Problem Set 2

Justin Ely

605.411 Foundations of Computer Architecture

13 September, 2016

1)

Hex	7	1	5	4	0	0	0	0
Bin	0111	0001	0101	0100	0000	0000	0000	0000

In IEEE 754 single precision, this binary value represents 1.05E30.

2)

Using the unsigned representation of 12 = 00001100:

a)

The excess-128 representation gives the signed value as 128 higher than the negative value. Thus -12 = 128-12 = 116 = 01110100 or 0x74.

b)

1's complement representation of -12 can be found simply by taking the complement of every bit: 11110011 or 0xf3.

c)

The sign and magnitude representation simply uses the first bit as the sign, and the remaining 7 bits as the magnitude: 10001100 or 0x8c.

d)

Two's complement is found by taking the complement of every bit and then adding 1. The complement being 11110011, with 1 added: -12 = 11110100 or 0xf4.

3)

Sign	Expoent	Fraction
0	10000100	11111010000000000000000

4)

Sign	Exponent	Fraction
0	10000100000	1111101000000000000000000000000000000000000000000000

5)

Iteration	Step	Multiplier	Multiplicand	Product
0	Init	01100010	0000000000010010	0000000000000000
0	1: 0: nothing	01100010	0000000000010010	0000000000000000
0	2: shift m-and left	01100010	0000000000100100	0000000000000000
0	3: shift m-er right	00110001	0000000000100100	0000000000000000
1	1: 1: add	00110001	0000000000100100	0000000000100100
1	2: shift m-and left	00110001	0000000001001000	0000000000100100
1	3: shift m-er right	00011000	0000000001001000	0000000000100100
2	1: 0: nothing	00011000	0000000001001000	0000000000100100
2	2: shift m-and left	00011000	0000000010010000	0000000000100100
2	3: shift m-er right	00001100	0000000010010000	0000000000100100
3	1: 0: nothing	00001100	0000000010010000	0000000000100100
3	2: shift m-and left	00001100	0000000100100000	0000000000100100
3	3: shift m-er right	00000110	0000000100100000	0000000000100100
4	1: 0: nothing	00000110	0000000100100000	0000000000100100
4	2: shift m-and left	00000110	0000001001000000	0000000000100100
4	3: shift m-er right	00000011	0000001001000000	0000000000100100
5	1: 1: add	00000011	0000001001000000	0000001001100100
5	2: shift m-and left	00000011	0000010010000000	0000001001100100
5	3: shift m-er right	00000001	0000010010000000	0000001001100100
6	1: 1: add	00000001	0000010010000000	0000011011100100
6	2: shift m-and left	00000001	0000100100000000	0000011011100100
6	3: shift m-er right	00000000	0000100100000000	0000011011100100
7	1: 0: nothing	00000000	0000100100000000	0000011011100100
7	2: shift m-and left	00000000	0001001000000000	0000011011100100
7	3: shift m-er right	00000000	0001001000000000	0000011011100100
8	1: 0: nothing	00000000	0001001000000000	0000011011100100
8	2: shift m-and left	00000000	0010010000000000	0000011011100100
8	3: shift m-er right	00000000	0010010000000000	0000011011100100

The value left in the Product register evaluates to 1764, which is equal to the product of 0x62 (98) and 0x12 (18).

6)

op code	source register	second source	dest. register	shift amount	function
000000	01000	01001	10001	00000	100000
0	8	9	17	0	32
R-type	\$t0	\$t1	\$s1	0	add

In HEX, this translates to:

Bin	0000	0001	0000	1001	1000	1000	0010	0000
Hex	0	1	0	9	8	8	2	0

7)

Hex	1	2	0	F	0	0	0	8
Bin	0001	0010	0000	1111	0000	0000	0000	1000

The first 6 bits signify an opcode of 4, which makes this an I-type.

opcode	rs	rt	immediate
000100	10000	01111	0000000000001000
beq	\$s0	\$t7	8

This translates into a MIPS command of ”beq \$t7, \$s0, 8”.