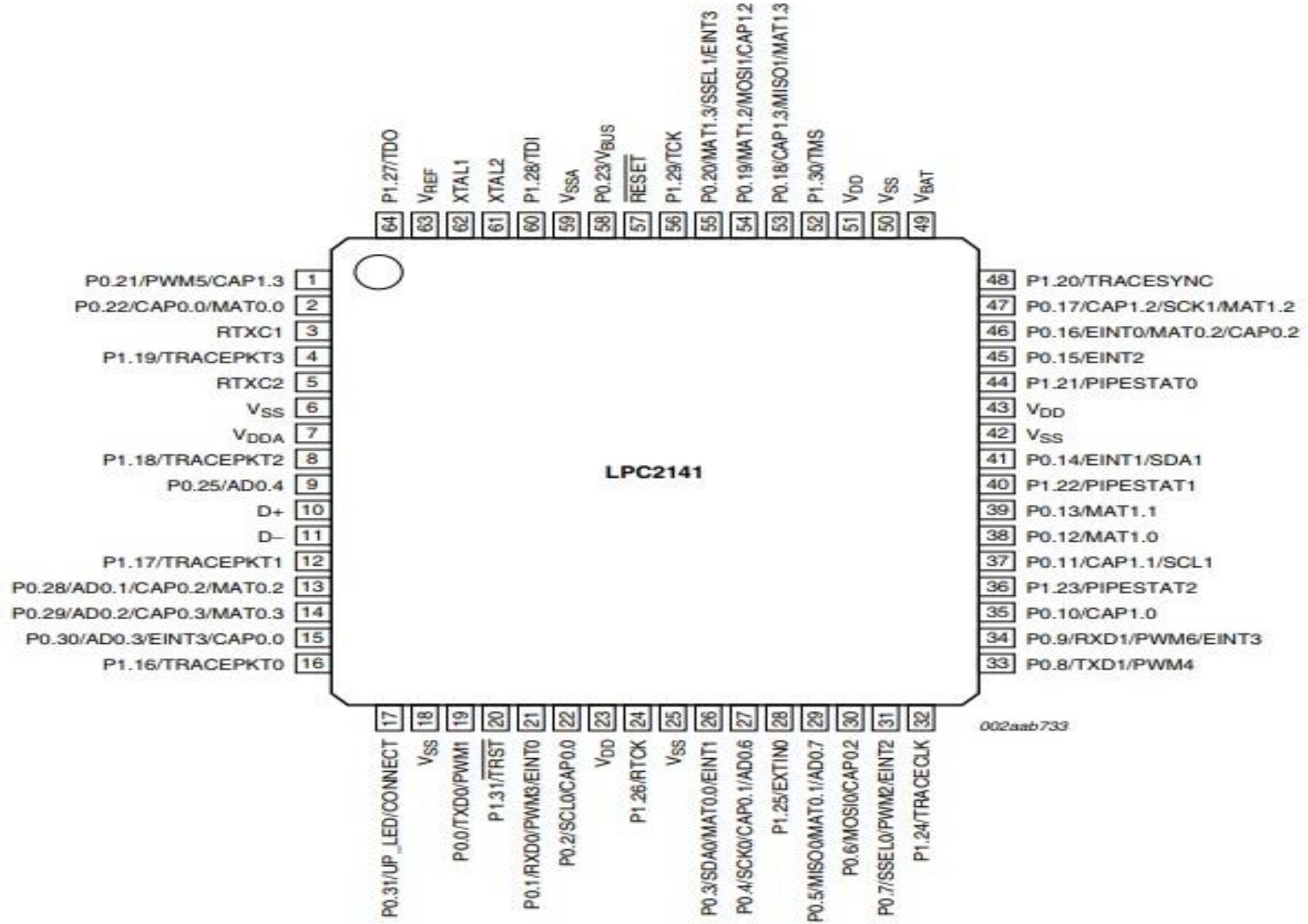
- ADC pins convert analog signals to digital values
- LPC2148 has 10-bit ADC channels
- DAC pin converts digital data to analog output

PWM and Timer Pins

- PWM pins generate pulse width modulated signals
- Used in motor control and power control
- Timer pins include capture and match functions

JTAG / Debug Pins

- TCK – Test Clock
- TMS – Test Mode Select
- TDI – Test Data Input
- TDO – Test Data Output
- Used for debugging and programming


LPC2141
P0.21/PWM5/CAP1.3 1
P0.22/CAP0.0/MAT0.0 2
RTXC1 3
P1.19/TRACEPKT3 4
RTXC2 5
VSS 6
VDDA 7
P1.18/TRACEPKT2 8
P0.25/AD0.4 9
D+ 10
D− 11
P1.17/TRACEPKT1 12
P0.28/AD0.1/CAP0.2/MAT0.2 13
P0.29/AD0.2/CAP0.3/MAT0.3 14
P0.30/AD0.3/EINT3/CAP0.0 15
P1.16/TRACEPKT0 16
P0.31/UP_LED/CONNECT 17
VSS 18
P0.0/TXD0/PWM1 19
P1.31/TRST 20
P0.1/RXD0/PWM3/EINT0 21
P0.2/SCL0/CAP0.0 22
VDD 23
P1.26/RTCK 24
VSS 25
P0.3/SDA0/MAT0.0/EINT1 26
P0.4/SCK0/CAP0.1/AD0.6 27
P1.25/EXTIN0 28
P0.5/MISO0/MAT0.1/AD0.7 29
P0.6/MOSI0/CAP0.2 30
P0.7/SSEL0/PWM2/EINT2 31
P1.24/TRACECLK 32
P0.8/TXD1/PWM4 33
P0.9/RXD1/PWM6/EINT3 34
P0.10/CAP1.0 35
P1.23/PIPESTAT2 36
P0.11/CAP1.1/SCL1 37
P0.12/MAT1.0 38
P0.13/MAT1.1 39
P1.22/PIPESTAT1 40
P0.14/EINT1/SDA1 41
VSS 42
VDD 43
P1.21/PIPESTAT0 44
P0.15/EINT2 45
P0.16/EINT0/MAT0.2/CAP0.2 46
P0.17/CAP1.2/SCK1/MAT1.2 47
P1.20/TRACESYNC 48
P1.27/TDO 64
VREF 63
XTAL1 62
XTAL2 61
P1.28/TDI 60
VSSA 59
P0.23/VBUS 58
RESET 57
P1.29/TCK 56
P0.20/MAT1.3/SSEL1/EINT3 55
P0.19/MAT1.2/MOSI1/CAP1.2 54
P0.18/CAP1.3/MISO1/MAT1.3 53
P1.30/TMS 52
VDD 51
VSS 50
VBAT 49
002aab733

Pin1-(P0.21/ PWM5CAP1.3/ AD1.6)

- P0.21 is a GPIO pin (general purpose I/O pin)
- AD1.6 is obtainable in LPC2144/46/48 microcontrollers only where an AD1.6 denotes ADC-1, i/p-6.
- PWM5 is a pulse width modulator output-5.
- CAP1.3 is a Capture i/p for Timer-1, channel-3

