Assignment #1

Jordan Hubscher

ID: 7019696

Section: PC

1.

a) 623/3 = 207 R: 2 answer: **62310 = 2120023**

207/3 = 69 R: 0

69/3 = 23 R: 0

23/3 = 7 R: 2

7/3 = 2 R: 1

2/3 => R: 2

b) 2341/7 = 334 R: 3 answer: **234110 = 65537**

334/7 = 41 R: 5

47/3 = 6 R: 5

6/7 => R: 6

2.

a) 43/2 = 21 R: 1 0.856\*2 = 1.712 => 1 answer: **43.85610 = 101011.1101102**

21/2 = 10 R: 1 0.712\*2 = 1.424 => 1

10/2 = 5 R: 0 0.424\*2 = 0.848 => 0

5/2 = 2 R: 1 0.848\*2 = 1.696 => 1

2/2 = 1 R: 0 0.696\*2 = 1.392 => 1

1/2 => R: 1 0.392\*2 = 0.784 => 0

b) 299/2 = 149 R: 1 0.5656\*2 = 1.1312 => 1 answer: **299.565610 = 100101011.1001002**

149/2 = 74 R: 1 0.1312\*2 = 0.2624 => 0

74/2 = 37 R: 0 0.2624\*2 = 0.5248 => 0

37/2 = 18 R: 1 0.5248\*2 = 1.0496 => 1

18/2 = 9 R: 0 0.0496\*2 = 0.0992 => 0

9/2 = 4 R: 1 0.0992\*2 = 0.1984 => 0

4/2 = 2 R: 0

2/2 = 1 R: 0

1/2 => R: 1

3. 01101110

a) unsigned representation:

(0\*27) + (1\*26) + (1\*25) + (1\*23) + (1\*22) + (1\*21) =

* 64 + 32 + 8 + 4 + 2 = 110
* answer: **11010**

b) two’s complement representation:

-undo two’s complement(if number is indeed negative, else just use previous answer);

* 10010001 + 1
* 10010010
* (1\*27) + (1\*24) (1\*21)
* 128 + 16 + 2 = 146
* answer: **14610**

4.

a) 10102\*102 => answer: **101002**

^shift left once

b) 110112\*1112 =

* 11011002 + 1101102 + 110112
* ^shift left twice ^shift left once ^no shift
* Add all components up and you get =
* answer: **101111012**

5.

a) 67/2 = 33 R: 1 0.25\*2 = 0.5 => 0 1000011.01\*20

33/2 = 16 R: 1 0.5\*2 = 1.0 => 1 0.100001101\*27+16

16/2 = 8 R: 0 0.100001101\*223

8/2 = 4 R: 0 2310 = 101112

4/2 = 2 R: 0 answer: **0|10111|10000110**

2/2 = 1 R: 0

1/2 => R: 1

47/2 = 23 R: 1 0.98\*2 = 1.96 => 1 101111.11\*20 => 0.10111111\*26

23/2 = 11 R: 1 0.96\*2 = 1.92 => 1 0.10111111\*26+16

11/2 = 5 R: 1 0.10111111\*222

5/2 = 2 R: 1 2210 = 101102

2/2 = 1 R: 0 answer: **0|10110|10111111**

1/2 => R: 1

b) 0|1000011.01 0.111001100\*223

+ 0|0101111.11 answer: **0|10111|11101110**

0|1110111.00

6.

a) 34/2 = 17 R: 0 0.55\*2 = 1.10 => 1 34.5510 = 100010.1000110011001100112

17/2 = 8 R: 1 0.10\*2 = 0.20 => 0 => 100010.100011001100110011\*20 =

8/2 = 4 R: 0 0.20\*2 = 0.40 => 0 => 1.00010100011001100110011\*25+127 =

4/2 = 2 R: 0 0.40\*2 = 0.80 => 0 => 1.00010100011001100110011\*2132

2/2 = 1 R: 0 0.80\*2 = 1.60 => 1 => 13310 = 100001102

1/2 => R: 1 0.60\*2 = 1.20 => 1 => answer: **0|10000100|00010100011001100110011**

