COMP102 Fall 2006
Assignment 1 Solutions

Note: No grade was given for providing the final answer only.

Q1: Conversion binary to decimal and vice versa [14]

Convert the following decimal numbers to binary:

[2] (a) 10

10/2 = 5 + 0/2
5/2 = 2 + 1/2
2/2 = 1 + 0/2
1/2 = 0 + 1/2 Answer: 1010_2

[2] (b) 2

2/2 = 1 + 0/2
1/2 = 0 + 1/2 Answer: 10_2

[2] (c) 64

64/2 = 32 + 0/2
32/2 = 16 + 0/2
16/2 = 8 + 0/2
8/2 = 4 + 0/2
4/2 = 2 + 0/2
2/2 = 1 + 0/2
1/2 = 0 + ½ Answer: 1000000_2

Convert the following binary numbers to decimal:

[2] (d) 100101

$100101_2 = 1x(2_{10})^5 + 0x(2_{10})^4 + 0x(2_{10})^3 + 1x(2_{10})^2 + 0x(2_{10})^1 + 0x(2_{10})^0$
$= 32_{10} + 0_{10} + 0_{10} + 4_{10} + 0_{10} + 1_{10}$
$= 37_{10}$

Answer: 37_{10}

[2] (e) 0110

$0110_2 = 0x(2_{10})^3 + 1x(2_{10})^2 + 1x(2_{10})^1 + 0x(2_{10})0$
$= 0_{10} + 4_{10} + 2_{10} + 0_{10}$
$= 6_{10}$

Answer: 6_{10}

[4] (f) 1111111111111111111

$1111111111111111111_2 = (2_{10})^{19} - 1_{10}$

$= 524287_{10}$

Answer: 524287_{10}

Q2: Binary arithmetic [16]

Perform the following computations on binary numbers, without converting them to decimal numbers. It may help to think of the steps taken to solve this problem when the numbers are represented in decimal.

[4] (a) 1101011 + 111010

		1	1	1		1		
		1	1	0	1	0	1	1
+			1	1	1	0	1	0
	1	0	1	0	0	1	0	1

Answer: 10100101_2

[4] (b) 100001 – 1110

	1	1	1	1		
	1	0	0	0	0	1
-			1	1	1	0
		1	0	0	1	1

Answer: 10011_2

[4] (c) 11001 / 10

```
      1 1 0 0 . 1
10 /  1 1 0 0 1
    - 1 1 0 0 0
      0 0 0 0 1 0
    -         1 0
                0
```

Answer: 1100.1_2

(Note: The effect is shifting the decimal point to the left, same when dividing by 10 in decimal)

IF THIS IS WHAT YOU DID AND YOU LOST MARKS FOR THIS QUESTION, COME SEE THE INSTRUCTOR.

[4] (d) 11001 * 100

			1	1	0	0	1
x					1	0	0
			0	0	0	0	0
		0	0	0	0	0	
+	1	1	0	0	1		
	1	1	0	0	1	0	0

Answer: 1100100_2
(Note: The effect is similar to that of decimal multiplication by 100)

IF THIS IS WHAT YOU DID AND YOU LOST MARKS FOR THIS QUESTION, COME SEE THE INSTRUCTOR.

Q3: Numbers of Babylon [10]
We learnt that the Babylonians used base 60 to represent numbers. Suggest some advantages of this number representation or discuss potential origins.
Be creative!
Note: We didn't want people to research this on the Internet and quote sources. However, no marks were taken off if you did. Any convincing answer that discussed either the advantages (at least 2) or potential origins got full marks.

A lot of creative answers… Here are a few:
Advantages:

- Makes some mathematical calculations easier,
    - Circle measurements & calculations (360^0)
    - time (60 minutes in an hour, 60 seconds in a minute)
    - Geographic coordinates
    - 60 has many divisors: 2,3,4,5,6,15,20,30
- Only uses 2 symbols to represent numbers (although no one mentioned how these symbols were be used)
- Larger numbers take less space to store
- Both 6 and 60 are perfect numbers (a sum of their factors)

Potential Origins:

- •It is the number of joints on fingers
- •Based on the number of days in a month (360 days = 6 months = 1 year)
- •Because it is the smallest number divisible by 2, 3, 4, & 5.
- •Because they divided weights into thirds
- •The equilateral triangle was the base of their construction (made of three 60^0 angles)

Q4: Paper tape [15]

In the paper tape featured on the right, what is the largest decimal number that can be represented in a data row? [5]

The largest decimal is the largest binary that can be represented by 8 holes:

$11111111_2 = 2_{10}^{8} – 1_{10} = 255_{10}$

Answer: 255_{10}

What is the smallest? [5]

The smallest decimal is the smallest binary that can be represented by 8 holes:

$00000000_2 = 0_{10}$

Answer: 0_{10}

What do you think is the significance of the sprocket holes (vertical column of holes very close together)? [5]

Answer:

1. The off-center position of the sprocket holes separates the bits into two sets and allow the operator to differentiate between the two sides of the tape (front & back)
2. The sprocket holes were used to *grab* and move the tape by the machine.