Pin2-(P0.22/ CAP0.0/AD1.7/ MAT0.0 2

- P0.22 is a GPIO digital pin
- AD1.7 pin is available in LPC2144/46/48 only where an AD1.7 denotes ADC-1, input-7
- CAP0.0 is a capture input pin for Timer-0, channel-0.
- MAT0.0 is a match o/p for Timer-0, channel-0

- **Pin3-RTXC1 3**
- It is an I/p to the RTC-oscillator circuit
- **Pin4- TRACEPKT3/ P1.19**
- TRACEPKT3 is a trace packet, bit-3, standard input/output port by the inner pull-up.
- P1.19 is a GPIO digital pin
- **Pin5-RTXC2**
- This is an output pin from the RTC oscillator circuit
- **Pin6, Pin18, Pin25, Pin42, and Pin50**
- These pins are a ground reference
- **Pin7-VDDA**
- This pin is an analog voltage power supply (3.3V), and this voltage is very useful for the on-chip analog to digital converters and digital to analog converters.

- **Pin8- P1.18/TRACEPKT2**
- P1.18 is a GPIO digital pin
- TRACEPKT2 is a trace packet, bit-2, standard input/output port by the inner pull-up.
- **Pin9- P0.25/AOUT/AD0.4**
- P0.25 is a GPIO digital pin I
- AD0.4 denotes ADC-0, input-4
- Aout- the output of DAC and that is accessible only in LPC2142/ LPC2144/ LPC2146/ LPC2148
- **Pin10- D+**
- This pin is a USB bidirectional D+ line
- **Pin11- D-**
- This pin is a USB bidirectional D- line

- **Pin12-P1.17/TRACEPKT1**
- P1.17 is a GPIO digital pin
- TRACEPKT1 is a trace packet, bit-1, standard input/output port by the inner pull-up.
- **Pin13-P0.28/ CAP0.2/ AD0.1/MAT0.2**
- P0.28 is a GPIO digital pin
- AD0.1 denotes ADC-0, input-1
- CAP0.2 is a capture i/p for Timer-0, channel-2.
- MAT0.2 is a match o/p for Timer-0, channel-2
- **Pin14-P0.29/ CAP0.3/ AD0.2/MAT0.3**
- P0.29 is a GPIO digital pin
- AD0.2 denotes ADC-0, input-2
- CAP0.3 is a capture i/p for Timer-0, channel-3.
- MAT0.3 is a match o/p for Timer-0, channel-3

- **Pin15-P0.30/ EINT3/ AD0.3/CAP0.0**
- P0.30 is a GPIO digital pin
- AD0.3 denotes ADC-0, input-3
- EINT3 is an external interrupt 3-input.
- CAP0.3 is a capture i/p for Timer-0, channel-0.
- **Pin16- P1.16/TRACEPKT0**
- P1.16 is a GPIO digital pin
- TRACEPKT1 is a trace packet, bit-0, standard input/output port by inner pull-up
- **Pin17-P0.31/UP_LED/CONNECT**
- P0.31 is a GPIO digital pin
- UP_LED is a USB good link LED indicator. When the device is arranged then it is low and when the device is not arranged, then it is high.
- CONNECT- This signal is used to control an exterior resistor (1.5 kΩ) under the control of a software control, and it is used by the feature of Soft Connect

- **Pin19- P0.0/PWM/TXD0**
- P0.0 is a GPIO digital pin
- TXD0 is a transmitter o/p for UART0.
- PWM1 is a pulse width modulator o/p-1.
- **Pin20- P1.31/TRST**
- P1.31 is a GPIO digital pin
- TRST is a test reset for JTAG interface.
- **Pin21-P0.1/ PWM3/ RXD0/EINT0**
- P0.1 is a GPIO digital pin
- RXD0 is a receiver i/p for UART0.
- PWM3 is a pulse width modulator o/p-3.
- EINT0 is an external interrupt 0-input
- **Pin22- P0.2/ CAP0.0/ SCL0**
- P0.2 is a GPIO digital pin
- SCL0 is an I2C0 clock I/O, and open-drain o/p
- CAP0.0 is a capture i/p for Timer-0, channel-0.

- **Pin 23, 43, and 51- VDD**
- These pins are power supply voltage for the I/O ports as well as the core.
- **Pin24- P1.26/RTCK**
- P1.26 is a GPIO digital pin
- RTCK is a returned test CLK o/p, an additional signal added to the JTAG-port. When the frequency of processor changes then it helps debugger synchronization.
- **Pin26- P0.3/ SDA0/ MAT0.0/EINT1**
- P0.3 is a GPIO digital pin
- SDA0 is an I2C0 data I/O and open drain o/p for I2C bus observance.
- MAT0.0 is matched o/p for timer-0, channel-0.
- EINT1 is an external interrupt 1-i/p.

- **Pin27-P0.4/ CAP0.1/ SCK0/AD0.6**
- P0.4 is a GPIO digital pin I/O
- SCK0 is a serial CLK for SPI0 and SPI CLK o/p from master/ i/p to slave.
- CAP0.1 is a capture i/p for timer-0, channel-0.
- IAD0.6 denotes ADC-0, input-6
- **Pin28-P1.25/EXTIN0**
- P1.25 is a GPIO digital pin I/O
- EXTIN0 is an external trigger i/p, and standard input/output with inner pull-up
- **Pin29- P0.5/MAT0.1/MISO0/AD0.7**
- P0.5 is a GPIO digital pin I/O
- MISO0 is a master in slave out for SPI0, data i/p to SPI-master/data o/p from SPI slave.
- MAT0.1 is a match o/p for timer-0, channel-1.
- AD0.7 denotes ADC-0, input-7.

- **Pin30-P0.6/MOSI0/CAP0.2/AD1.0**
- P0.6 is a GPIO digital pin I/O
- MOSI0 is a master out slave in for SPI0, and data o/p from SPI master/ data i/p to SPI slave.
- CAP0.2 is a capture i/p for Timer-0, channel-2.
- **Pin31-P0.7/ PWM2/ SSEL0/EINT2**
- P0.7 is a GPIO digital pin I/O
- SSEL0 is a slave select for SPI0 and chooses the SPI-interface as a slave.
- PWM2 is a pulse width modulator output-2.
- EINT2 is an external interrupt 2-input.
- **Pin32-P1.24/TRACECLK**
- P1.24 is a GPIO digital pin I/O.
- TRACECLK is a trace CLK and standard input/output port with inner pull-up

