1. Answer the following questions:
   1. Convert 3.7 seconds into milliseconds

1 second = 103millisecond

Therefore 3.7 seconds = 3.7x103 milliseconds=3700 milliseconds

* 1. How many nanoseconds are in 3 microseconds?

1 second = 109 nanosecond, 1 second = 106 microsecond

Therefore 109 nanosecond = 106 microseconds

Therefore 103 nanosecond = 1 microseconds

Therefore 3 microseconds = 3x103 nanoseconds = 3000 nanoseconds

* 1. How many megabytes (MB) are in 1.3 gigabytes (GB)?

1 MB = 106 B, 1 GB = 109 B

Therefore 1 GB = 103 MB

Therefore 1.3 GB = 1.3 x 103 MB = 1300 MB

1. In the von Neumann model, explain the purpose of the:
   1. Processing unit: performs all the arithmetic and logic functions.
   2. Program counter: holds the address of the next instruction to be executed in a program.
2. Briefly explain the three main types of Cloud computing platforms.
   1. Infrastructure-as-a-service (IaaS): used for Internet-based access to storage and computing power. It lets you rent IT infrastructure servers and virtual machines, storage, networks, and operating systems from a cloud provider on a pay-as-you-go basis.
   2. Platform-as-a-service (PaaS): gives developers the tools to build and host web applications. PaaS is designed to give users access to the components they require to quickly develop and operate web or mobile applications over the Internet, without worrying about setting up or managing the underlying infrastructure of servers, storage, networks, and databases.
   3. Software-as-a-service (SaaS): a method for delivering software applications over the Internet where cloud providers host and manage the software applications making it easier to have the same application on all of your devices at once by accessing it in the cloud.
3. Convert the following decimal values to unsigned binary representation.
   1. 28.3125:

Integer part:

28÷2=14 remainder 0

14÷2=7 remainder 0

7÷2=3 remainder 1

3÷2=1 remainder 1

1÷2=0 remainder 1 stop

Fraction part:

0.3125x2=0.625 integer 0

0.625x2=1.25 integer 1

0.25x2=0.5 integer 0

0.5x2=1.0 integer 1

0.0x2=0.0 integer 0 stop

Therefore (28.3125)10= (11100.0101)2

* 1. 127.0

Integer part:

127÷2=63 remainder 1

63÷2=31 remainder 1

31÷2=15 remainder 1

15÷2=7 remainder 1

7÷2=3 remainder 1

3÷2=1 remainder 1

1÷2=0 remainder 1

Therefore (127.0)10= (1111111)2

* 1. 128.0

Integer part:

128÷2=64 remainder 0

64÷2=32 remainder 0

32÷2=16 remainder 0

16÷2=8 remainder 0

8÷2=4 remainder 0

4÷2=2 remainder 0

2÷2=1 remainder 0

1÷2=0 remainder 1

Therefore (128)10= (10000000)2

* 1. 132.5625

Integer part:

132÷2=66 remainder 0

66÷2=33 remainder 0

33÷2=16 remainder 1

16÷2=8 remainder 0

8÷2=4 remainder 0

4÷2=2 remainder 0

2÷2=1 remainder 0

1÷2=0 remainder 1 stop

Fraction part:

0.5625x2=1.125 integer 1

0.125x2=0.25 integer 0

0.25x2=0.5 integer 0

0.5x2=1.0 integer 1

0.0x2=0.0 integer 0 stop

Therefore (132.5625)10= (10000100.1001)2

* 1. 13.2

Integer part:

13÷2=6 remainder 1

6÷2=3 remainder 0

3÷2=1 remainder 1

1÷2=0 remainder 1 stop

Fraction part:

0.2x2=0.4 integer 0

0.4x2=0.8 integer 0

0.8x2=1.6 integer 1

0.6x2=1.2 integer 1

0.2x2=0.4 integer 0

……

Therefore (13.2)10= (1101.00110011….)2

1. Convert the following unsigned binary numbers to decimal representation.
   1. 110.110 =1x22 +1x21 +0x20 +1x2-1 +1x2-2=4+2+0+0.5+0.25=6.75
   2. 1.101=1x20 +1x2-1 +0x2-2 +1x2-3=1+0.5+0+0.125=1.625
   3. 10111.0111=1x24 +0x23 +1x22 +1x21 +1x20+0x2-1 +1x2-2 +1x2-3 +1x2-4

=16+0+4+2+1+0+0.25+0.125+0.0625=23.4375

* 1. 11111111=27 +26 +25 +24 +23 +22 +21 +20 =128+64+32+16+8+4+2+1=255
  2. 1110.01=1x23 +1x22 +1x21 +0x20 +0x2-1 +1x2-2 =8+4+2+0+0+0.25=14.25

1. Suppose a computer uses radix-3 (ternary) logic instead of radix-2 (binary) logic to represent unsigned integers.
   1. a. What is the radix-3 representation of the decimal value 37?

37÷3=12 remainder 1

12÷3=4 remainder 0

4÷3=1 remainder 1

1÷3=0 remainder 1 stop

Therefore (37)10 = (1101)3

* 1. What is the largest value that can be represented by 6 digits?

Since ternary takes values 0, 1, and 2

Therefore, the largest 6-digit number will be (222222)3

(222222)3=2x35 +2x34 +2x33 +2x32 +2x31 +2x30=486+162+54+18+6+2 = (728)10

* 1. Why do you think that binary logic is much more commonly used than ternary logic? Be brief

I think that’s because all modern computers operate using "switches" that are either in an "on" state or an "off" state depending on whether electricity is running through them and are represented by 1 and 0 respectively in the binary system. That is, electric signals can have only two states: on (1) when the current flows and off (0) when it doesn't, which doesn’t make sense to have ternary logic used for no reason which also adds more expenses to build those ternary logic systems and increases error rates.

1. Convert the hexadecimal number FEED A BEE to binary

F: 1111, E: 1110, E: 1110, D: 1101, A: 1010, B: 1011, E: 1110, E: 1110

Therefore (FEEDABEE)16 = (11111110111011011010101111101110)2

1. Represent each of the following decimal numbers in binary using 8-bit signed magnitude, one’s complement, two’s complement, and excess-127 representations:
   1. 35
      1. 8-bit signed magnitude:

35÷2=17 remainder 1

17÷2=8 remainder 1

8÷2=4 remainder 0

4÷2=2 remainder 0

2÷2=1 remainder 0

1÷2=0 remainder 1 stop

Therefore (35)10 = (00100011)2

* + 1. one’s complement:
    2. two’s complement:

Since it is a positive number, this is the same as the signed magnitude

representation = (00100011)2

* + 1. excess-127:
  1. -35
     1. 8-bit signed magnitude:

Same as previous question (8.a) but replace the first bit to be 1 to represent the -ve sign instead of 0 (+ve), so the answer will be (10100011)2

* + 1. one’s complement:
    2. two’s complement:

bit inversion of 00100011 = 11011100

11011100

+00000001

=11011101 (no carry)

Therefore = (11011101)2

* + 1. excess-127:
  1. 97
     1. 8-bit signed magnitude:

97÷2=48 remainder 1

48÷2=24 remainder 0

24÷2=12 remainder 0

12÷2=6 remainder 0

6÷2=3 remainder 0

3÷2=1 remainder 1

1÷2=0 remainder 1 stop

Therefore (97)10 = (01100001)2

* + 1. one’s complement:
    2. two’s complement:

Since it is a positive number, this is the same as the signed magnitude

representation = (01100001)2

* + 1. excess-127:
  1. -97
     1. 8-bit signed magnitude:

Same as previous question (8.c) but replace the first bit to be 1 to represent the -ve sign instead of 0 (+ve), so the answer will be (11100001)2

* + 1. one’s complement:
    2. two’s complement:

bit inversion of 01100001 = 10011110

10011110

+00000001

=10011111 (no carry)

Therefore = (10011111)2

* + 1. excess-127:

1. 9
2. 10