ARM Processor

Introduction to 16/32-bit processors -

- 32-bit processors execute 32-bit operation at a time in single cycle.
- 32-bit processor having a subset of instructions for 16bit coding is called 16/32 bit processor.
- These processors provides solutions for open platforms running complex operating systems for wireless, consumer and imaging applications, secure applications including smart cards and payment terminals.
- Normally pre-scalar architecture is implemented. It means that multiple instructions can be executed in a single cycle using pipelining technique.
- RISC architecture is implemented in them (Reduced Instruction set coding)

RISC	CISC
1. It emphasizes on s/w to optimize the instruction set	1. It emphasizes on h/w.
2. It has simple decoding of instruction	2. It has complex decoding
3. It uses less n.o. instructions	3. It uses large n.o. instructions
4. less time for execution	4. execution time is longer
5. Pipelining is Simple	5. Difficult
6. It can be used with high end applications like tele communication, image processing & vedio processing etc.	6. It can be used with low end app's like home automation & security systems etc
7. It has fixed format instructions	7. It has variable format
8. Ex:- ARM, Sparc	8. Ex:- Motorola 68000 family, Intel x86 family

ARM: Means advanced RISC machine developed at ACRON Computers ~~machine~~ ltd. of Cambridge England b/w 1983 & 85

→ It will be used for commercial purpose. It offers a wide range of processor cores based on a common architecture & delivering high performance with low power consumption & cost.

→ It is used in many applications like wireless, consumer, imaging, industrial, networking, vedeo gaming, modems, mobile phone, hand cams etc.

Features of ARM :-

→ Architectural simplicity, it allows very small implementations which results in low power consumption,
→ large uniform register file
→ load - store architecture
→ uniform & fixed length instruction fields
→ ARM designed a family of RISC super scalar processor architectures based on VLSI
→ 3 address instruction formats
→ ARM has various architectures like ARM2, ARM3, ARM6, ARM7, ARM8 & ARM9

ARM 7 features :-

→ HCMOS (High performance complementary MOS technology)
→ low power consumption
→ 32-bit data bus
→ 32-bit address bus
→ 4GB memory capacity
→ 32-bit ALU
→ 16, 32 bit registers
→ high performance multipliers
→ co-processor interface
→ All the RISC features
→ fast high priority request & interrupt request
→ 32-bit instruction set & 16-bit thumb instruction subset

ARM architecture :-

→ Based on Berkely RISC machine ARM architecture has fixed length instruction, pipelining, load/store architecture
→ It has 32-bit architecture.
→ Most ARMs impliment 2 instruction sets : 32 bit ARM inst set
16 bit Thumb " "
→ ARM architecture provides an overview of the processor and describes how data moves b/w it's different parts
→ data enters the coprocessor core to data bus
→ The data may be an instruction to be executed by the Processor or any data item to be processed
→ The instruction decoder translates the instruction before they are executed
→ Each istruction executed belongs to a particular instruction set

→ It uses load-store architecture means it has 2 types of instructions for transferring data in & out of the Processor

→ load instructions copy data from memory to registers in the code

→ Store instruction copy data from registers to memory

→ As data is read from memory & placed into register file

→ ARM have 2 source registers R_n, R_m & a single result or destination register R_d

→ Source operands are read from register file using the internal buses A & B

→ The ALU takes the register values R_n & R_m from A & B buses & computes the result.

