3.10 Concluding remarks

Over the decades, computer arithmetic has become largely standardized, greatly enhancing the portability of programs. Two's complement binary integer arithmetic is found in every computer sold today, and if it includes floating point support, it offers the IEEE 754 binary floating-point arithmetic.

Computer arithmetic is distinguished from paper-and-pencil arithmetic by the constraints of limited precision. This limit may result in invalid operations through calculating numbers larger or smaller than the predefined limits. Such anomalies, called "overflow" or "underflow," may result in exceptions or interrupts, emergency events similar to unplanned subroutine calls. COD Chapters 4 (The Processor) and 5 (Large and Fast: Exploiting Memory Hierarchy) discuss exceptions in more detail.

Floating-point arithmetic has the added challenge of being an approximation of real numbers, and care needs to be taken to ensure that the computer number selected is the representation closest to the actual number. The challenges of imprecision and limited representation of floating point are part of the inspiration for the field of numerical analysis. The switch to **parallelism** will shine the searchlight on numerical analysis again, as solutions that were long considered safe on sequential computers must be reconsidered when trying to find the fastest algorithm for parallel computers that still achieves a correct result.

Data-level parallelism, specifically subword parallelism, offers a simple path to higher performance for programs that are intensive in arithmetic operations for either integer or floating-point data. We showed that we could speed up matrix multiply nearly fourfold by using instructions that could execute four floating-point operations at a time.

PARTICIPATION ACTIVITY 3.10.1: Computer arithmetic.

standard underflow numerical analysis

IEEE 754 is a _____ for floating-point representation and computation.

The field of _____ examines how to solve mathematical problems using imprecision and limited representation of data.

A situation where an operation yields a number that is too small to represent with the fixed number of bits available is called _____.

Reset

With the explanation of computer arithmetic in this chapter comes a description of much more of the LEGv8 instruction set. One point of confusion is the instructions covered in these chapters versus instructions executed by LEGv8 chips versus the instructions accepted by LEGv8 assemblers. Two figures try to make this clear.

The figure below lists the LEGv8 instructions covered in this chapter and COD Chapter 2 (Instructions: Language of the Computer). We call the set of instructions on the left-hand side of the figure the *LEGv8 core*. The instructions on the right we call the *LEGv8 arithmetic core*.

Figure 3.10.1: The LEGv8 instruction set (COD Figure 3.27).

This book concentrates on the instructions in the left column.

LEGv8 core instructions	Name	Format	LEGv8 arithmetic core	Name	Format
add	ADD	R	multiply	MUL	R
subtract	SUB	R	signed multiply high	SMULH	R
add immediate	ADDI	I	unsigned multiply high	UMULH	R
subtract immediate	SUBI	I	signed divide	SDIV	R
add and set flags	ADDS	R	unsigned divide	UDIV	R
subtract and set flags	SUBS	R	floating-point add single	FADDS	R
add immediate and set flags	ADDIS	I	floating-point subtract single	FSUBS	R
subtract immediate and set flags	SUBIS	I	floating-point multiply single	FMULS	R
load register	LDUR	D	floating-point divide single	FDIVS	R
store register	STUR	D	floating-point add double	FADDD	R
load signed word	LDURSW	D	floating-point subtract double	FSUBD	R
store word	STURW	D	floating-point multiply double	FMULD	R
load half	LDURH	D	floating-point divide double	FDIVD	R
store half	STURH	D	floating-point compare single	FCMPS	R
load byte	LDURB	D	floating-point compare double	FCMPD	R
store byte	STURB	D	load single floating-point	LDURS	D
load exclusive register	LDXR	D	load double floating-point	LDURD	D
store exclusive register	STXR	D	store single floating-point	STURS	D
move wide with zero	MOVZ	IM	store double floating-point	STURD	D
move wide with keep	MOVK	IM			
and	AND	R			
inclusive or	ORR	R			
exclusive or	EOR	R			
and immediate	ANDI	I			
inclusive or immediate	ORRI	I			
exclusive or immediate	EORI	I			
logical shift left	LSL	R			
logical shift right	LSR	R			
compare and branch on equal 0	CBZ	CB			
compare and branch on not equal 0	CBNZ	CB			
branch conditionally	B.cond	CB			
branch	B	B			
branch to register	BR	R			
branch with link	BL	B			

PARTICIPATION ACTIVITY 3.10.2: LEGv8 instructions.

