Chapter 5 Representing Numerical Data

1) How do computers store all data and program instructions?

a) As decimal numbers.

b) As ASCII characters

**c) As binary numbers.**

d) As algebraic equations.

2) The binary numbers in a computer might represent

c) images.

b) numbers.

a) characters.

**d) All of the above**

3) What numbers are generally manipulated as characters?

a) Zip code.

b) Telephone number.

c) Grade point average.

**d) Both a and c.**

4) When the number to be expressed is outside of the integer range of the computer (too large or too small), or when the number contains a fractional part it must be stored as a(n)

a) constant.

b) exponent.

c) complement.

**d) real number.**

5) An 8-bit storage location can store any unsigned integer of value between 0 and

a) 7

b) 16

**c) 255**

d) 512

6) What does BCD stand for?

**a) Binary-Coded Decimal**

b) Binary Calculating Device

c) Binary Common Denominator

d) Binary Character Data

7) What is the range of a 1 byte number stored in BCD format?

a) 0-9

**b) 0-99**

c) 0-999

d) 0-9999

8) How many BCD digits can be stored in one byte?

a) 1

**b) 2**

c) 7

d) 255

9) What is the most common way to represent negative integers in binary form?

a) As BCD

**b) Using 2's complement**

c) Using sign-and-magnitude

d) None of the above

10) If we complement the value twice, it will

a) be twice as big.

**b) return to its original value.**

c) cause an overflow error.

d) reset the carry flag.

11) A combination of numbers that produces a result outside the available range is known as

a) overload.

**b) overflow.**

c) spillover.

d) wraparound.

12) Changing every 0 to a 1 and every 1 to a 0 is also known as

a) reversion.

**b) inversion.**

c) diversion.

d) conversion.

13) Using sign-and-magnitude representation, the largest positive number that can be stored in 8 bits is

a) 7

**b) 127**

c) 255

d) 512

14) Using sign-and-magnitude representation, if the leftmost bit is 1 the number is

a) positive.

**b) negative.**

c) an error.

d) a character

15) If both inputs to an addition have the same sign, and the output sign is different then

a) the leftmost bit should wrap around.

b) the leftmost bit should be disregarded.

**c) the range is insufficient to hold the result.**

d) you must take the complement of the result.

16) Using sign-and-magnitude representation, storing the number -12 in 4 bits is

a) 1100

b) 0011

c) 0100

**d) impossible.**

17) In 1's and 2’s complement representations, a negative number begins with

a) -1

b) 0

**c) 1**

d) -0

18) How do you find the 2's complement of positive numbers?

a) Invert the numbers

**b) Invert the numbers and add one**

c) Invert the numbers and wrap around the leftmost bit

d) Do nothing, the complement is the same as the original

19) How do you find the 2's complement of negative numbers?

a) Invert the numbers.

**b) Invert the numbers and add one.**

c) Invert the numbers and wrap around the leftmost bit.

d) Do nothing, the complement is the same as the original.

20) When adding two numbers using 2's complement, carries beyond the leftmost digit are

a) inverted.

**b) ignored.**

c) shifted left.

d) shifted right.

21) What is the 8-bit 2’s complement representation for -35?

**a) 11011101**

b) 01011101

c) 11011100

d) 11011111

22) To correct for carries and borrows that occur when large numbers must be separated into parts to perform additions and subtractions, we use

a) a bit hold.

**b) a carry flag.**

c) an error flag.

d) an overflow flag.

23) What is the number 12.345 x 10 2 without using exponential notation?

a) 0.12345

b) 123.45

**c) 1234.5**

d) 12345

24) In excess-50 notation, an exponent can range from

a) 0 to 50

**b) -50 to 49**

c) -49 to 50

d) -99 to 99

25) In excess-50 notation, an exponent equaling 17 is stored as

a) -37

b) 17

**c) 67**

d) 87

26) Shifting numbers left and increasing the exponent until leading zeros are eliminated is called

a) conversion.

b) factorization.

**c) normalization.**

d) excess notation.

27) The leftmost bit in an IEEE standard floating point number represents

a) the exponent.

b) the mantissa.

**c) the sign of the mantissa.**

d) the sign of the exponent.

28) The exponent of a floating point number is stored using

**a) excess N notation.**

b) one's complement.

c) two's complement.

d) binary coded decimal.

29) In the IEEE 754 standard 32 bit single-precision floating point format, how many bits are allocated to the exponent?

a) 1

b) 2

c) 7

**d) 8**

29) In the IEEE 754 standard 32 bit single-precision floating point format, how many bits are allocated to the mantissa?

a) 8

b) 16

**c) 23**

d) 24

31) With floating point numbers, to add and subtract require that

a) the sign in each number be equal.

b) the mantissa in each number be equal.