→ load-store instructions use the ALU to generate an address to be held in address register ~~in address~~ & broadcast on address bus

→ The contents of the R_m register alternatively can be preprocessed in the barrel shifter before applying as an i/p to the ALU. A wide range of expressions and addresses can be calculated using barrel shifter & ALU

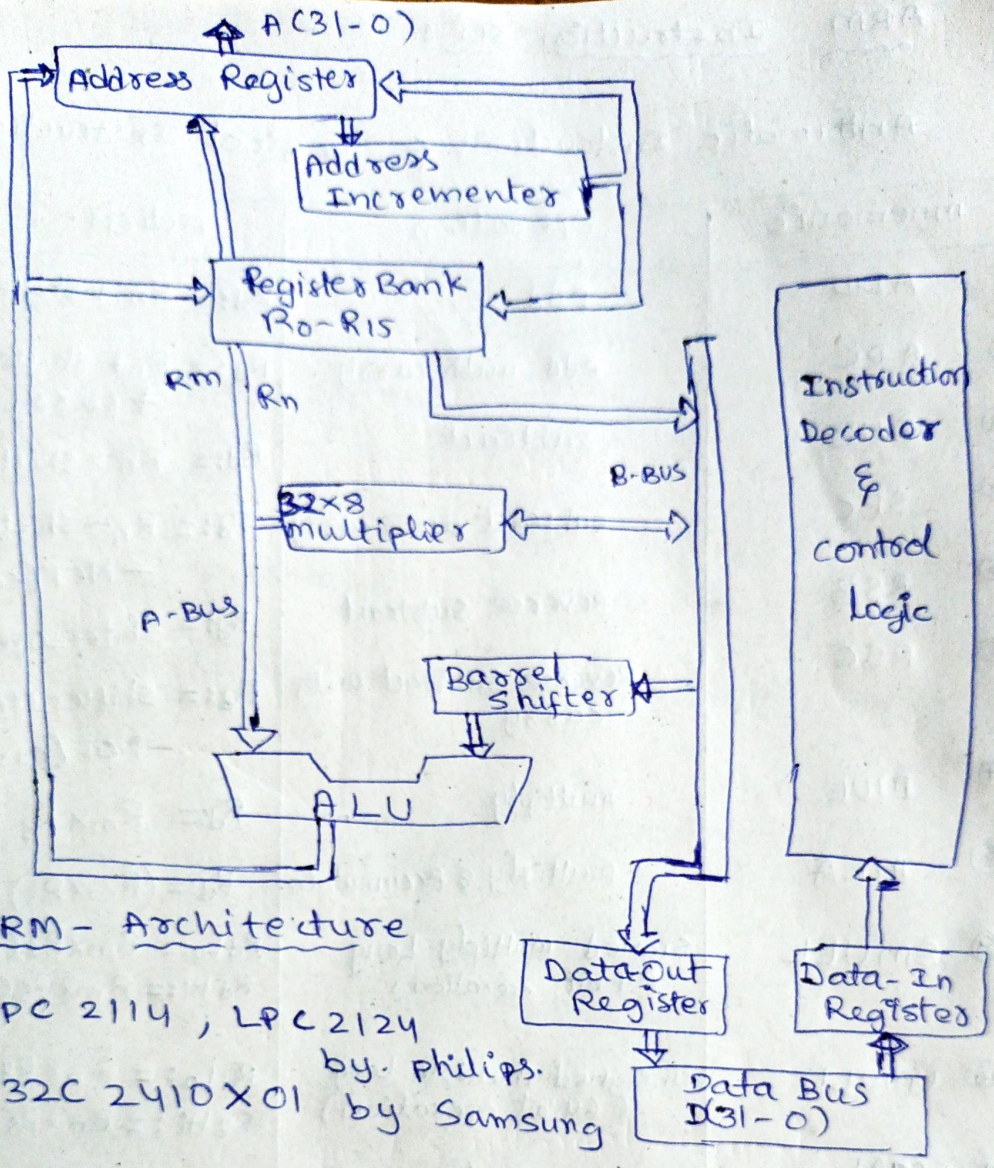
→ For load & store instructions, the incrementer updates the contents of the address register before the processor core reads & writes the next register value from & to the consecutive memory location.

→ The processor core continues the execution of instructions only when an exception & interrupt occurs, the normal execution is changed.

⇒ ARM Based MCU's (micro controller units).

Several companies have designed mcu's based on ARM architecture

→ STR710, STR720 by the ST microsystems Ltd.