- **Pin33-P0.8/TXD1/PWM4/AD1.1**
- P0.8 is a GPIO digital pin I/O
- TXD1 is a transmitter o/p for UART1.
- PWM4 is a pulse width modulator o/p-4.
- AD1.1 denotes ADC-1, input-1, and it is obtainable only in LPC2144/46/48.
- **Pin34- P0.9/PWM6/RXD1/EINT3**
- P0.9 is a GPIO digital pin I/O
- RXD1 is a receiver i/p for UART1.
- PWM6 is a pulse width modulator o/p-6.
- EINT3 is an external interrupt 3-input
- **Pin35-P0.10/RTS1/CAP1.0/AD1.2**
- P0.10 is a GPIO digital pin I/O
- RTS1 is requesting to send o/p for UART1 and LPC2144/46/48.
- CAP1.0 is a capture i/p for timer-1, channel-0.
- AD1.2 denotes ADC-1, input-2, and it is obtainable only in LPC2144/46/48

- **Pin36-P1.23/PIPESTAT2**
- P1.23 is a GPIO digital pin I/O
- PIPESTAT2 is a pipeline status, bit-2., and standard Input/Output port with inner pull-up
- **Pin37-P0.11/ CAP1.1/CTS1/ SCL1**
- P0.11 is a GPIO digital pin I/O
- CTS1 is clear to send i/p for UART1, and these are accessible only in LPC2144/46/48
- CAP1.1 is a capture i/p for timer-1, channel-1.
- SCL1 — I2C1 CLK I/O, and open drain o/p for the I2C-bus observance
- **Pin38-P0.12/ MAT1.0/AD1.3/ DSR1**
- P0.12 is a GPIO digital pin I/O
- DSR1 is a data set ready i/p for UART1, and these are accessible only in LPC2144/46/48.
- MAT1.0 is a match o/p for timer-1, channel-0.
- AD1.3 denotes ADC input-3, and it is accessible only in LPC2144/46/48.

ARM Processor Modes and Registers

- There were six privileged modes and a non-privileged user mode.
- Privilege is the ability to perform certain tasks that cannot be done from User (Unprivileged) mode.
- In User mode, there are limitations on operations that affect overall system configuration, for example, MMU configuration and cache operations.
- Modes are associated with exception events, which are described in Exception Handling.

Table 1. ARM Processor Modes and Privileges

Mode	Function	Privilege
User (USR)	Mode in which most programs and applications run	Unprivileged
FIQ	Entered on an FIQ interrupt exception	Privileged
IRQ	Entered on an IRQ interrupt exception	
Supervisor (SVC)	Entered on reset or when a Supervisor Call instruction (SVC) is executed	
Abort (ABT)	Entered on a memory access exception	
Undef (UND)	Entered when an undefined instruction executed	
System (SYS)	Mode in which the OS runs, sharing the register view with User mode	

ARM Processor Modes and Registers

- The ARM microcontroller architecture (ARM7/9/11) utilizes a 32-bit RISC design with 37 registers (31 general-purpose, 6 status) and 7 operating modes.
- 16 directly accessible registers (r0-r15).
- with banked registers(r8-r14) for fast interrupt (FIQ) and privileged modes.

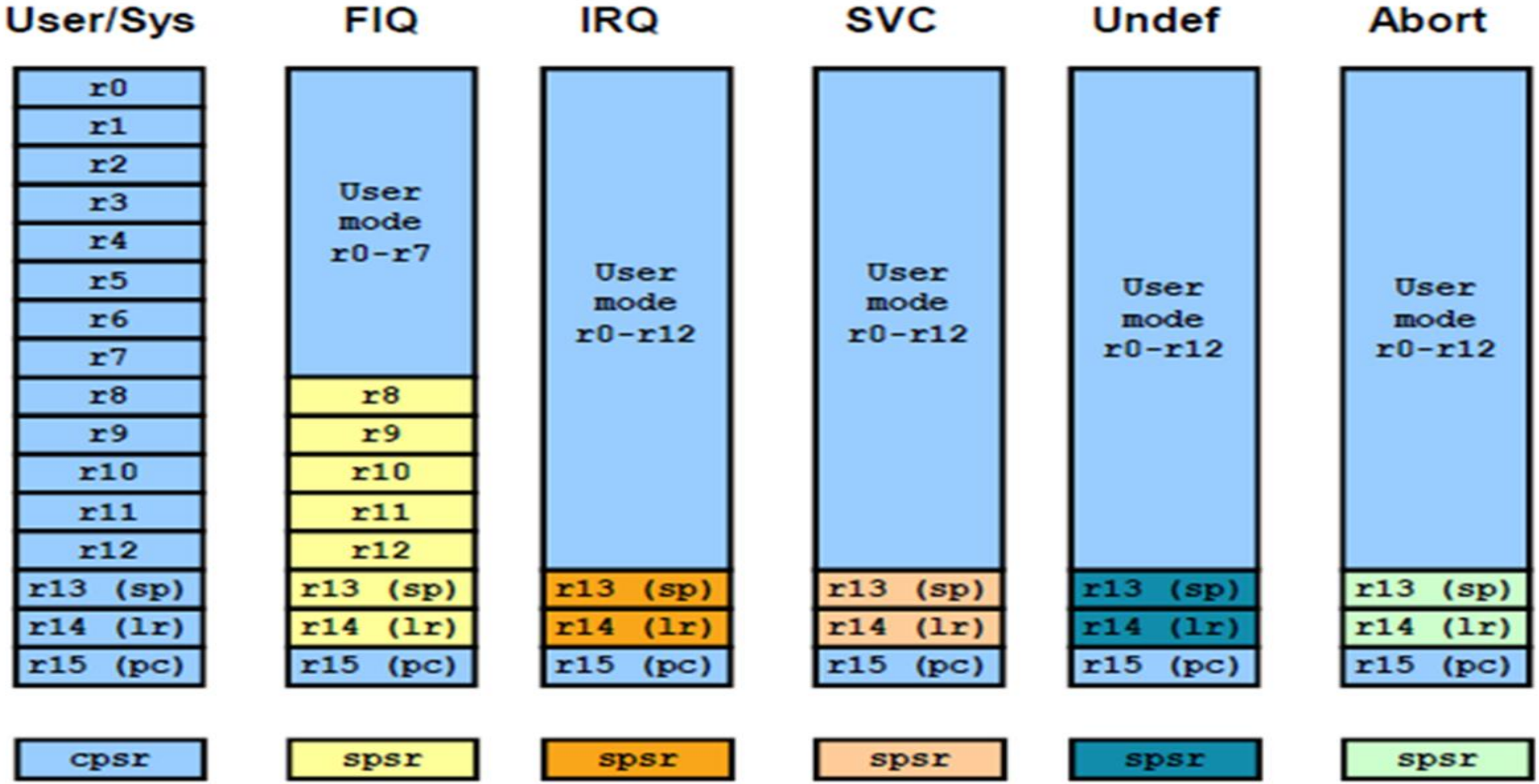
ARM Register Set (32-bit)

- ARM processors have 16 visible registers (r0-r15).
- one or two Program Status Registers (CPSR, SPSR).
-
  - $R0$-$R12$: General-purpose registers for data storage.
  - **$R13$ (SP)**: Stack Pointer, handles stack operations for the current mode.
  - **$R14$ (LR)**: Link Register, stores the return address during function calls/subroutines.
  - **$R15$ (PC)**: Program Counter, holds the address of the next instruction.
  - **CPSR (Current Program Status Register)**: Contains condition code flags (N, Z, C, V), interrupt disable bits, and current mode bits.
  - **SPSR (Saved Program Status Register)**: Stores the previous mode's CPSR when an exception occurs.

