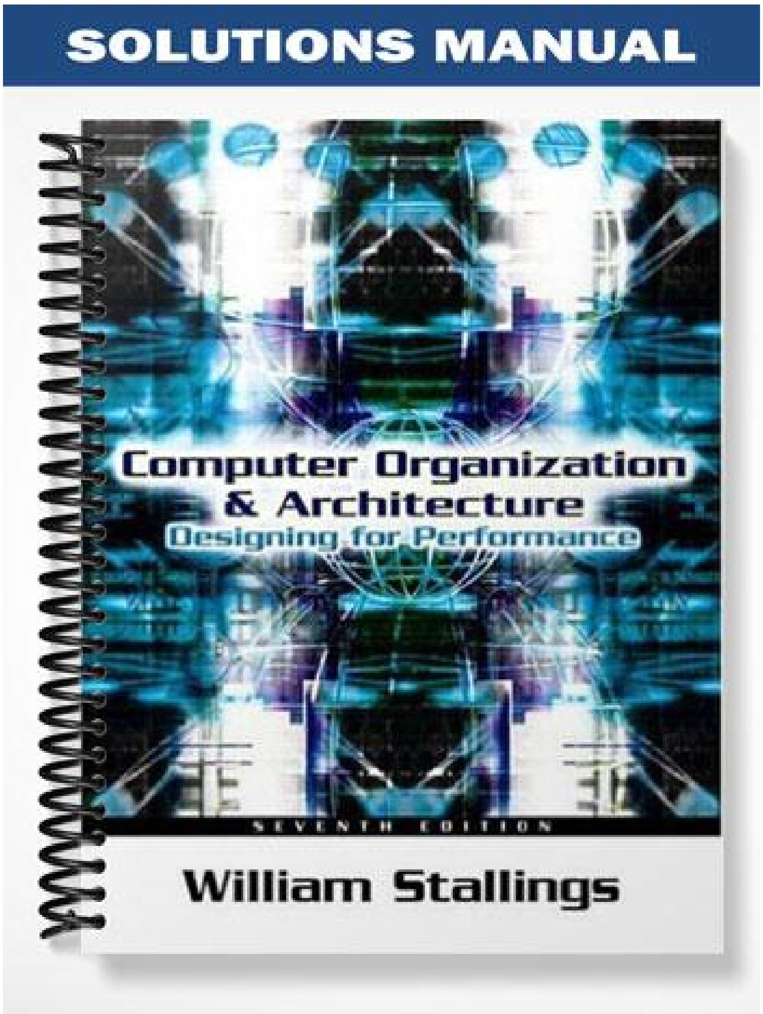

SOLUTIONS MANUAL

Computer Organization & Architecture

Designing for Performance

SEVENTH EDITION

William Stallings

# Chapter 10
# Instruction Sets: Characteristics and Functions

## Answers to Questions

**10.1** The essential elements of a computer instruction are the opcode, which specifies the operation to be performed, the source and destination operand references, which specify the input and output locations for the operation, and a next instruction reference, which is usually implicit.

**10.2** Registers and memory.

**10.3** Two operands, one result, and the address of the next instruction.

**10.4** **Operation repertoire:** How many and which operations to provide, and how complex operations should be. **Data types:** The various types of data upon which operations are performed. **Instruction format:** Instruction length (in bits), number of addresses, size of various fields, and so on. **Registers:** Number of CPU registers that can be referenced by instructions, and their use. **Addressing:** The mode or modes by which the address of an operand is specified.

**10.5** Addresses, numbers, characters, logical data.

**10.6** For the IRA bit pattern 011XXXX, the digits 0 through 9 are represented by their binary equivalents, 0000 through 1001, in the right-most 4 bits. This is the same code as packed decimal.

**10.7** With a **logical shift**, the bits of a word are shifted left or right. On one end, the bit shifted out is lost. On the other end, a 0 is shifted in. The **arithmetic shift** operation treats the data as a signed integer and does not shift the sign bit. On a right arithmetic shift, the sign bit is replicated into the bit position to its right. On a left arithmetic shift, a logical left shift is performed on all bits but the sign bit, which is retained.

