Jong Park  
10/03/2018  
CS 472 – Computer Architecture  
HW1

1. a) 12-bit binary representation of -14210­

-14210 = -0b 0000 1000 1110   
 = -0b 1111 0111 0001 +1  
 = **0b 1111 0111 0010**

b) 12-bit binary: 119 + (-142)

11910 = 0b 0000 0111 0111  
-14210 = 0b 1111 0111 0010  
119+ (-142) = -23  
= **0b 1111 1110 1001**

c) Smallest number of bits that can correctly represent the signed binary of 5684110

5684110 = 0b 0 1101 1110 0001 0010   
16 bits + 1 (signed) = **17 minimum bits.**

d) Unsigned hexadecimal of 5684110

5684110 = 0b 1101 1110 0001 0010 (unsigned)  
 = **0x DE12**

1. Using the 32-bit IEEE-754 standard:
2. Binary of 572.37510:

572.37510 = 0b 0010 0011 1100 . 0110  
 = (-1)1 \* 1.0001111000110 \* 29

S = 0  
Fraction = 0 0011 1100 0110  
Exp = 9 + 127 = 136 = 0b 1000 1000 (single)  
 = 9 + 1023 = 1032 = 0b 100 0000 1000 (double)

= **0b 0 1000 1010 000 1111 0001 1000 0000 0000** (single)  
= **0b 0 100 0000 1000 0 0011 1100 0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 000** (double)

1. Most negative floating-point number in 32-bit IEEE-754 format  
   x = (-1)1 \* (1 + 223) \* 2(254-127)

= -340282346638528859811704183484516925440  
 = **-3.402823 \* 1038**

1. Binary representation  
   S=1

Fraction = 111 1111 1111 1111 1111 1111

Exp = 127 + 127 = 254 = 0b 1111 1110  
= **0b 1 1111 1110 111 1111 1111 1111 1111 1111**

1. Decimal 472.2
   1. Why can’t we use 32-bit IEEE 754 to represent 472.2 accurately?

**Same reason why we can’t represent 1/3 in decimals accurately. 0.2 has a repeating mantissa (1100) when represented in binary.**

* 1. Can we use 64-bit IEEE-754 instead?

**No, repeating mantissa (1100) will keep on repeating throughout history.**

1. Given Register values:

.text

$t1 = 0x0004

$s0 = 0x01A7

$a1 = 0x6B20

$sp = 0x8034

.code

addi $sp, $sp, -4 ; $sp = 0x8030

sw $s0, 0($sp) ; store address of stack pointer (word) to $s0   
 add $s0, $zero, $zero ; $s0 = 0x0000

add $t1, $s0, $a1 ; $t1 = 0x0000 + 0x6B20

Result:

**$t1 = 0x6B20  
 $s0 = 0x0000  
 $a1 = 0x6B20  
 $sp = 0x8030**

1. a) see prime\_number.c

b) see run.sh

c) I checked. It looks like it works.

d) More command added to run.sh

e) out of the 8 registers for x86 (eax, ebx, ecx, edx, esi, edi, ebp, esp), it only used:

**eax, edx**

f) more code added to run.sh

g) Out of the 32 registers for MIPS architecture, it only used:

**$0** (zero), **$31** (return address), **$sp** (stack pointer) and **$2** (return value)