=> repeating decimal => 0.20\*2 = 0.40 => 0

0.40\*2 = 0.80 => 0

0.80\*2 = 1.60 => 1

0.60\*2 = 1.20 => 1

0.20\*2 = 0.40 => 0

0.40\*2 = 0.80 => 0

0.80\*2 = 1.60 => 1

0.60\*2 = 1.20 => 1

0.20\*2 = 0.40 => 0

0.40\*2 = 0.80 => 0

0.80\*2 = 1.60 => 1

0.60\*2 = 1.20 => 1

b) 201/2 = 100 R: 1 0.601\*2 = 1.202 => 1 201.60110 = 11001001.1001100111011012

100/2 = 50 R: 0 0.202\*2 = 0.404 => 0 ^negative will be displayed in sign bit later on

50/2 = 25 R: 0 0.404\*2 = 0.808 => 0 => 11001001.1001100111011011\*20 =

25/2 = 12 R: 1 0.808\*2 = 1.616 => 1 => 1.10010011001100111011011\*27+127 =

12/2 = 6 R: 0 0.616\*2 = 1.232 => 1 => 1.10010011001100111011011\*2134

6/2 = 3 R: 0 0.232\*2 = 0.464 => 0 => 13410 = 100001102

3/2 = 1 R: 1 0.464\*2 = 0.928 => 0 => answer:

1/2 => R: 1 0.928\*2 = 1.856 => 1 => **1|10000111|10010011001100111011011**

0.856\*2 = 1.712 => 1

0.712\*2 = 1.424 => 1

0.424\*2 = 0.848 => 0

0.848\*2 = 1.696 => 1

0.696\*2 = 1.392 => 1

0.392\*2 = 0.784 => 0

0.784\*2 = 1.568 => 1

0.568\*2= 1.136 => 1

7. 1. 1110001110010111 1. & 2. : D(min) = 8 answer : **D(min) = 5**

2. 1001011010001110 1. & 3. : D(min) = 9

3. 0010111101101111 1. & 4. : D(min) = 5

4. 1100000000011111 2. & 3. : D(min) = 9

2. & 4. : D(min) = 7

3. & 4. : D(min) = 10

8.

01111010101

m = 7, r = 4

(m + r +1) ≤ 2r

(7 + r +1) ≤ 2r

r => 4 since (7 + 4 + 1 = 12) ≤ (24 = 16)

0 1 1 1 1 0 1 0 1 0 1

- - - - - - - - - - -

11 10 9 **8** 7 6 5 **4** 3 **2****1** <= the bold numbers are parity bits; 1, 2, 4, and 8.

1 = 1 5 = 1 + 4 9 = 1 + 8

2 = 2 6 = 2 + 4 10 = 2 + 8

3 = 1 + 2 7 = 1 + 2 + 4 11 = 1 + 2 + 8

4 = 4 8 = 8 12 = 4 + 8

Bit 1 checks 1, 3, 5, 7, 9, and 11: With even parity, this produces an error.

Bit 2 checks 2, 3, 6, 7, 10, and 11: With even parity, this produces an error.

Bit 4 checks 4, 5, 6, and 7: This is ok.

Bit 8 checks 8, 9, 10, and 11: With even parity, this produces an error.

Parity bits 1, 2, and 8 show error. They all check 11 (1 + 2 + 8 = 11).

answer: **11111010101**

9. 101

a) 1011) 100010 answer: **quotient = 1012**

1011 answer: **remainder = 1012**

001110

1011

0101

b) 11101

100) 1110111 answer: **quotient = 111012**

100 answer: **remainder = 112**

0110

100

0101

100

00111

100

011

10.

11100010

1101) 10001111000 quotient = 111000102

1101 remainder = 102

01011 answer: 100011110002 + 102 =

1101 => **100011110102**

01101

1101

00001100

1101

00010