Proof of Study 2

Qianlang Chen

U1172983

Problem 1

The decimal number that I am dealing with is **1172.1**.

Part 1 - Convert 1172 into hex by using the unsigned binary notation:

The following table shows how the integer 1172 can be converted into binary:

	1172	even	**0**
1172/2	586	even	**0**
586/2	293	odd	**1**
293/2	146	even	**0**
146/2	73	odd	**1**
73/2	36	even	**0**
36/2	18	even	**0**
18/2	9	odd	**1**
9/2	4	even	**0**
4/2	2	even	**0**
2/2	1	odd	**1**

Therefore,

$$\begin{aligned}
1172_{dec} &= (100\ 1001\ 0100)_{bin} \\
&= (100 * 1\ 0000\ 0000 + 1001 * 1\ 0000 + 100 * 1)_{bin} \\
&= (4 * 100 + 9 * 10 + 4 * 1)_{hex} \\
&= \mathbf{0x494}
\end{aligned}$$

Part 2 - Convert -1172 into hex by using two's complement representation:

According to the previous problem and using 32 bits,

$$1172_{dec} = (0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1001\ 0100)_{bin}$$

Using two's complement, **~N = -(N + 1)**, where '~' inverses all binary digits:

$$\begin{aligned}
-1172_{dec} &= -[(1172 - 1) + 1]_{dec} = \sim(1172 - 1)_{dec} \\
&= \sim(0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1001\ 0100 - 1)_{bin} \\
&= \sim(0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1001\ 0011)_{bin} \\
&= (1111\ 1111\ 1111\ 1111\ 1111\ 1011\ 0110\ 1100)_{bin} \\
&= (F * 1000\ 0000 + F * 100\ 0000 + F * 10\ 0000 \\
&\quad + F * 1\ 0000 + F * 1000 + B * 100 + 6 * 10 + C * 1)_{hex} \\
&= \mathbf{0xFFFF\ FB6C}
\end{aligned}$$

Part 3 - Convert 1172.1 into hex as a single-precision floating-point number.

The following table shows how the fractional part (0.1_{dec}) can be converted into binary. Since binary cannot express 0.1_{dec} with a terminated fractional part, the repeated part is marked in **bold**:

0.1*2	0.2	**0**
0.2*2	0.4	**0**
0.4*2	0.8	**0**
0.8*2	1.6	**1**
0.6*2	1.2	**1**
0.2*2	**0.4**	**0**
0.4*2	**0.8**	**0**
0.8*2	**1.6**	**1**
0.6*2	**1.2**	**1**

Therefore, the number 1172.1 can be written in binary (with up to 23 places of mantissa):

$$
\begin{aligned}
1172.1_{dec} &= (1172 + 0.1)_{dec} \\
&= (10010010100 + 0.0001100110011)_{bin} \\
&= (1.0010\ 0101\ 0000\ 0110\ 0110\ 011 * 10^{1010})_{bin}
\end{aligned}
$$

Meaning that this floating-point number will have:

Sign Bit: 0 (positive)

Exponent: 1010 + 0111 1111 = $(1000\ 1001)_{bin}$

Mantissa: 001 0010 1000 0011 0011 0011

Or as a string of bits (in the order of the sign-bit, exponent, and finally mantissa) with the equivalent hex value:

0100 0100 1001 0010 1000 0011 0011 0011

= **0x4492 8333**

Problem 2

One line of Java that does the job:

```
Num = Data[11] * 100 + Data[12] * 10 + Data[13];
```

And a portion of assembly code that does the job:

```
# This piece of assembly code concats the digits at 3 different
# places in an array into an integer.
# Get the address of the Data variable and load in the three digits.
    la    $t0,  Data
    lw    $t1,  44($t0)    # 11th place * 4 bytes
    lw    $t2,  48($t0)    # 12 * 4
    lw    $t3,  52($t0)    # 13 * 4
# Now the three digits should be stored in t1, t2, and t3,
respectively.
# Perform "t1 *= 100" by splitting it up into "t1 * 10 * 10".
    add   $t4,  $t1,  $t1
    add   $t4,  $t4,  $t1
    add   $t4,  $t4,  $t1
    add   $t4,  $t4,  $t1  # t1 * 5 so far
    add   $t4,  $t4,  $t4  # Now t4 should hold "t1 * 10"
    add   $t1,  $t4,  $t4
    add   $t1,  $t1,  $t4
    add   $t1,  $t1,  $t4
    add   $t1,  $t1,  $t4  # t4 * 5 so far
    add   $t1,  $t1,  $t1  # Now t1 should hold "t4 * 10"
```

```
# Perform "t2 *= 10" using similar ideas.

      add   $t4, $t2, $t2

      add   $t4, $t4, $t2

      add   $t4, $t4, $t2

      add   $t4, $t4, $t2   # t2 * 5 so far

      add   $t2, $t4, $t4   # Now t2 should hold "t2 * 10"

# Perform "t1 += t2 + t3".

      add   $t2, $t1, $t2

      add   $t1, $t2, $t3

# Load the address of the Num variable (and override the old Data's
address)

# and store the rusult from t1.

      la    $t0, Num

      sw    $t1, 0($t0)     # without offset

# Complete.
```