Match the LEGv8 core instruction with the instruction name.

`SUBI` `SDIV` `MOVZ`

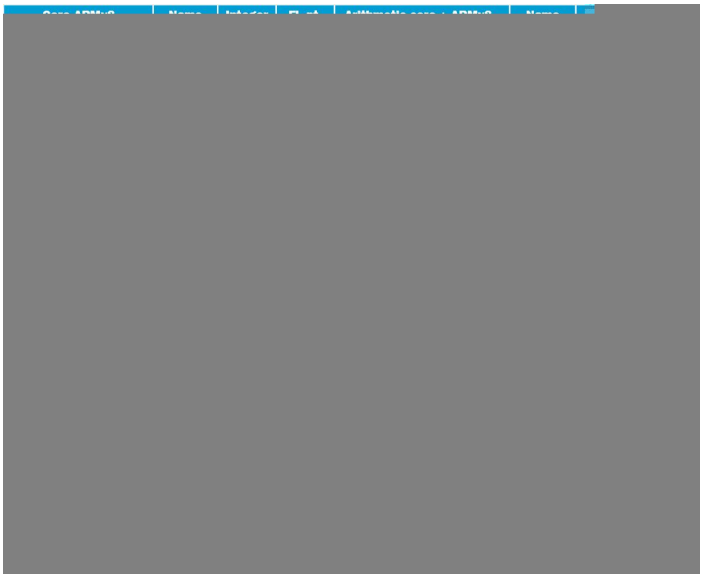
	Subtract immediate
	Move wide with zero
	Signed divide

Reset

The figure below gives the popularity of the LEGv8 instructions for SPEC CPU2006 integer and floating-point benchmarks. All instructions are listed that were responsible for at least 0.2% of the instructions executed.

Figure 3.10.2: The frequency of the LEGv8 instructions for SPEC CPU2006 integer and floating point (COD Figure 3.28).

All instructions that accounted for at least 0.2% of the instructions are included in the table. Pseudoinstructions are converted into LEGv8 before execution, and hence do not appear here.



Note that although programmers and compiler writers may use the full ARMv8 instruction set to have a richer menu of options, LEGv8 core instructions dominate integer SPEC CPU2006 execution, and the integer core plus arithmetic core dominate SPEC CPU2006 floating point, as the figure below shows.

Figure 3.10.3: The frequency of LEGv8 core, LEGv8 arithmetic core, and Remaining ARMv8 instructions for SPEC CPU2006 integer and floating point.

Instruction subset	Integer	Fl. pt.
LEGv8 core	98%	31%
LEGv8 arithmetic core	2%	66%
Remaining ARMv8	0%	3%

For the rest of the book, we concentrate on the LEGv8 core instructions—the integer instruction set excluding multiply and divide—to make the explanation of computer design easier. As you can see, the LEGv8 core includes the most popular LEGv8 instructions; be assured that understanding a computer that runs the LEGv8 core will give you sufficient background to understand even more ambitious computers. No matter what the instruction set or its size—ARMv8, ARMv7, MIPS, x86—never forget that bit patterns have no inherent meaning. The same bit pattern may represent a signed integer, unsigned integer, floating-point number, string, instruction, and so on. In stored-program computers, it is the operation on the bit pattern that determines its meaning.

PARTICIPATION ACTIVITY 3.10.3: Concluding remarks.

1) Which of the following instruction classifications occur most frequently in the execution of the SPEC CPU2006 floating-point benchmarks?
   - ○ LEGv8 core
   - ○ Remaining ARMv8

- Pseudo ARMv8

2) The leading 1 indicates that the bit pattern below represents a negative number.

```
1010 ... 0010 1100
```

- True
- False
- Not enough information to determine what the bit pattern represents.

Provide feedback on this section