Chapter 1 - Computer Abstractions and Technology

dynamic random access memory (DRAM) Memory built as an integrated circuit; it provides random access to any location. Access times are 50 nanoseconds and cost per gigabyte in 2012 was $5 to $10.

instruction set architecture Also **called architecture.** An abstract interface between the hardware and the lowest-level software that encompasses all the information necessary to write a machine language program that will run correctly, including instructions, registers, memory access, I/O, and so on.

application binary interface (ABI) The user portion of the instruction set plus the operating system interfaces used by application programmers. It defines a standard for binary portability across computers.

The big picture:
Both hardware and software consist of hierarchical layers using abstraction, with each lower layer hiding details from the level above. One key interface between the levels of abstraction is the instruction set architecture—the interface between the hardware and low-level software. This abstract interface enables many implementations of varying cost and performance to run identical software.

implementation
Hardware that obeys the architecture abstraction.

Operands of the computer Hardware
Unlike programs in high-level languages, the operands of arithmetic instructions are restricted; they must be from a limited number of special locations built directly in hardware called **registers**. **Registers** are primitives used in hardware design that are also visible to the programmer when the computer is completed, so you can **think of registers as the bricks of computer construction**. The size of a register in the ARM architecture is 32 bits; groups of **32 bits occur so frequently that they are given the name word in the ARM architecture. NOTE: 32 bits = 4 bytes. Word = 32 bits or 4 bytes.**

Load
The data transfer instruction that copies data from memory to a register is traditionally called load. The format of the load instruction is the name of the operation followed by the register to be loaded, then a constant and register used to access memory. The sum of the constant portion of the instruction and the contents of the second register forms the memory address. The actual ARM name for this instruction is LDR, standing for load word into register

Store
The instruction complementary to load is traditionally called store; it copies data from a register to memory. The format of a store is similar to that of a load: the name of the operation, followed by the register to be stored, then offset to select the array element, and fi nally the base register. Once again, the ARM address is specifi ed in part by a constant and in part by the contents of a register. **The actual ARM name is STR, standing for store word from a register.**

In ARM, words must start at addresses that are multiples of 4. This requirement is called an **alignment restriction**, and many architectures have it.

Chapter 2

ARM operands

Name	Example	Comments
16 registers	r0, r1, r2, . . . , r11, r12, sp, lr, pc	Fast locations for data. In ARM, data must be in registers to perform arithmetic, register
2^{30} memory words	Memory[0], Memory[4], . . . , Memory[4294967292]	Accessed only by data transfer instructions. ARM uses byte addresses, so sequential word addresses differ by 4. Memory holds data structures, arrays, and spilled registers.

ARM assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	ADD r1,r2,r3	r1 = r2 – r3	3 register operands
	subtract	SUB r1,r2,r3	r1 = r2 + r3	3 register operands
Data transfer	load register	LDR r1, [r2,#20]	r1 = Memory[r2 + 20]	Word from memory to register
	store register	STR r1, [r2,#20]	Memory[r2 + 20] = r1	Word from memory to register
	load register halfword	LDRH r1, [r2,#20]	r1 = Memory[r2 + 20]	Halfword memory to register
	load register halfword signed	LDRHS r1, [r2,#20]	r1 = Memory[r2 + 20]	Halfword memory to register
	store register halfword	STRH r1, [r2,#20]	Memory[r2 + 20] = r1	Halfword register to memory
	load register byte	LDRB r1, [r2,#20]	r1 = Memory[r2 + 20]	Byte from memory to register
	load register byte signed	LDRBS r1, [r2,#20]	r1 = Memory[r2 + 20]	Byte from memory to register
	store register byte	STRB r1, [r2,#20]	Memory[r2 + 20] = r1	Byte from register to memory
	swap	SWP r1, [r2,#20]	r1 = Memory[r2 + 20], Memory[r2 + 20] = r1	Atomic swap register and memory
	mov	MOV r1, r2	r1 = r2	Copy value into register
Logical	and	AND r1, r2, r3	r1 = r2 & r3	Three reg. operands; bit-by-bit AND
	or	ORR r1, r2, r3	r1 = r2 | r3	Three reg. operands; bit-by-bit OR
	not	MVN r1, r2	r1 = ~ r2	Two reg. operands; bit-by-bit NOT
	logical shift left (optional operation)	LSL r1, r2, #10	r1 = r2 << 10	Shift left by constant
	logical shift right (optional operation)	LSR r1, r2, #10	r1 = r2 >> 10	Shift right by constant
Conditional Branch	compare	CMP r1, r2	cond. flag = r1 – r2	Compare for conditional branch
	branch on EQ, NE, LT, LE, GT, GE, LO, LS, HI, HS, VS, VC, MI, PL	BEQ 25	if (r1 == r2) go to PC + 8 + 100	Conditional Test; PC-relative
Unconditional Branch	branch (always)	B 2500	go to PC + 8 + 10000	Branch
	branch and link	BL 2500	r14 = PC + 4; go to PC + 8 + 10000	For procedure call

data transfer instruction
A command that moves data between memory and registers.

address
A value used to delineate the location of a specifi c data element within a memory array.

Word
32-bit segment. That is, 4 bytes per word.

ARM Fields

ARM fields are given names to make them easier to discuss:

Cond	F	I	Opcode	S	Rn	Rd	Operand2
4 bits	2 bits	1 bit	4 bits	1 bit	4 bits	4 bits	12 bits

Here is the meaning of each name of the fields in ARM instructions:

■ Opcode: Basic operation of the instruction, traditionally called the opcode.

■ Rd: The register destination operand. It gets the result of the operation.

■ Rn: The first register source operand.

■ Operand2: The second source operand.

■ I: Immediate. If I is 0, the second source operand is a register. If I is 1, the second source operand is a 12-bit immediate. (Section 2.10 goes into details on ARM immediates, but for now we'll just assume it's just a plain constant.)

■ S: Set Condition Code. Described in Section 2.7, this field is related to conditional branch instructions.

■ Cond: Condition. Described in Section 2.7, this fi eld is related to conditional branch instructions.

■ F: Instruction Format. This fi eld allows ARM to different instruction formats when needed.

Instruction	Format	Cond	F	I	op	S	Rn	Rd	Operand2
ADD	DP	14	0	0	4_{ten}	0	reg	reg	reg
SUB (subtract)	DP	14	0	0	$2s_{ten}$	0	reg	reg	reg
ADD (immediate)	DP	14	0	1	4_{ten}	0	reg	reg	constant
LDR (load word)	DT	14	1	n.a.	24_{ten}	n.a.	reg	reg	address
STR (store word)	DT	14	1	n.a.	25_{ten}	n.a.	reg	reg	address

ARM machine language

Name	Format	Example								Comments
ADD	DP	14	0	0	4	0	2	1	3	ADD r1,r2,r3
SUB	DP	14	0	0	2	0	2	1	3	SUB r1,r2,r3
LDR	DT	14	1	24			2	1	100	LDR r1,100(r2)
STR	DT	14	1	25			2	1	100	STR r1,100(r2)
Field size		4 bits	2 bits	1 bit	4 bits	1 bit	4 bits	4 bits	12 bits	All ARM instructions are 32 bits long
DP format	DP	Cond	F	I	Opcode	S	Rn	Rd	Operand2	Arithmetic instruction format
DT format	DT	Cond	F	Opcode			Rn	Rd	Offset12	Data transfer format

FIGURE 2.6 ARM architecture revealed through Section 2.5. The two ARM instruction formats so far are DP and DT. The last 16 bits have the same sized fields: both contain an *Rn* field, giving one of the sources; and an *Rd* field, specifying the destination register.