**c) the exponents in each number be equal.**

d) you convert to decimal to perform the operation.

32) In the normalized IEEE 754 standard 32 bit single-precision floating point format, the leading bit of the mantissa is

a) the sign bit.

**b) not stored.**

c) always a zero.

d) used for larger exponents.

33) In the IEEE 754 standard 32 bit single-precision floating point format the number "zero" is

a) impossible to represent.

**b) treated as a special case.**

c) has a mantissa of all ones.

d) has an exponent of all ones.

34) The Department of Motor Vehicles is developing a software program that uses a variable to count the number of cars being sold each day. The maximum number of cars sold is expected to be several million over the life of the program and there are no fractional sales. What data type will you use for this variable?

a) float 32-bit

b) float 64-bit

**c) integer long 64-bit**

d) integer short 32-bit

35) Calculations with floating point numbers

a) are faster than integer calculations.

b) are more precise than integer calculations.

c) typically require less storage space than integers for the results.

**d) None of the above.**

Discussion Questions

1) Calculate largest positive integer that can be stored in 2 bytes using a) an unsigned integer

b) an integer using sign and magnitude, and c) an unsigned BCD number.

**Sol: This is useful: The relationship between largest positive integer and the number (n) of bits used is**

***largest positive number using n bits = 2^n-1***

**a) Largest positive integer = 2^16 – 1**

**= 65536 – 1**

**= 65,535**

**b) In two bytes there are 16 bits, of those 15 are available for positive numbers:**

**= 2^15 – 1**

**= 32768 – 1**

**= 32,767**

**c) BCD uses 4 bits per digit, so we can use up to 4 digits: 9999**

 **(9999 = 1001 1001 1001 1001)**

2) How does overflow occur when using a fixed word size to represent integers?

**Sol: from the text: “A fixed word size results in a range of some particular fixed size; it is always possible to have a combination of numbers that produces a result outside the range. This condition is known as overflow.”**

3) The 9’s complementary operation.

a) Find the 9’s complementary representation for the *three*-digit number -351.

b) Find the 9’s complementary representation for the *four*-digit number -351.

**Sol:**

**a) In three-digits, the 9’s complementary representation is**

 **999**

**- 351**

**-----------**

 **648**

**b) In four-digits, the 9’s complementary representation is**

 **9999**

**- 351**

**-----------**

 **9648**

4) The 9’s complementary representation.

a) What is the sign-and-magnitude value of the three-digit number represented in 9’s complement by 461?

b) What is the sign-and-magnitude value of the three-digit number represented in 9’s complement by 747?

**Sol:**

**a) In 9’s complement, numbers 0 to 499 are positive, hence 461 is +461.**

**b) In 9’s complement, numbers 500 to 998 are negative, so we find the complement:**

 **999**

**-747**

**-------------**

**252**

**Adjust sign: answer is -252**

5) Two's complement

a) Find the 2’s complementary representation in 8-bits of the number -51.

b) Find the 2’s complementary representation in 16-bits of the number -331.

**Sol:**

**a) Step 1: 51 in binary using 8-bits is 00110011**

**Step 2: Flip the bits: 11001100**

**Step 3: add 1: 11001101**

**b) Step 1: 331 in binary using 16-bits is 0000 0001 0100 1011**

**Step 2: Flip the bits: 1111 1110 1011 0100**

**Step 3: add 1: 1111 1110 1011 0101**

6) Two's complement.

a) What is the decimal sign-and-magnitude value of the 8-bit number represented in 2's complement by 11010011?

b) What is the decimal sign-and-magnitude value of the 8-bit number represented in 2's complement by 01000011?

**Sol:**

**a) Since the leftmost digit is 1, the number is negative. So, invert the number and add one:**

**11010011 --> 00101100 + 1 = 45. Include the negative sign -45.**

**b) Since the leftmost digit is 0, the number is positive. So, just convert to decimal:**

**01000011 (base 2) = 67 (base 10).**

7) The exponent for a given number is negative 25. What value is stored to represent the exponent in excess-50 notation?

**Sol: In excess-50 notation, 50 is the midvalue or offset. So, we subtract 25 from 50 = 25.**

8) What purpose is served by "normalizing" the mantissa? That is—given the extra work—

what is gained?

**Sol: Moving the decimal point such that there are no leading zeros ensures the fewest number of significant digits are truncated, which maximizes the precision.**

**From the text: “To maximize the precision for a given number of digits, numbers will be stored whenever possible with no leading zeros. This means that, when necessary, numbers are shifted left by increasing the exponent until leading zeros are eliminated. This process is called normalization.”**

*In problems 9-13 use the floating point format SEEMMMMM, where S = 0 is "+" and 5 is "-"; EE is the exponent in excess-50, and MMMMM are five digits of mantissa.*

9) Show how 12.25 is represented in SEEMMMMM format.