ARM Processor Operating Modes

The processor operates in seven modes, determined by the 5-bit field in the CPSR:

1. **User (usr)**: Unprivileged mode for running application tasks.
2. **System (sys)**: Privileged mode for operating system tasks, uses User mode registers.
3. **Supervisor (svc)**: Entered on reset or software interrupt (SWI).
4. **IRQ (irq)**: Low-priority interrupt mode.
5. **FIQ (fiq)**: High-priority, fast interrupt mode with banked registers ($R8$-$R14$) to minimize context saving.
6. **Abort (abt)**: Entered on memory access violations.
7. **Undefined (und)**: Entered when handling undefined instructions.



Note: System mode uses the User mode register set

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) There are 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
- Registers R1 to R12 can be used for data movement. These are the general purpose registers. The register R13, R14 and R15 are also used for data movement.
- 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 MVN instruction.

MVN instruction

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 result will be stuck in the first_register.
- **Example :**
- MOV R5, R7 ; copy content of R7 to R5 MOV R9, R3 ; copy content of R3 to R9
- MVN Rl, R4 ; move the complemented value of R4 to R1

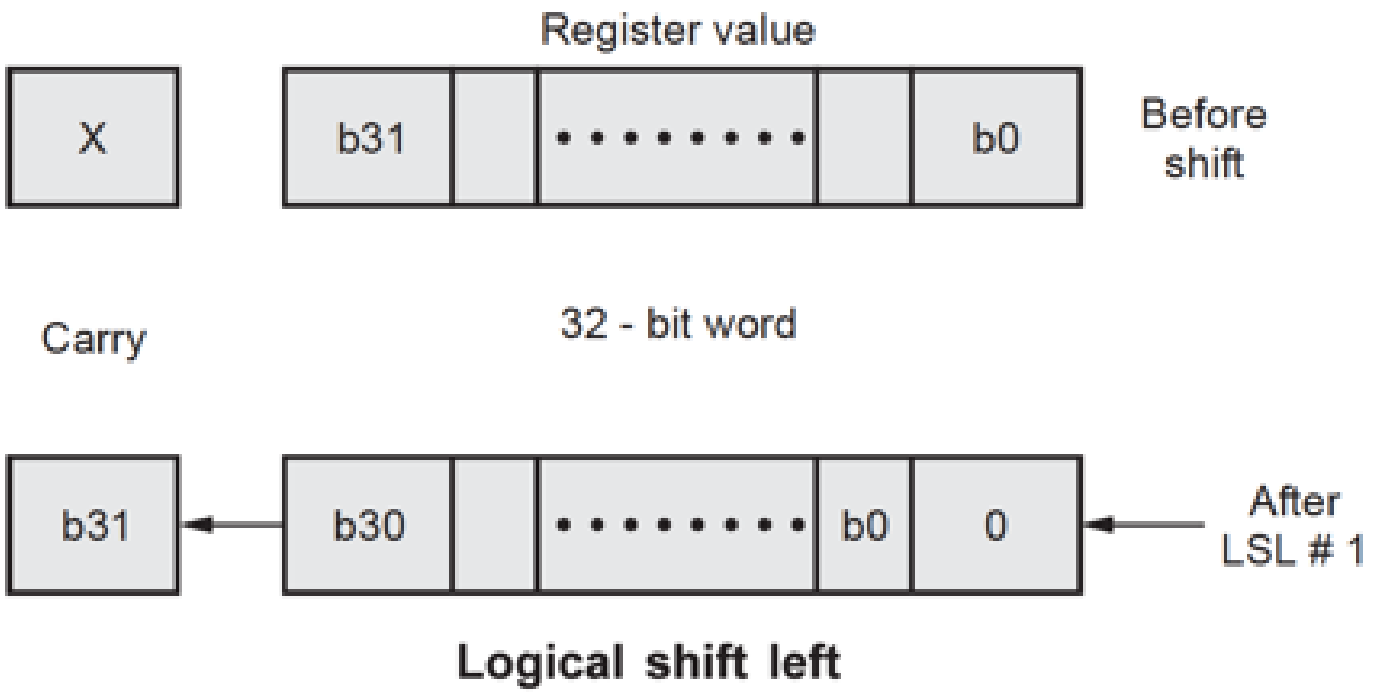
Shift and Rotate

- Logical and arithmetic are the two shift types. There are six mnemonics for the different shift types :
- Logical Shift Left (LSL)
- Arithmetic Shift Left (ASL)
- Logical Shift Right (LSR)
- Arithmetic Shift Right (ASR)
- Rotate Right (ROR)
- Rotate Right with Extend (RRX)

