Introduction to Computer Science-103

Midterm

1. Convert the following hexadecimal and octal numbers to decimal without using a calculator, showing your work. (6%)

   a. $(ABC.D)_{16}$ 2748.8125

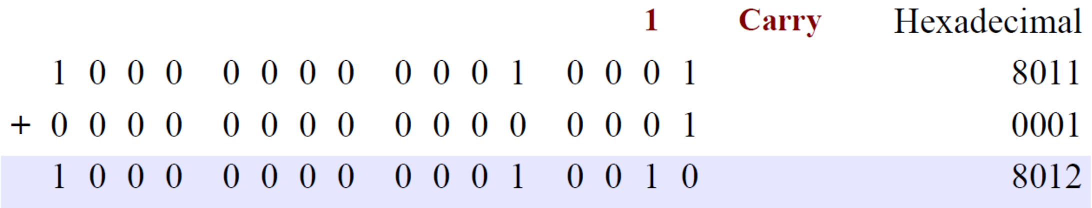
   b. $(411)_{8}$ 265

2. A step-by-step solution to a problem is called a(n) _algorithm_ . (3%)

3. In floating-point representation of real numbers in 32-bit IEEE format, _Mantissa_ is the bit sequence to the right of the decimal point after normalization. (3%)

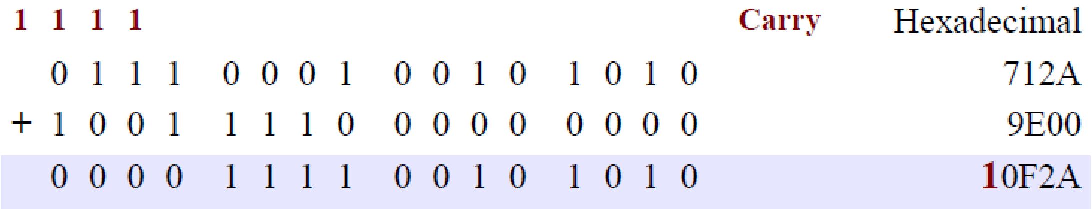
4. Show the result of the following operations assuming that the numbers are stored in 16-bit two's complement representation. Show the result in hexadecimal notation. (6%)

   a. $(8011)_{16} + (0001)_{16}$



															1		Carry	Hexadecimal
	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1		8011
+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		0001
	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0		8012

   b. $(712A)_{16} + (9E00)_{16}$



1	1	1	1														Carry	Hexadecimal
	0	1	1	1	0	0	0	1	0	0	1	0	1	0	1	0		712A
+	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0		9E00
	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0		10F2A

   Note that the result is not valid because of overflow.

5. Show how the computer finds the result of the following floating-point operations. (12%)

   a. (+5.75) + (-7.0234375) = -1.2734375.

These two numbers can be stored in floating-point format, as shown below:

	S	E	M
A	0	10000001	01110000000000000000000
B	1	10000001	11000001100000000000000

De-normalization results in:

	S	E	Denormalized M
A	0	10000010	101110000000000000000000
B	1	10000010	111000001100000000000000

Alignment is not needed (both exponents are the same), so we apply addition operation on the combinations of sign and mantissa. The result is shown below, in which the sign of the result is negative:

	S	E	Denormalized M
R	1	10000010	$\boxed{001}$010001100000000000000

Now we need to normalize. We decrement the exponent three times and shift the de-normalized mantissa to the left three positions:

	S	E	M
R	1	01111111	010001100000000000000000

The mantissa is now 24 bits, so we round it to 23 bits.

	S	E	M
R	1	01111111	01000110000000000000000

The result is R = $-\ 2^{127-127} \times 1.0100011 = -$ **1.2734375**, as expected.

b. 33.1875 − 0.4375

$33.1875 - 0.4375 = (100001.0011)_2 - (0.0111)_2 = 2^5 \times (1.000010011)_2 - 2^{-2} \times (1.11)_2$. These two numbers are stored in floating-point format as shown, but we need to remember that each number has a hidden 1 (which is not stored, but assumed). $E_1 = 127 + 5 = 132 = (10000100)_2$ and $E_2 = 127 + (-2) = 125 = (01111101)_2$

	S	**E**	**M**
A	0	10000100	00001001100000000000000
B	0	01111101	11000000000000000000000

The first two steps in UML diagram is not needed. Since the operation is subtraction, we change the sing of the second number.

	S	**E**	**M**
A	0	10000100	00001001100000000000000
B	1	01111101	11000000000000000000000

We denormalize the numbers by adding the hidden 1's to the mantissa and incrementing the exponent. Now both denormalized mantissas are 24 bits and include the hidden 1's. They should store in a location to hold all 24 bits. Each exponent is incremented.

	S	**E**	**Denormalized M**
A	0	10000101	**1**00001001100000000000000
B	1	01111110	**1**11000000000000000000000

We align the mantissas. We increment the second exponent by 7 and shift its mantissa to the right seven times.

	S	**E**	**Denormalized M**
A	0	10000101	**1**00001001100000000000000
B	1	10000101	0000000**1**1100000000000000

Now we do sign-and-magnitude addition treating the sign and the mantissa of each number as one integer stored in sign-and-magnitude representation.