**10.8** **1.** In the practical use of computers, it is essential to be able to execute each instruction more than once and perhaps many thousands of times. It may require thousands or perhaps millions of instructions to implement an application. This would be unthinkable if each instruction had to be written out separately. If a table or a list of items is to be processed, a program loop is needed. One sequence of instructions is executed repeatedly to process all the data. **2.** Virtually all programs involve some decision making. We would like the computer to do one thing if one condition holds, and another thing if another condition holds. **3.** To compose correctly a large or even medium-size computer program is an exceedingly difficult task. It helps if there are mechanisms for breaking the task up into smaller pieces that can be worked on one at a time.

**10.9** First, most machines provide a 1-bit or multiple-bit condition code that is set as the result of some operations. Another approach that can be used with a three-address instruction format is to perform a comparison and specify a branch in the same instruction.

**10.10** The term refers to the occurrence of a procedure call inside a procedure.

**10.11** Register, start of procedure, top of stack.

**10.12** A reentrant procedure is one in which it is possible to have several calls open to it at the same time.

**10.13** An assembly language uses symbolic names for addresses that are not fixed to specific physical addresses; this is not the case with machine language.

**10.14** In this notation, the operator follows its two operands.

**10.15** A multibyte numerical value stored with the most significant byte in the lowest numerical address is stored in **big-endian** fashion. A multibyte numerical value stored with the most significant byte in the highest numerical address is stored in **little-endian** fashion.

## Answers to Problems

**10.1** **a.** 23
**b.** 32 33

**10.2** **a.** 7309
**b.** 582
**c.** 1010 is not a valid packed decimal number, so there is an error

**10.3** **a.** 0; 255
**b.** –127; 127
**c.** –127; 127
**d.** –128; 127
**e.** 0; 99
**f.** –9; +9

**10.4** Perform the addition four bits at a time. If the 4-bit digit of the result of binary addition is greater then 9 (binary 1001), then add 6 to get the correct result.

| | | | | |
|---|---|---|---|---|
| 1698 | 0001 | 0110 | 1001 | 1000 |
| + 1798 | 0001 | 0111 | 1000 | 0110 |
| | 0010 | 1100 | 1 0001 | 1110 |
| | 1 | 1 | 1 | 0110 |
| | 0011 | 1110 | 0110 | 1 0100 |
| | | 0110 | 1000 | |
| | | 1 0100 | | |
| 3484 | 0011 | 0100 | 1000 | 0100 |

**10.5** The tens complement of a number is formed by subtracting each digit from 9, and adding 1 to the result, in a manner similar to twos complement. To subtract, simply take the tens complement and add:

```
0736
9674
   1     0410
```

**10.6**

| | | | |
|---|---|---|---|
| PUSH A | LOAD E | MOV R0, E | MUL R0, E, F |
| PUSH B | MUL F | MUL RO, F | SUB R0, D, R0 |
| PUSH C | STORE T | MOV R1, D | MUL R1, B, C |
| MUL | LOAD D | SUB R1, R0 | ADD R1, A, R1 |
| ADD | SUB T | MOV R0, B | DIV X, R0, R1 |
| PUSH D | STORE T | MOV R0, C | |
| PUSH E | LOAD B | ADD R0, A | |
| PUSH F | MUL C | DIV R0, R1 | |
| MUL | ADD A | MOV X, R0 | |
| SUB | DIV T | | |
| DIV | STO X | | |
| POP X | | | |

Source: [TANE90]

**10.7** **a.** A memory location whose initial contents are zero is needed for both X → AC and AC → X. The program for X → AC, and its effects are shown below. Assume AC initially contains the value a.

| Instruction | AC | Effect on M(0) | M(X) |
|---|---|---|---|
| SUBS 0 | a | a | x |
| SUBS 0 | 0 | 0 | x |
| SUBS X | –x | 0 | –x |
| SUBS 0 | –x | –x | –x |
| SUBS 0 | 0 | 0 | –x |
| SUBS X | x | 0 | x |

**b.** For addition, we again need a location, M(0), whose initial value is 0. We also need destination location, M(1). Assume the initial value in M(1) is y.