and speculative LOADs to gain speed. All in all, it may represent a significant advance over the Pentium 4, but it puts much of the burden of parallelization on the compiler.

PROBLEMS

1. A word on a little-endian computer has the numerical value of 3. If it is transmitted to a big-endian computer byte by byte and stored there, with byte 0 in byte 0, and so on, what is its numerical value on the big endian machine?
2. On the Pentium 4, instructions can contain any number of bytes, even or odd. On the UltraSPARC III, all instructions contain an integral number of words, that is, an even number of bytes. Give one advantage of the Pentium 4 scheme.
3. Design an expanding opcode to allow all the following to be encoded in a 36-bit instruction:

   7 instructions with two 15-bit addresses and one 3-bit register number
   500 instructions with one 15-bit address and one 3-bit register number
   40 instructions with no addresses or registers

4. A certain machine has 16-bit instructions and 6-bit addresses. Some instructions have one address and others have two. If there are n two-address instructions, what is the maximum number of one-address instructions?
5. Is it possible to design an expanding opcode to allow the following to be encoded in a 12-bit instruction? A register is 3 bits.

   4 instructions with three registers
   255 instructions with one register
   16 instructions with zero registers

6. Given the memory values below and a one-address machine with an accumulator, what values do the following instructions load into the accumulator?

   word 20 contains 40
   word 30 contains 50
   word 40 contains 60
   word 50 contains 70

   a. LOAD IMMEDIATE 20
   b. LOAD DIRECT 20
   c. LOAD INDIRECT 20
   d. LOAD IMMEDIATE 30
   e. LOAD DIRECT 30
   f. LOAD INDIRECT 30

7. Compare 0-, 1-, 2-, and 3-address machines by writing programs to compute

$$X = (A + B \times C) / (D - E \times F)$$

for each of the four machines. The instructions available for use are as follows:

0 Address	1 Address	2 Address	3 Address
PUSH M	LOAD M	MOV (X = Y)	MOV (X = Y)
POP M	STORE M	ADD (X = X+Y)	ADD (X = Y+Z)
ADD	ADD M	SUB (X = X–Y)	SUB (X = Y–Z)
SUB	SUB M	MUL (X = X*Y)	MUL (X = Y*Z)
MUL	MUL M	DIV (X = X/Y)	DIV (X = Y/Z)
DIV	DIV M		

M is a 16-bit memory address, and *X*, *Y*, and *Z* are either 16-bit addresses or 4-bit registers. The 0-address machine uses a stack, the 1-address machine uses an accumulator, and the other two have 16 registers and instructions operating on all combinations of memory locations and registers. SUB X,Y subtracts *Y* from *X* and SUB X,Y,Z subtracts Z from *Y* and puts the result in *X*. With 8-bit opcodes and instruction lengths that are multiples of 4 bits, how many bits does each machine need to compute *X*?

8. Devise an addressing mechanism that allows an arbitrary set of 64 addresses, not necessarily contiguous, in a large address space to be specifiable in a 6-bit field.

9. Give a disadvantage of self-modifying code that was not mentioned in the text.

10. Convert the following formulas from infix to reverse Polish notation.

a. A+B+C+D–E
b. (A–B)×(C+D)+E
c. (A×B)+(C×D)+E
d. (A–B)×(((C–D×E) / F) / G)×H

11. Which of the following pairs of reverse Polish notation formulas are mathematically equivalent?

a. AB+C+ and ABC++
b. AB–C– and ABC––
c. AB×C+ and ABC+×

12. Convert the following reverse Polish notation formulas to infix.

a. AB–C+D×
b. AB/CD/+
c. ABCDE+××/
d. ABCDE×F/+G–H/×+

13. Write three reverse Polish notation formulas that cannot be converted to infix.

14. Convert the following infix Boolean formulas to reverse Polish notation.

a. (A AND B) OR C
b. (A OR B) AND (A OR C)
c. (A AND B) OR (C AND D)

15. Convert the following infix formula to reverse Polish notation and generate IJVM code to evaluate it.

$$(5 \times 2 + 7) - (4 / 2 + 1)$$

16. The assembly language instruction

```
MOV REG,ADDR
```

means load a register from memory on the Pentium 4. However, on the UltraSPARC III, to load a register from memory one writes

```
LOAD ADDR,REG
```

Why is the operand order different?

17. How many registers does the machine whose instruction formats are given in Fig. 5-25 have?

18. In Fig. 5-25, bit 23 is used to distinguish the use of format 1 from format 2. No bit is provided to distinguish the use of format 3. How does the hardware know to use it?