Q5: Tally sticks [15]

Suppose you are using tally sticks to count days. To do this, you cut out a gash in the stick for each additional day that passes.

What is the base of the number system that you are using? [5]

Answer: Base = 1

(No grade was given for any other answers)

Write “9” using this number system. [2]

Answer: 111111111

(No grade was given for any other answers)

How does this number system compare to the other number systems we have seen in class in terms of number of symbols required to represent numbers? [4]

Answer:The number of symbols (gashes) required to represent numbers in this system is a lot smaller than the number of symbols used in binary, octal, decimal or hexadecimal.

Is there any number that cannot be represented using this number system? Why or why not? [4]
Answer :Zero cannot be represented by this number system, because there is only one symbol (1) and it cannot be used to represent zero.
(Note: Neither can you represent negative numbers, fractions, complex numbers, etc.)

Q6: Text Representation [15]
The following data is represented as a text file.
[5] (a) How large is the file (in bytes)?

Answer: bits = 88bits x 1byte / 8 bits = 11 bytes
(To get points you have to show that the file is 88 bits = 88bits x 1byte / 8 bits = 11 bytes)

[10] (b) What are the contents of the file?
0100011101101111001000000100100001000001010000100101001100100000010001110110111100 100001
Answer: Go HABS Go!

01000111_2 = 71_{10} = G

01101111_2 = 111_{10} = o

00100000_2 = 32_{10} = SPACE

01001000_2 = 72_{10} = H

01000001_2 = 65_{10} = A

01000010_2 = 66_{10} = B

01010011_2 = 83_{10} = S

00100000_2 = 32_{10} = SPACE

01000111_2 = 71_{10} = G

01101111_2 = 111_{10} = o

00100001_2 = 33_{10} = !

(To get points you have to show that each letter is 8 bits of the file. Or at least mention that you've used ASCII)

Q7: Image encoding [15]
Suppose you were given a binary image file of a triangle, where 0 means that the pixel is white and 1 means that the pixel is black, like so:

0000000000000000000
0000000001000000000
0000001111111000000

0001111111111111000
1111111111111111111
0000000000000000000

You need to create a more compact file format. Can you suggest a way to encode this image so as to minimize the number of bits used, but without losing any of the information? [10] What is the number of bits that you save using your strategy? [5]

There are many possibilities for this question. To get full marks, you need to show a representation that reduces the number of bits of the final binary representation. If you gave a good representation not in binary you got 10-13 points. If your representation is not clear or does not work, you get 5. The following are two good answers (the 3rd is based on answers given by students in the class):

Answer 1:

For each line we can specify how many 0's and 1's are used:

19 (zeros)
9 (zeros) 1 (ones) 9 (zeros)
6 (zeros) 7 (ones) 6 (zeros)
3 (zeros) 13 (ones) 3 (zeros)
19 (ones)
19 (zeros)

Then we use 5 bits to encode each decimal number and we add a bit after each number to indicate if it's a 1 or 0:

10011 0
01001 0 00001 1 01001 0
00110 0 00111 1 00110 0
00011 0 01101 1 00011 0
10011 1
10011 0

So compacted format will look like:
100110
010010000011010010
001100001111001100
000110011011000110
100111
100110

new representation = 72 bits

old representation = 114 bits

Saved = 114 – 72 = 42 bits.

<u>Answer 2</u>:
Same as question 1 but this time assume that the computer knows that the picture is 19x6 bits, we just have to tell it the sequence of 0's and 1's. So:

28 (zeros)11100 0
1 (ones) 00001 1
15 (zeros)01111 0
7 (ones) 00111 1
9 (zeros) 01001 0
13 (ones) 01101 1
3 (zeros) 00011 0
19 (ones) 10011 1
19 (zeros)10011 0

So compacted format will look like:

111000 000011 011110 001111 010010 011011 000110 100111 100110

new representation = 54 bits
old representation = 114 bits

Saved = 114 – 54 = 60 bits.

<u>Answer 3</u>:
Consider the following 2-bit codes:

	0		0		0		0
	0		0		0		1
	0		0		1		1
00->	0	01->	1	10->	1	11->	1
	1		1		1		1
	0		0		0		0

Then, the compressed file would be:

00000010101101010111010100101010000000
If we save both the 2-bit number encoding and the image encoding, then the file becomes:

000010
000110
001110
011110
00000001010110101011101010010101000000

new representation = 62 bits
old representation = 114 bits
Saved = 114 – 62 = 52 bits.