1.1.3

1. How do assemblers and linkers work together?

Answer: An assembler converts source-code programs from assembly language into machine language, which is referred to as object-code. A linker combines individual files created by an assembler into a single executable program.

1. What is meant by a one-to-many relationship when comparing a high-level language to machine language?

Answer: In a one-to-many relationship, a single statement expands into many assembly language or machine instructions.

8. Do you suppose type checking on pointer variables is stronger (stricter) in assembly language, or in C and C++?

Answer: C++ does not allow a pointer of one type to be assigned to a pointer of another type. On the other hand, Assembly language has no restrictions regarding pointers.

12. Challenge: Translate the following C++ expression to assembly language, using the example presented earlier in this chapter as a guide: X (Y \* 4) 3.

Answer:

mov eax, Y  
mov ebx, 4  
imul, ebx  
add eax, 3  
mov X, eax

1.2.1

2. Why do you suppose translated programs often execute more quickly than interpreted ones?

Answer: Translated programs often execute more quickly than interpreted ones. This is because the translated programs are now converted into a language that can be understood by the machine, and executed with no hassle. On the other hand, interpreted programs are translated and executed simultaneously, causing them to be executed less quickly.

3. (True/False): When an interpreted program written in language L1 runs, each of its instructions is decoded and executed by a program written in language L0.

Answer: True

10. Statements at the assembly language level of a virtual machine are translated into statements at which other level?

Answer: 2nd level

1.3.9

2. What is the decimal representation of each of the following unsigned binary integers?

a. 11111000

b. 11001010

c. 11110000

Answer:

a. 248

b. 202

c. 240

3. What is the sum of each pair of binary integers?

a. 00001111 + 00000010

b. 11010101 + 01101011

c. 00001111 + 00001111

Answer:

a. 00010001

b. 101000000

c. 00011110

5. What is the minimum number of binary bits needed to represent each of the following unsigned decimal integers?

a. 65

b. 409

c. 16385

Answer:

a. 7

b. 9

c. 15

6. What is the hexadecimal representation of each of the following binary numbers?

a. 0011 0101 1101 1010

b. 1100 1110 1010 0011

c. 1111 1110 1101 1011

Answer:

a. 35DA

b. CEA3

c. FEDB

7. What is the binary representation of the following hexadecimal numbers?

a. A4693FBC

b. B697C7A1

c. 2B3D9461

Answer:

a. 1010 0100 0110 1001 0011 1111 1011 1100

b. 1011 0110 1001 0111 1100 0111 1010 0001

c. 0010 1011 0011 1101 1001 0100 0110 0001

1.4.2

1. **X not(X) Y (not X) or Y**

F T F T

F T T T

T F F F

T F T T

Take the opposite of whatever the Boolean value of X is. If X or Y is true, so is the expression. The expression can only have a false value if both X and Y are false.

2. **X Y X AND Y**

F F F

T F F

F T F

T T T

The expression is false if even one of the variables are false as well. This is also the case if both variables are false. The expression can only be true if both variables are true.

3. T

1. F
2. T

1.7.1

1. Most significant bit (the highest numbered bit).

2. (a) 53

(b) 150

(c) 204

3. (a) 110001010

(b) 110010110

(c) 100100001

4. 00000110

5. (a) 8

(b) 32

(c) 64

(d) 128

6. (a) 12

(b) 16

(c) 16

7. (a) 35DA

(b) CEA3

(c) FEDB

8. (a) 0000 0001 0010 0110 1111 1001 1101 0100

(b) 0110 1010 1100 1101 1111 1010 1001 0101

(c) 1111 0110 1001 1011 1101 1100 0010 1010

9. (a) 58 (b) 447 (c) 16534

10. (a) 98 (b) 1203 (c) 671

11. (a) FFE8 (b) FEB5

12. (a) FFEB (b) FFD3

13. (a) 27641 (b) −16093

14. (a) 19666 (b) −32208

15. (a) −75 (b) +42 (c) −16

16. (a) −128 (b) −52 (c) −73

17. (a) 11111011 (b) 11010110 (c) 11110000

18. (a) 10111000 (b) 10011110 (c) 11100110

19. (a) AB2 (b) 1106

20. (a) B82 (b) 1316

21. 42h and 66d

22. 47h and 71d

25. **A B A (or) B not (A or B)**

F F F T

F T T F

T F T F

T T T F

1.7.2

Question 1:

int stringToInt(string s)

{

int num = 0;

for(int i = 0; ; i++)

{

If ( s[i] >= '0' && s[i] <= '9' )

{

num = num \* 16 + s[i]-'0';

}

else if ( s[i] >= 'A' && s[i] <= 'F' )

{

num = num \* 16 + (s[i]-'A'+10);

}

else

{

break;

}

}

return num;

}