19. It is common in programming for a program to need to determine where a variable X is with respect to the interval A to B. If a three-address instruction were available with operands A, B, and X, how many condition code bits would have to be set by this instruction?

20. The Pentium 4 has a condition code bit that keeps track of the carry out of bit 3 after an arithmetic operation. What good is it?

21. The UltraSPARC III has no instruction to load a 32-bit number into a register. Instead a sequence of two instructions, SETHI and ADD, are normally used. Is there more than one way to load a 32-bit number into a register? Discuss your answer.

22. One of your friends has just come bursting into your room at 3 A.M., out of breath, to tell you about his brilliant new idea: an instruction with two opcodes. Should you send your friend off to the patent office or back to the drawing board?

23. The 8051 does not have instructions with offsets longer than 8 bits. Does this mean it cannot address memory above 255? If it can, how does it do it?

24. Tests of the form

```
if (k == 0) ...
if (a > b) ...
if (k < 5) ...
```

are common in programming. Devise an instruction to perform these tests efficiently. What fields are present in your instruction?

25. For the 16-bit binary number 1001 0101 1100 0011, show the effect of:

a. A right shift of 4 bits with zero fill.
b. A right shift of 4 bits with sign extension.
c. A left shift of 4 bits.
d. A left rotate of 4 bits.
e. A right rotate of 4 bits.

26. How can you clear a memory word on a machine with no CLR instruction?

27. Compute the Boolean expression (*A* AND *B*) OR *C* for

A = 1101 0000 1010 0011
B = 1111 1111 0000 1111
C = 0000 0000 0010 0000

28. Devise a way to interchange two variables *A* and *B* without using a third variable or register. *Hint*: Think about the EXCLUSIVE OR instruction.

29. On a certain computer it is possible to move a number from one register to another, shift each of them left by different amounts, and add the results in less time than a multiplication takes. Under what condition is this instruction sequence useful for computing "constant × variable"?

30. Different machines have different instruction densities (number of bytes required to perform a certain computation). For the following Java code fragments, translate each one into Pentium 4 assembly language, UltraSPARC III assembly language, and IJVM. Then compute how many bytes each expression requires for each machine. Assume that *i* and *j* are local variables in memory, but otherwise make the most optimistic assumptions in all cases

a. i = 3;
b. i = j;
c. i = j – 1;

31. The loop instructions discussed in the text were for handling for loops. Design an instruction that might be useful for handling common while loops instead.

32. Assume that the monks in Hanoi can move 1 disk per minute (they are in no hurry to finish the job because employment opportunities for people with this particular skill are limited in Hanoi). How long will it take them to solve the entire 64-disk problem? Express your result in years.

33. Why do I/O devices place the interrupt vector on the bus? Would it be possible to store that information in a table in memory instead?

34. A computer uses DMA to read from its disk. The disk has 64 512-byte sectors per track. The disk rotation time is 16 msec. The bus is 16 bits wide, and bus transfers take 500 nsec each. The average CPU instruction requires two bus cycles. How much is the CPU slowed down by DMA?

35. Why do interrupt service routines have priorities associated with them whereas normal procedures do not have priorities?

36. The IA-64 architecture contains an unusually large number of registers (64). Was the choice to have so many of them related to the use of predication? If so, in what way? If not, why are there so many?

37. In the text, the concept of speculative LOAD instructions is discussed. However, there is no mention of speculative STORE instructions. Why not? Are they essentially the same as speculative LOAD instructions or is there another reason not to discuss them?

38. When two local area networks are to be connected, a computer called a bridge is inserted between them, connected to both. Each packet transmitted on either network causes an interrupt on the bridge, to let the bridge see if the packet has to be forwarded. Suppose that it takes 250 μsec per packet to handle the interrupt and inspect the packet, but forwarding it, if need be, is done by DMA hardware without burdening the CPU. If all packets are 1 KB, what is the maximum data rate on each of the networks that can be tolerated without having the bridge lose packets?

39. In Fig. 5-43, the frame pointer points to the first local variable. What information does the program need in order to return from a procedure?

40. Write an assembly language subroutine to convert a signed binary integer to ASCII.

41. Write an assembly language subroutine to convert an infix formula to reverse Polish notation.

42. The towers of Hanoi is not the only little recursive procedure much loved by computer scientists. Another all-time favorite is $n!$, where $n! = n(n-1)!$ subject to the limiting condition that $0! = 1$. Write a procedure in your favorite assembly language to compute $n!$.

43. If you are not convinced that recursion is at times indispensable, try programming the Towers of Hanoi without using recursion and without simulating the recursive solution by maintaining a stack in an array. Be warned, however, that you will probably not be able to find the solution.