**Sol:**

**12.25 = .1225 x 10^2**

**S=0**

**EE=52**

**MMMMM = 12250, where the added zero is added to make five digits**

**Answer: 05212250**

10) Show how -0.00012325 is represented in SEEMMMMM format.

**Sol:**

**0.00012325 = .1225 x 10^-4**

**S=5**

**EE=46**

**MMMMM = 12325**

**Answer: 54612325**

11) Convert 55698799 in SEEMMMMM to a decimal integer without exponents.

**Sol:**

**5 = S; sign is negative**

**56 = EE; exponent is 6**

**98799 = MMMMM**

**- .98799 x 10^6 = -987990.**

12) Add the two numbers 04933460 and 04877456 (in SEEMMMMM format.) Represent your answer in decimal form without using exponents.

**Sol:**

**04933460 ---> +.33460 x 10 ^-1 or 0.033460**

**04877456 ---> +.77456 x10 ^-2 or 0.0077456**

**0.033460 + 0.0077456 = 0.0412056**

13) Multiply the two numbers 05312675 and 05144359 (in SEEMMMMM format.) Leave the answer in SEEMMMMM rounding if necessary.

**Sol:**

**05312675 --> +.12675 x 10^3 = 126.75**

**05144359 --> +.44359 x 10^1 = 4.4359**

**126.75 x 4.4359 = 562.250325**

**562.250325 = .562250325 x 10^3 = .56225 x 10^3 rounded to five decimal places**

**Build the SEEMMMMM**

**05356225**

14) What are the trade-offs between integer and floating calculations?

**Sol: From the text: “The trade-offs between integer and floating point are clear. Integer calculations are easier for the computer to perform, have the potential to provide higher precision, and are obviously much faster to execute. Integer values usually take up fewer storage locations. As you will see later, it takes a certain amount of time to access each storage location; thus, the use of fewer storage locations saves time, as well as space.”**

15) Suppose you are writing a program that needs to represent a maximum 50,000 whole things (i.e. integer data type). What would be better: to use a short integer (16 bits) or long integer (64 bits)? Why?

**Sol:**

**First check that 16 bits is sufficient: 2^16 = 65,536 greater than 50,000, so it possible to use short integer.**

**From the text: “The longer integer formats may require multiple-word calculation algorithms, and as such are slower to execute than short formats. The short format is preferable when it is sufficient for the values that you expect. It may also be necessary to consider the limitations of other systems that the same program may have to operate on.”**

|  |  |  |
| --- | --- | --- |
| Problem | Answer | Section in text / comments |
| 1 | c | Section 5.0 Introduction |
| 2 | d | Section 5.0 Introduction |
| 3 | d | Section 5.0 Introduction |
| 4 | d | Section 5.0 Introduction |
| 5 | c | Section 5.1 Unsigned Binary and Binary-Coded Decimal Representations |
| 6 | a | Section 5.1 Unsigned Binary and Binary-Coded Decimal Representations |
| 7 | b | Section 5.1 Unsigned Binary and Binary-Coded Decimal Representations |
| 8 | b | Section 5.1 Unsigned Binary and Binary-Coded Decimal Representations |
| 9 | b | Section 5.2 Representations for Signed Integers |
| 10 | b | Section 5.2 Representations for Signed Integers |
| 11 | b | Section 5.2 Representations for Signed Integers |
| 12 | b | Section 5.2 Representations for Signed Integers |
| 13 | b | Section 5.2 Representations for Signed Integers |
| 14 | b | Section 5.2 Representations for Signed Integers |
| 15 | c | Section 5.2 Representations for Signed Integers |
| 16 | d | Section 5.2 Representations for Signed Integers |
| 17 | c | Section 5.2 Representations for Signed Integers |
| 18 | b | Section 5.2 Representations for Signed Integers |
| 19 | b | Section 5.2 Representations for Signed Integers |
| 20 | b | Section 5.2 Representations for Signed Integers |
| 21 | a | Section 5.2 Representations for Signed Integers |
| 22 | b | Section 5.2 Representations for Signed Integers |
| 23 | c | Section 5.3 Real Numbers |
| 24 | b | Section 5.3 Real Numbers |
| 25 | c | Section 5.3 Real Numbers |
| 26 | c | Section 5.3 Real Numbers |
| 27 | c | Section 5.3 Real Numbers |
| 28 | a | Section 5.3 Real Numbers |
| 29 | d | Section 5.3 Real Numbers |
| 30 | c | Section 5.3 Real Numbers |
| 31 | c | Section 5.3 Real Numbers |
| 32 | b | Section 5.3 Real Numbers |
| 33 | b | Section 5.4 Programming Considerations |
| 34 | c | Section 5.4 Programming Considerations |
| 35 | d | Section 5.4 Programming Considerations |