ARM - Architecture

→ LPC 2114, LPC 2124 by philips.

→ S32C 2410X01 by Samsung etc.

The above mcu's have ARM based CPUs some of the features are.

→ STR710 has fast Flash memory, high speed SRAM, 5 SPI + UART + I²C, USB IF, ~~CANIF~~ CANIF.

→ LPC2114 has fast Flash upto 256 kb, SRAM of 16 kb, cache of 8 kb for instruction and data, MMU, 4 SPI + UART + I²C, USB IF and CANIF.

→ S32C 2410X01 has fast boot loader, cache of 16 kb for instructions, 16 kb cache for data, 2 SPI + 3 channel UART, 2 port USB host, 1 port USB device, 4 DMA channels and 8 channel 10 bit DAC.

here
⇒ SPI (serial port Interface), I²C (Inter Integrated circuits)
MMU (memory management Unit)

ARM Instruction set:

Arithmatic Instructions & Logical Instructions

	mnemonic	operation	Action.
①	ADD	Add	Rd := Rn + shifter_operand
②	ADC	Add with carry	Rd := Rn + shifter_operand + carry_flag
③	SUB	subtract	Rd := Rn − shifter_operand
④	SBC	subtract with carry	Rd := Rn − shifter_operand − NOT(carry_flag)
⑤	RSB	Reverse subtract	Rd := Shifter_operand − Rn
⑥	RSC	Reverse Subtract with carry	Rd := shifter_operand − Rn − NOT(carry_flag)
⑦	MUL	multiply	Rd := Rm × Rs
⑧	MLA	multiply & Accumulate	Rd := (Rm × Rs) + Rn
⑨	SMULL	Signed multiply Long (64 bit operation)	RdLo := Rm × Rs (31:0) RdHi := Rm × Rs (63:32)
⑩	UMULL	unsigned multiply Long (64 bit operation)	RdLo := Rm × Rs (31:0) RdHi := Rm × Rs (63:32)
⑪	SMLAL	signed multiply Accumulate long (64 bit operation)	RdLo := Rm × Rs (31:0) + RdLo RdHi := Rm × Rs (63:32) + RdHi + carry from (Rm × Rs (31:0) + RdLo)
⑫	UMLAL	unsigned multiply Accumulate long (64 bit operation)	RdLo := Rm × Rs (31:0) + RdLo RdHi := Rm × Rs (63:32) + RdHi + carry from (Rm × Rs (31:0) + RdLo)
⑬	CLZ	count leading zero	Number of Leading zeros in Rm (Source Reg) is put in Rd
⑭	AND	Logical AND operation	Rd := Rn AND Shift_operand
⑮	ORR	logical OR operation	Rd := Rn OR Shift_operand

(16)	EOR	logical ExclusiveOR	$R_d := R_n$ Ex-OR Shift-operand
(17)	MOV	move	$R_d :=$ Shifter_operand (no 1st operand)
(18)	MVN	move NOT	$R_d :=$ NOT (Shifter_operand) (no 1st operand).
(19)	CMP	compare	Update flag after doing Rn - Shifter_operand
(20)	CMN	Compare Negated	Update flag after doing Rn + Shifter_operand
(21)	TST	Test	Update flag after Rn AND Shifter-operand
(22)	TEQ	Test Equivalence	Update flag after Rn EOR shifter_operand
(23)	BIC	Bit clear (logical NAND)	$R_d :=$ Rn AND NOT (Shifter_operand)

Load and Store Instructions :

	mnemonic	operation
(1)	LDR	Load Register (word)
(2)	LDRB	Load Byte
(3)	LDRSB	Load Signed Byte.
(4)	LDRH	Load Half Word
(5)	LDRSH	Load Signed Halfword.
(6)	STR	Store Register (word)
(7)	STRB	store Byte
(8)	STRH	store Half word
(9)	STM	store multiple Registers
(10)	LDM	Load multiple Register