Logical Shift Left

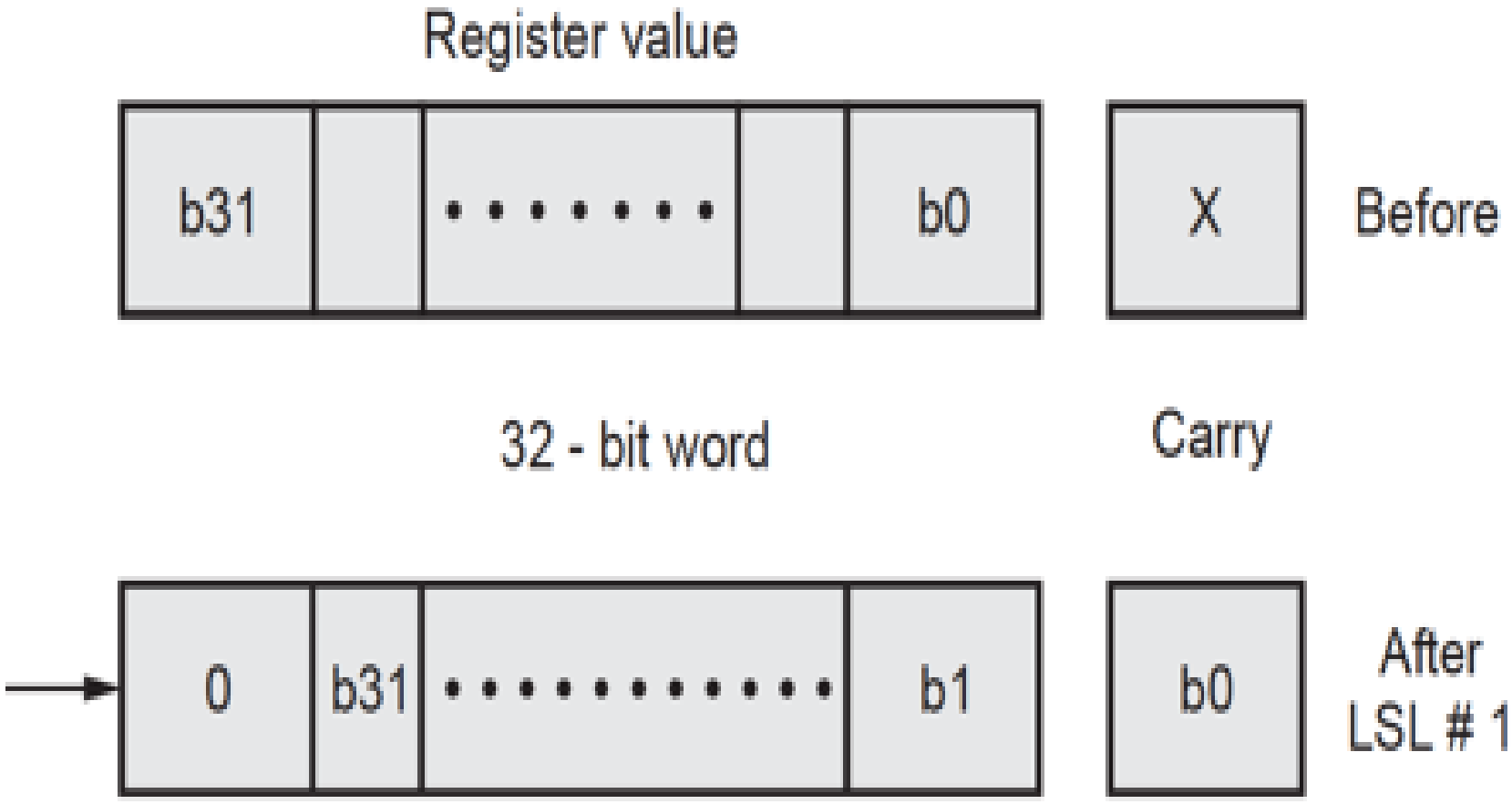
- Shifts left by the specified amount. LSL means "logical shift left by the specified number of bits." This instruction is executed in a single clock cycle.
- Example : LSL # n
- Here n is the number of bit positions by which the value is shifted. Shifting left by n-bit on a signed or unsigned binary number has the effect of multiplying it by 2n

Logical Shift Left



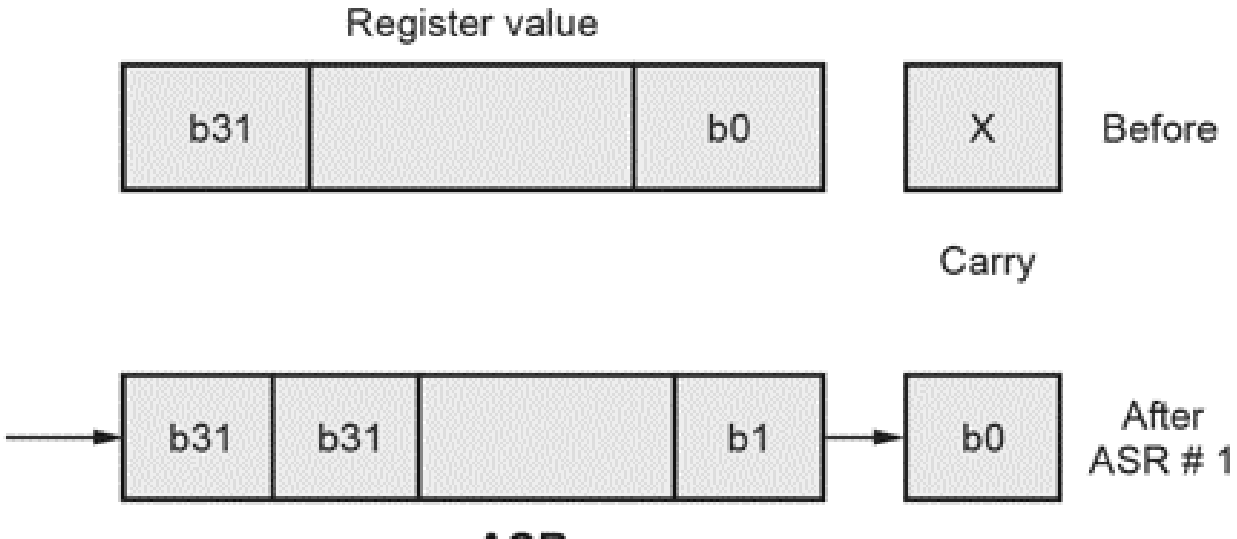
Logical Shift Right (LSR)

- LSR by 0 to 32 places, fill the vacated bits at the most significant end of the word with zeros.
- Logical shifts can be useful as efficient ways of performing multiplication or division of unsigned integers by powers of two.
- Shifting right by n-bit on an unsigned binary number has the effect of dividing it by 2^n.


Register value
b31
b0
X
Before
32 - bit word
Carry
0
b31
b1
b0
After
LSL # 1

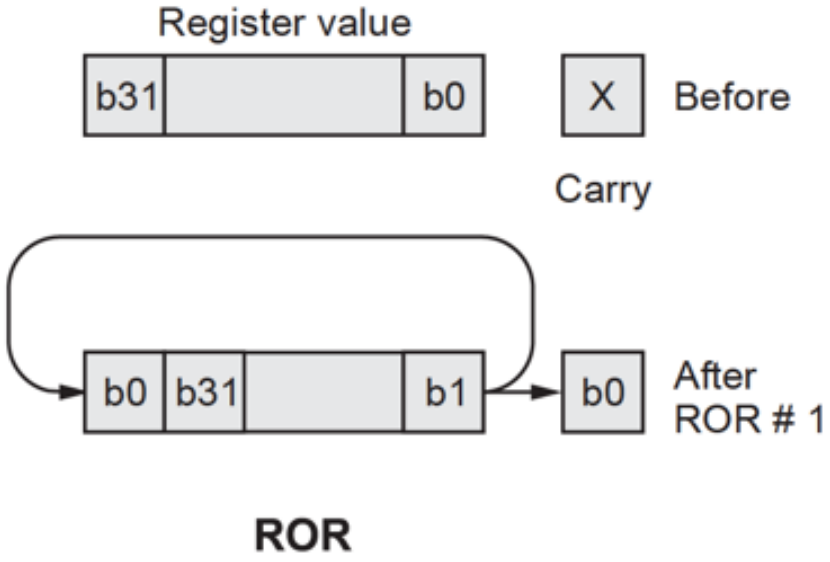
Arithmetic Shift Left and Right

- 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.



Rotate Right (ROR)

- Rotate right by 0 to 32 places. The last bit rotated out is available in the carry 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).



Conditional Code Instruction

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

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 destination value is not changed only conditional flags are affected.

