Each line number represents an offset (or the number of bytes that an instruction is from the beginning of the current method). Notice that

```
Z = Max (X,Y);
```

gets compiled to the following bytecode:

```
14 iload_1
15 iload_2
16 invokestatic #3 <Method int Max(int, int)>
19 istore_3
```

---

It should be very obvious that Java bytecode is stack-based. For example, the `iadd` instruction pops two integers from the stack, adds them, and then pushes the result back to the stack. There is no such thing as "add r0, r1, f2" or "add AC, X". The `iload_1` (integer load) instruction also uses the stack by pushing slot 1 onto the stack (slot 1 in main contains *X*, so *X* is pushed onto the stack). *Y* is pushed onto the stack by instruction 15. The `invokestatic` instruction actually performs the `Max` method call. When the method has finished, the `istore_3` instruction pops the top element of the stack and stores it in *Z*.

We will explore the Java language and the JVM in more detail in Chapter 8.

## CHAPTER SUMMARY

The core elements of an instruction set architecture include the memory model (word size and how the address space is split), registers, data types, instruction formats, addressing, and instruction types. Even though most computers today have general-purpose register sets and specify operands by combinations of memory and register locations, instructions vary in size, type, format, and the number of operands allowed. Instructions also have strict requirements for the locations of these operands. Operands can be located on the stack, in registers, in memory, or a combination of the three.

Many decisions must be made when ISAs are designed. Larger instruction sets mandate longer instructions, which means a longer fetch and decode time. Instructions having a fixed length are easier to decode but can waste space. Expanding opcodes represent a compromise between the need for large instruction sets and the desire to have short instructions. Perhaps the most interesting debate is that of little versus big endian byte ordering.

There are three choices for internal storage in the CPU: stacks, an accumulator, or general-purpose registers. Each has its advantages and disadvantages, which must be considered in the context of the proposed architecture's applications. The internal storage scheme has a direct impact on the instruction format, particularly the number of operands the instruction is allowed to reference. Stack [illegible] zero operands, which fits well with RPN notation

Instructions are classified into the following categories: data movement, arithmetic, Boolean, bit manipulation, I/O, transfer of control, and special purpose. Some ISAs have many instructions in each category, others have very few in each category, and many have a mix of each. Orthogonal instruction sets are consistent, with no restrictions on the operand/opcode relationship.

The advances in memory technology, resulting in larger memories, have prompted the need for alternative addressing modes. The various addressing modes introduced included immediate, direct, indirect, register, indexed, and stack. Having these different modes provides flexibility and convenience for the programmer without changing the fundamental operations of the CPU.

Instruction-level pipelining is one example of instruction-level parallelism. It is a common but complex technique that can speed up the fetch–decode–execute cycle. With pipelining we can overlap the execution of instructions, thus executing multiple instructions in parallel. However, we also saw that the amount of parallelism can be limited by conflicts in the pipeline. Whereas pipelining performs different stages of multiple instructions at the same time, superscalar architectures allow us to perform multiple operations at the same time. Superpipelining, a combination of superscalar and pipelining, in addition to VLIW, was also briefly introduced. There are many types of parallelism, but at the computer organization and architecture level, we are really concerned mainly with ILP.

Intel and MIPS have interesting ISAs, as we have seen in this chapter as well as in Chapter 4. However, the Java Virtual Machine is a unique ISA, because the ISA is built-in software, thus allowing Java programs to run on any machine that supports the JVM. Chapter 8 covers the JVM in great detail.

## FURTHER READING

Instruction sets, addressing, and instruction formats are covered in detail in almost every computer architecture book. The books by Patterson and Hennessy (1997), Stallings (2006), and Tanenbaum (2006) all provide excellent coverage in these areas. Many books, such as Brey (2003), Messmer (1993), Abel (2001) and Jones (2001) are devoted to the Intel x86 architecture. For those interested in the Motorola 68000 series, we suggest Wray and Greenfield (1999) or Miller (1992).

Sohi (1990) gives a very nice discussion of instruction-level pipelining. Kaeli and Emma (1991) provide an interesting overview of how branching affects pipeline performance. For a nice history of pipelining, see Rau and Fisher (1993). To get a better idea of the limitations and problems with pipelining, see Wall (1993).

We investigated specific architectures in Chapter 4, but there are many important instruction set architectures worth mentioning. Atanasoff's ABC computer (Burks and Burks [1988], Von Neumann's EDVAC and Mauchly and Eckert's UNIVAC (Stern [1981] for information on both) had very simple instruction set architectures but required programming to be done in machine language. The Intel 8080

(a one-address machine) was the predecessor to the 80x86 family of chips introduced in Chapter 4. See Brey (2003) for a thorough and readable introduction to the Intel family of processors. Hauck and Dent (1968) provide good coverage of the Burroughs zero-address machine. Struble (1975) has a nice presentation of IBM's 360 family. Brunner (1991) gives details about DEC's VAX systems, which incorporated two-address architectures with more sophisticated instruction sets. SPARC (1994) provides a great overview of the SPARC architecture. Meyer and Downing (1991), Lindholm and Yellin, and Venner provide very interesting coverage of the JVM.

## REFERENCES


Abel, Peter. *IBM PC Assembly Language and Programming*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

Brey, B. *Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, and Pentium Pro Processor, Pentium II, Pentium III, and Pentium IV: Architecture, Programming, and Interfacing*, 6th ed. Englewood Cliffs, NJ: Prentice Hall, 2003.

Brunner, R. A. *VAX Architecture Reference Manual*, 2nd ed. Herndon, VA: Digital Press, 1991.

Burks, Alice, & Burks, Arthur. *The First Electronic Computer: The Atanasoff Story.* Ann Arbor, MI: University of Michigan Press, 1988.

Hauck, E. A., & Dent, B. A. "Burroughs B6500/B7500 Stack Mechanism." *Proceedings of AFIPS SJCC*:32, 1968, pp. 245–251.

Jones, William. *Assembly Language Programming for the IBM PC Family*, 3rd ed. El Granada, CA: Scott/Jones Publishing, 2001.

Kaeli, D., & Emma, P. "Branch History Table Prediction of Moving Target Branches Due to Subroutine Returns." *Proceedings of the 18th Annual International Symposium on Computer Architecture*, May 1991.

Lindholm, Tim, & Yellin, Frank. *The Java Virtual Machine Specification*. Online at *java.sun.com/docs/books/vmspec/html/VMSpecTOC.cod.html.*

Messmer, H. *The Indispensable PC Hardware Book*. Reading, MA: Addison-Wesley, 1993.

Meyer, J., & Downing, T. *Java Virtual Machine*. Sebastopol, CA: O'Reilly & Associates, 1991.

Miller, M. A. *The 6800 Family, Architecture Programming and Applications*, 2nd ed. Columbus, OH: Charles E. Merrill, 1992.

Patterson, D. A., & Hennessy, J. L. *Computer Organization and Design, The Hardware/Software Interface*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 1997.

Rau, B. Ramakrishna, & Fisher, Joseph A. "Instruction-Level Parallel Processing: History, Overview and Perspective." *Journal of Supercomputing* 7:1, Jan. 1993, pp. 9–50.

Sohi, G. "Instruction Issue Logic for High-Performance Interruptible, Multiple Functional Unit, Pipelined Computers." *IEEE Transactions on Computers*, March 1990.

SPARC International, Inc., *The SPARC Architecture Manual: Version 9*. Upper Saddle River, NJ: Prentice Hall, 1994.

Stallings, W. *Computer Organization and Architecture*, 7th ed. Upper Saddle River, NJ: P...