	S	**E**	**Denormalized M**
R	0	10000101	100000110000000000000000

There is no overflow in mantissa, so we normalized.

	S	**E**	**M**
R	0	10000100	00000110000000000000000

The mantissa is only 23 bits because there is no overflow, no rounding is needed.

$$E = (10000100)_2 = 132,\ M = 0000011$$

The result is

$$(1.0000011)_2 \times 2^{132-127} = (100000.11)_2 = 32.75$$

6. A number less than b^k can be represented using k digits in base b. Show the

number of digits needed in this case: (6%)

a. Integers less than 16^{14} in binary

$\log_2 16^{14} = 56$

b. Integers less than 10^8 in decimal

$\log_{10} 10^8 = 8$

7. Change the following decimal numbers to 16-bit two’s complement integers. (6%)

a. -153 1111111101100111

b. 74256 overflow

8. Show the result of the following operations. (6%)

a. $[(99)_{16}$ AND $(33)_{16}]$ OR $[(00)_{16}$ AND $(FF)_{16}]$

$[(99)_{16}$ AND $(33)_{16}]$ OR $[(00)_{16}$ AND $(FF)_{16}] =$

$[(10011001)_2$ AND $(00110011)_2]$ OR $[(00000000)_2$ AND $(11111111)_2] =$

$(00010001)_2$ OR $(00000000)_2 = (00010001)_2 = (11)_{16}$

b. $(99)_{16}$ OR [NOT $(00)_{16}]$

$(99)_{16}$ OR [NOT $(00)_{16}] = (10011001)_2$ OR [NOT $(00000000)_2] =$

$(10011001)_2$ OR $(11111111)_2 = (11111111)_2 = (FF)_{16}$

9. Convert the following numbers in 32-bit IEEE format. (6%)

a. – 0.375

$-0.375 = -0.011 = -2^{-2} \times 1.1$

S = 1

$E = -2 + 127 = 125 = (01111101)_2$

M = 1 (plus 22 zero at the right)

→ 1 01111101 10000000000000000000000

b. 7.1875

$7.1875 = (111.0011)_2 = 2^2 \times 1.110011$

S = 0

E = 2 + 127 = 129 = $(10000001)_2$

M = 110011 (plus 17 zero at the right)

→ 0 10000001 11001100000000000000000

10. In which of the following situations does an overflow never occur? (4%)

    a. Adding two positive integers.

    b. Adding one positive integer to a negative integer.

    c. Subtracting one positive integer from a negative integer.

    d. Subtracting two negative integers.

    We assume that both operands are in the presentable range.

    a. Overflow can occur because the magnitude of the result is greater than the magnitude of each number and could fall out of the presentable range.

    b. Overflow does not occur because the magnitude of the result is smaller than one of the numbers; the result is in the presentable range.

    c. When we subtract a positive integer from a negative integer, the magnitudes of the numbers are added. This is the negative version of case *a*. Overflow can occur.

    d. When we subtract two negative numbers, the magnitudes are subtracted from each other. This is the negative version of case *b*. Overflow does not occur.

11. We need to unset (force to 0) the four rightmost bits and set (force to 1) the three leftmost bits of a pattern. Show the masks and the operations. (6%)

    Mask1= $(11110000)_2$ Mask2 = $(11100000)_2$

    Operation: [Mask1 AND $(xxxxxxxx)_2$] OR Mask2 = $(111x0000)_2$

12. A computer uses memory-mapped I/O addressing. The address bus uses 10 lines (10 bits). If memory made up of 1,000 words, how many three-register controllers can be accessed by the computer. (6%)?

    The address bus uses 10 lines which means that it can address 2^{10} = 1024 words. Since the memory is made of 1000 words and the system uses shared (memory-mapped I/O) addressing, 1024 − 1000 = 24 words are available for I/O controllers. If each controller has 3 registers, then 24/3 = 8 controllers can be accessed in this system.

13. Match the following to one or more layers of the TCP/IP protocol suite. (6%)

a. providing services for the end user

The application layer provides services for the end users.

b. route determination

The network layer is responsible for route determination.

14. An imaginary computer has sixteen data registers (R0 to R15), 4096 words in memory, and 16 different instructions (add, subtract, and so on). What is the minimum size of an instruction in bits if a typical instruction uses the following format: **Instruction** M R2. (6%)

4 + 4 + 12 = 20

15. 32 bits are used to represent an address, eight bits for each symbol in dotted decimal notation. For example, the address 10.200.14.72 can also be represented as 00001010 11001000 00001110 01001000. Show the bit representation of the following Internet addresses. (6%)

a. 110.11.5.88 01101110 00001011 00000101 01011000

b. 12.74.16.18 00001100 01001010 00010000 00010010

16. What does a technique called pipelining mean? (6%)

Pipelining allows different types of phases belonging to different cycles to be done simultaneously. Pipelining can increase the throughput of the computer.

17. In the TCP/IP protocol suite, please distinguish between communication at the network layer and communication at the data-link layer. (6%)

Communication at the network layer is host-to-host.

Communication at the data-link layer is node-to-node.