Multiplication Instruction

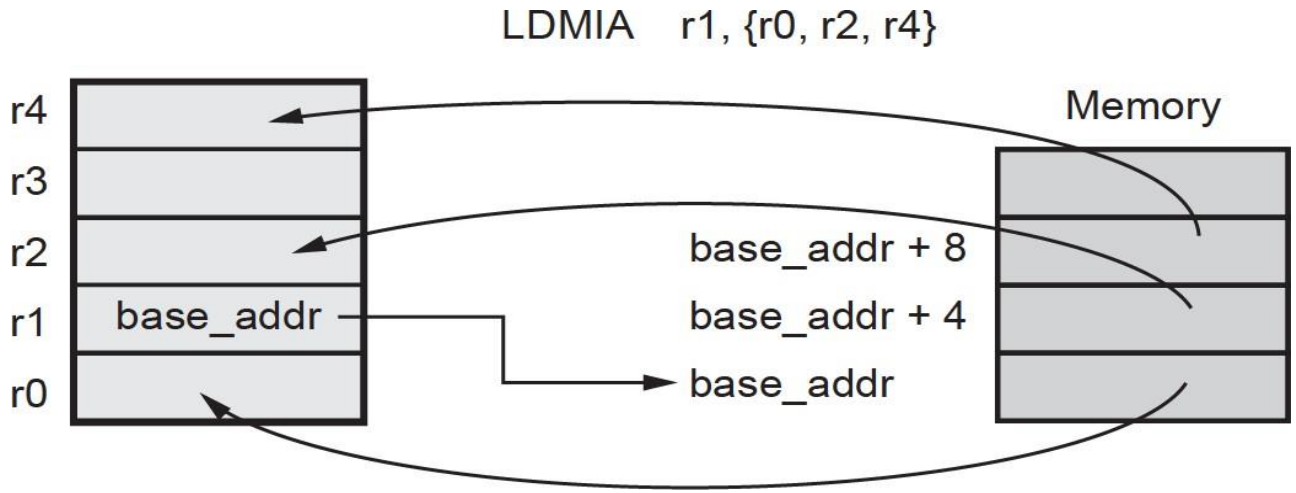
- Multiplication instruction takes more than one cycle. it also requires hardware to perform operation.
- **Multiply**
- Syntax : **MUL Rd, Rm, Rs** where Rd = Destination register Rm, Rs = Source register
- Example : **MUL RO, R1, R2 @ RO = R1 x R2**
- Features :
  - Second operand cannot be immediate.
  - The result register must be different from the first operand.
  - 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**

STM Instruction

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

Branch Instruction

- The branch instructions cause the processor to execute instructions from a different address. Two branch instruction are available B and BL. These instructions are only executed if the condition is true.
- The BL instruction in addition to branching, also stores the return address in the lr register, and hence can be used for sub-routine invocation.
- 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.

ARM Memory Organization

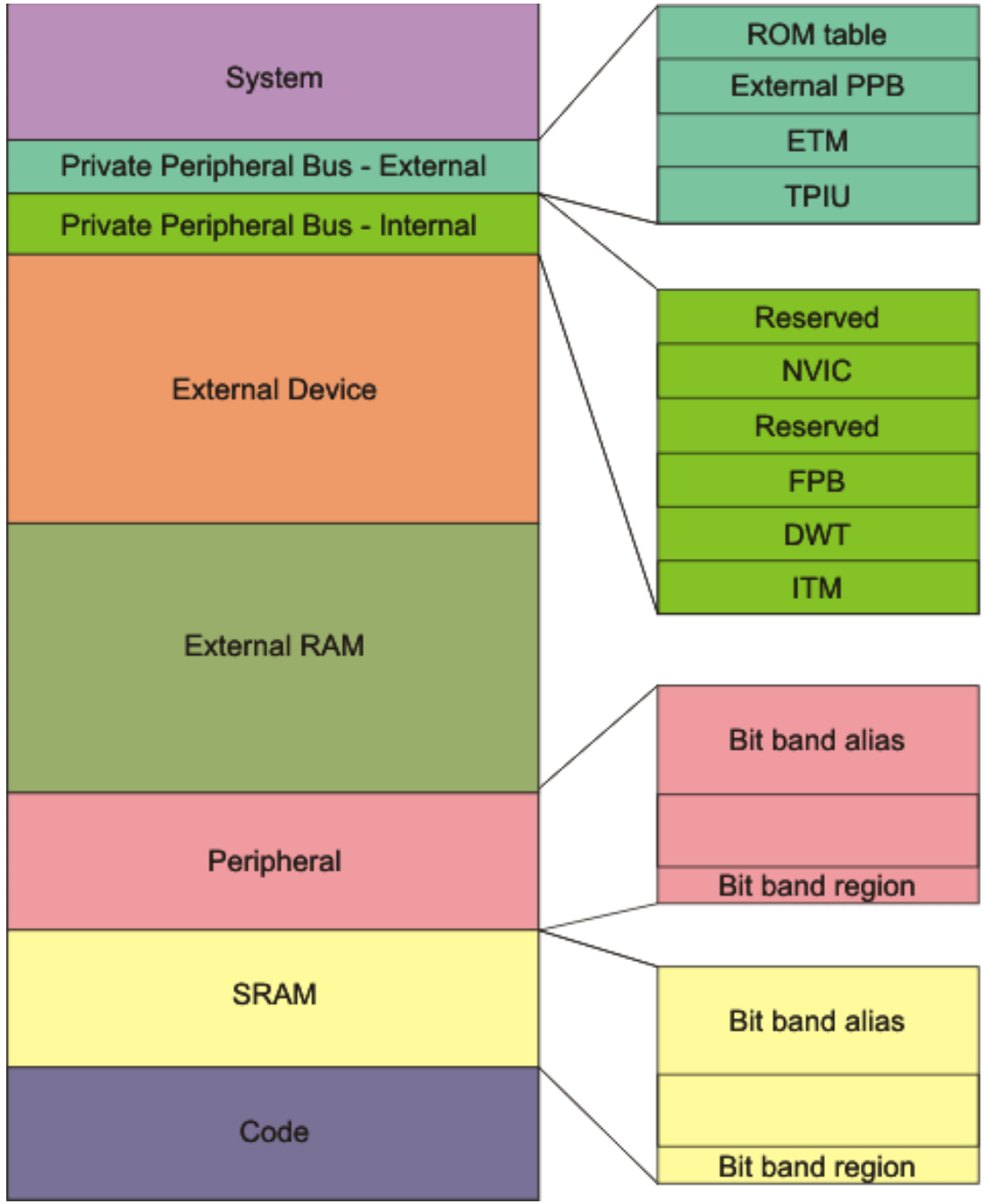
- ARM Cortex-M microcontrollers use a **32-bit linear address space (4 GB)** divided into fixed regions for code, data, peripherals, and system control. The architecture supports **memory-mapped I/O** and **bit-banding** for efficient bit manipulation.
- **Unified Address Space (4 GB):**
  - ARM processors use a 32-bit architecture, supporting a flat memory space from **0x00000000 - 0x1FFFFFFF.**
  - **Memory-Mapped Peripherals:** Peripherals are not accessed via special I/O instructions but rather mapped directly into the address space, treating registers (e.g., GPIO, UART) like memory locations.

Memory Regions (Cortex-M Example)

- **Code Region (0x00000000 - 0x1FFFFFFF):** Typically stores executable code (Flash).
- **SRAM Region (0x20000000 - 0x3FFFFFFF):** Used for data, stack, and heap.
- **Peripheral Region (0x40000000 - 0x5FFFFFFF):** Reserved for on-chip peripherals.
- **System/External Region:** External RAM, device, and private peripheral bus (PPB) for debug components.

ARM Memory Organization

- **Load-Store Architecture:** Only load and store instructions can access memory; all data manipulation occurs within the large register file (R0-R15), minimizing memory access time.
- **Endianness:** Supports both Little-Endian and Big-Endian, though most ARM systems use Little-Endian.
- **Memory Types:**
  - **Normal:** Used for RAM/Flash, allows buffering and reordering.
  - **Device:** Used for peripherals, ensures strict access ordering.
  - **Strongly Ordered:** Ensures strict access order for critical system controls.
- **Hierarchy:** Features fast, local cache memory (L1/L2) close to the core, with slower main memory (SRAM/Flash) further away.


System
Private Peripheral Bus - External
Private Peripheral Bus - Internal
External Device
External RAM
Peripheral
SRAM
Code
ROM table
External PPB
ETM
TPIU
Reserved
NVIC
Reserved
FPB
DWT
ITM
Bit band alias
Bit band region
Bit band alias
Bit band region

ARM Memory Organization

- **Private Peripheral Bus - External** - Provides access to :
  - the Trace Port Interface Unit (TPIU),
  - the Embedded Trace Macrocell (ETM),
  - the ROM table,
  - implementation-specific areas of the PPB memory map.
- **Private Peripheral Bus - External** - Provides access to :
  - the Instrumentation Trace Macrocell (ITM),
  - the Data Watchpoint and Trace (DWT),
  - the Flashpatch and Breakpoint (FPB),
  - the System Control Space (SCS), including the MPU and the Nested Vectored Interrupt Controller (NVIC).

ARM Memory Organization

- **External Device -** This region is used for external device memory.
- **External RAM -** This region is used for data.
- **Peripheral -** This region includes bit band and bit band alias areas.
  - Peripheral Bit-band alias - Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
  - Peripheral bit-band region - Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write.

ARM Memory Organization

- **SRAM -** This executable region is for data storage. Code can also be stored here. This region includes bit band and bit band alias areas.
  - SRAM Bit-band alias - Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
  - SRAM bit-band region - Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write.
- **Code -** This executable region is for program code. Data can also be stored here.

Programming ports

- ARM microcontrollers, such as the common LPC2148 (ARM7), utilize specialized ports for programming, debugging, and general-purpose I/O (GPIO). The primary methods include Joint Test Action Group (JTAG) for debugging, In-System Programming (ISP) via UART for flashing code, and versatile GPIO pins (e.g., P0, P1) for peripheral interfacing.

Programming ports

- **JTAG Port (Joint Test Action Group)**:
- Typically a 5-pin interface (TDI, TDO, TMS, TCK, TRST) used for debugging and programming, allowing full control over the ARM core, including halting and stepping through code.
- **ISP (In-System Programming) Port**:
- Uses UART0 or UART1, allowing the flash memory to be reprogrammed through a serial connection (COM port) without removing the microcontroller from the circuit.

Programming ports

- **GPIO Ports (General Purpose Input/Output)**
- **Port 0 (P0):** A 32-bit port (P0.0–P0.31) where most pins are configurable as inputs or outputs.
- **Port 1 (P1):** A 32-bit port (P1.16–P1.31 commonly used in LPC2148).
- These pins are configured using PINSEL0/PINSEL1 registers to choose between GPIO or alternate functions (UART, PWM, ADC).

- **Key Registers for Port Configuration**
- **PINSEL0/1:** Pin Select registers to configure multiplexed pins.
- **IODIR:** Input/Output Direction register, where 1 sets a pin as output and 0 as input.
- **IOSET/IOCLR:** Used to set (high) or clear (low) output pins.
- **Programming Sequence**
- Connect via JTAG or UART (ISP).
- Configure PINSEL registers to define pin functionality.
- Set direction using IODIR.
- Write/Read data using IOSET/IOCLR or IOPIN.

What is Timer ?

- 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.

Timer in LPC2148

- 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 generation

- 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.

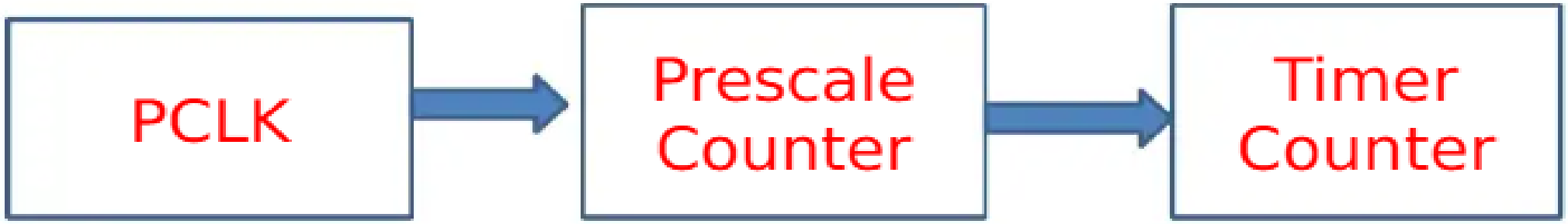
Capture Register: As the name suggests it is used to Capture Input signal. When a transition event occurs on a Capture pin , it can be used to copy the value of TC into any of the 4 Capture Register or to genreate an Interrupt. Hence these can be also used to demodulated PWM signals.

Match Register:

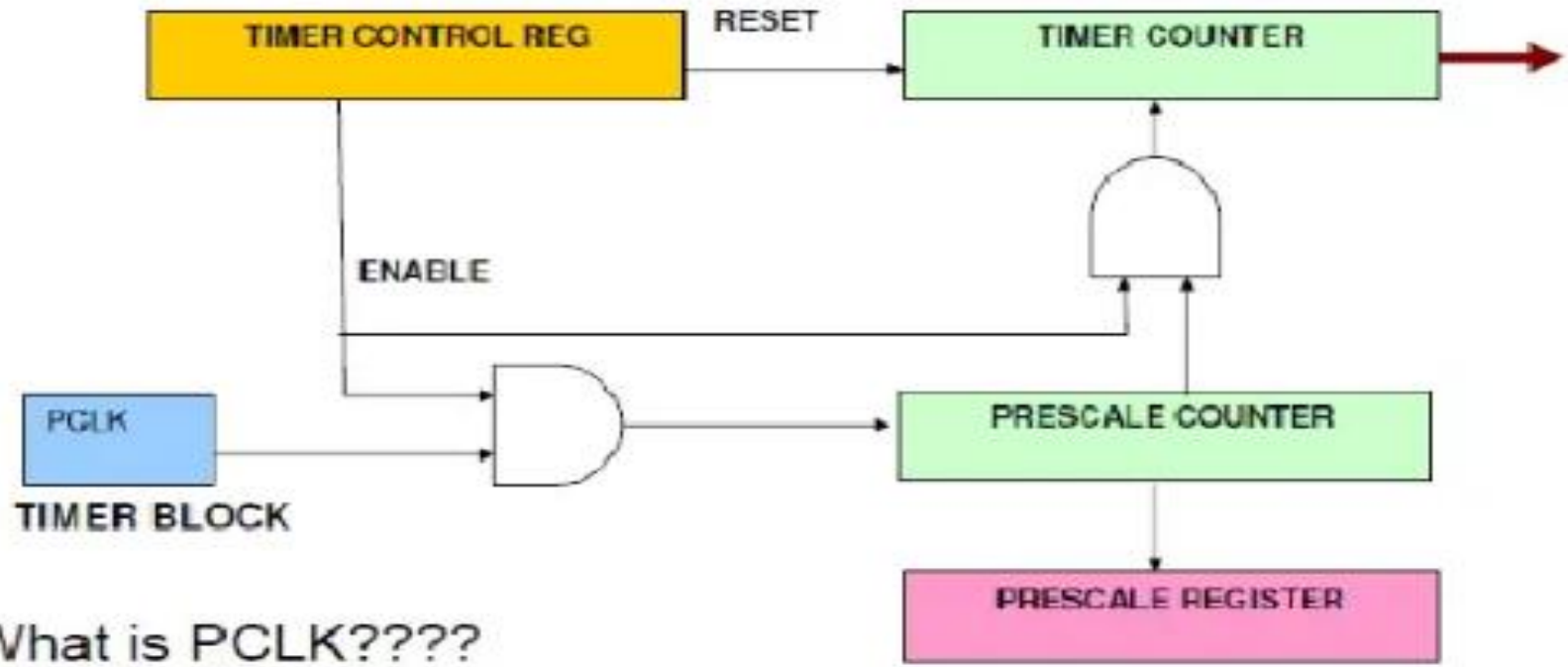
- A Match Register is a Register which contains a specific value set by the user. When the Timer starts – every time after TC is incremented the value in TC is compared with match register. If it matches then it can Reset the Timer or can generate an interrupt as defined by the user. We are only concerned with match registers in this PPT.

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.



LPC214x Timer Block

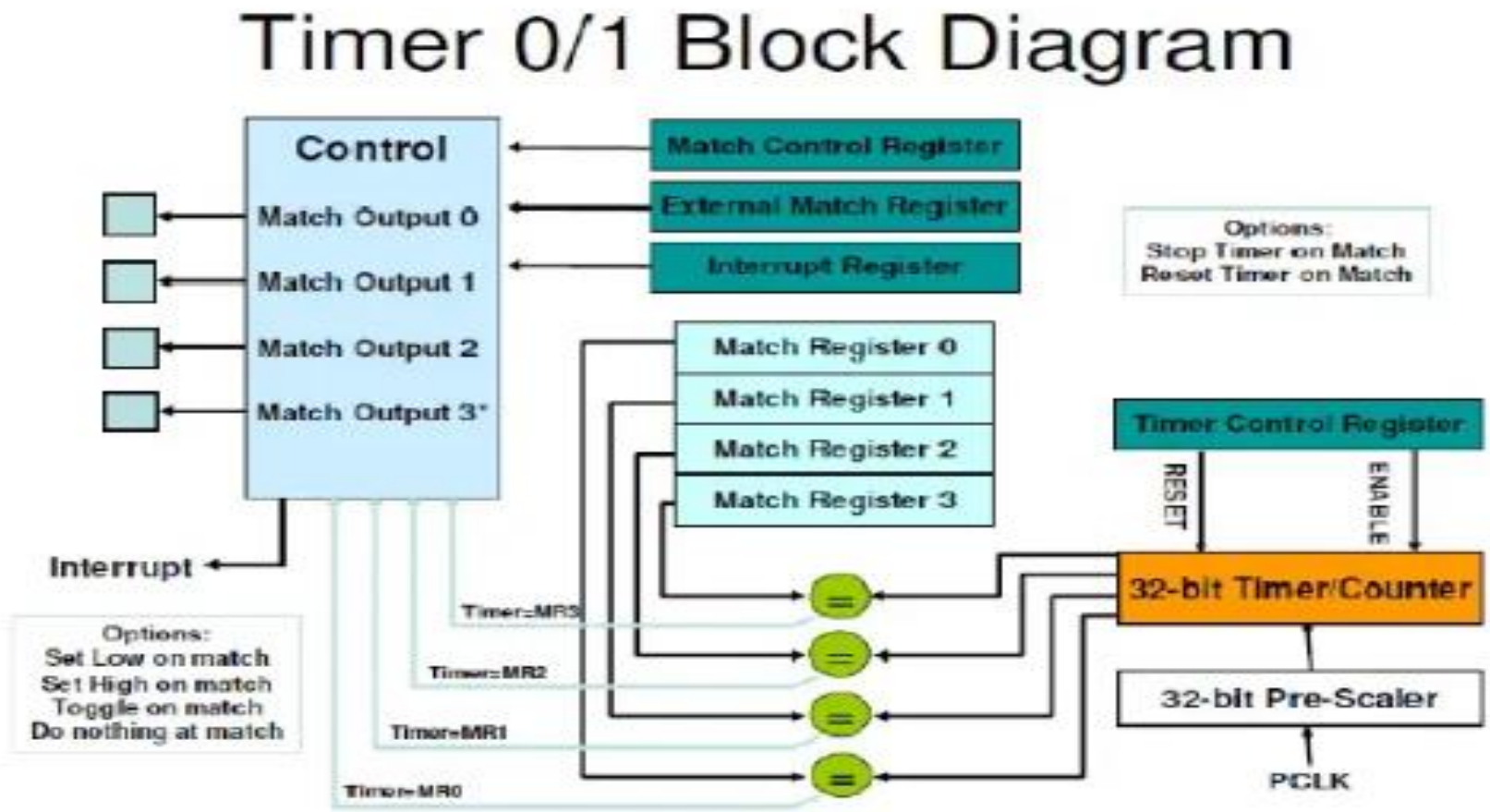


What is PCLK????

- 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 Match register values are continuously compared to the Timer Counter value, when the two values are equal, actions (Match pin / Interrupt) can be triggered automatically



The Match register values are continuously compared to the Timer Counter value, when the two values are equal, actions (Match pin / Interrupt) can be triggered automatically.

Timer Control Register (TCR, TIMER0: T0TCR)

Timer Control register used to control the timer control functions. We'll enable, disable and reset Timer Counter (TC) through this register

Bit	Symbol	Description	Reset value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA