CSE 2421
Spring, 2020
Homework 2

Due Date: Friday, March 6, 11:30PM. Only submissions via Carmen will be accepted without prior approval. To receive credit, solutions must be clearly written or typed and appear in the same order as the questions below.

1. For the six 16 bit values shown below, show what the 32 bit sign extended value would be and what the value truncated to 8 bits would be. Then, for each truncated bit pattern, say whether the value of the original 16 bit representation is preserved or not, and explain how you determined your answer.

a. 0000 0000 1001 0111 (signed, that is, interpreted as B2T)

i. 32 bit sign extended value: 0000 0000 0000 0000 0000 0000 1001 0111
ii. 8 bit truncated value: 1001 0111
iii. Does the value change when truncated to 8 bits? Yes.
iv. How did you decide? The sign bit becomes to 1, which means when we truncated to 8 bits, the value becomes negative.

b. 1111 1111 0110 1010 (unsigned, that is, interpreted as B2U)

i. 32 bit sign extended value: 1111 1111 1111 1111 1111 1111 0110 1010
ii. 8 bit truncated value: 0110 1010
iii. Does the value change when truncated to 8 bits? Yes.
iv. How did you decide? It is unsigned, every bit count for all 16 bits original binary representation, after we truncated, we are losing values.

c. 1111 1111 0011 1010 (signed, that is, interpreted as B2T)

i. 32 bit sign extended value: 1111 1111 1111 1111 1111 1111 0011 1010
ii. 8 bit truncated value: 0011 1010
iii. Does the value change when truncated to 8 bits? Yes.
iv. How did you decide? The sign bit was truncated out after we truncated it to 8 bits, which means when we truncated to 8 bits, the value becomes positive.

d. 0000 0000 1100 0011 (unsigned, that is, interpreted as B2U)

i. 32 bit sign extended value: 0000 0000 0000 0000 0000 0000 1100 0011
ii. 8 bit truncated value: 1100 0011
iii. Does the value change when truncated to 8 bits? No.
iv. How did you decide? First 16 bits are 0, therefore, there is no change when the binary represent as unsigned int (since unsigned int does not have a bit represent sign).

e. 1101 1111 0011 1010 (signed, that is, interpreted as B2T)

i. 32 bit sign extended value: 1111 1111 1111 1111 1101 1111 0011 1010
ii. 8 bit truncated value: 0011 1010
iii. Does the value change when truncated to 8 bits? Yes.
iv. How did you decide? The sign changed after we truncated to 8 bits.

f. 0010 0000 1100 0011 (unsigned, that is, interpreted as B2U)

i. 32 bit sign extended value: 0000 0000 0000 0000 0010 0000 1100 0011
ii. 8 bit truncated value: 1100 0011
iii. Does the value change when truncated to 8 bits? Yes.
iv. How did you decide? We lost a bit after we truncated to 8 bits, therefore, the value changed.

2. Without converting the numbers below to decimal, solve the following arithmetic problems, giving your answers in hexadecimal format. Don't forget that hexadecimal numbers have a "0x" prefix; Decimal numbers do not.

   a. 0x51C3
   +0x0009
   0x51CC

   b. 0x50C3
   -0x0041
   0x5082

   c. 0x50C3
   + 65
   0x5104

   d. 0x51EA
   -0x50C3
   0x0127

   e. 0x7000
   -0x0001
   0x6FFF

   f. 0x6F10
   +0x000A
   0x6F1A

   g. 0x5CF0
   +0x001F
   0x5D0F

   h. 0x5000
   -0x0008
   0x4FF8

3. For addition of the following 16-bit pair of signed binary values, show the values of carry in, sum, and carry out for each pair of bits in the operands. Assuming that the result must fit in 16 bits, indicate whether there is overflow for this sum. Explain briefly how the hardware (the CPU) can determine whether or not there is overflow.

Carry In	1100 1111 0000 0000
	1110 0011 1001 1001
	0111 0101 1010 0110
Sum	0101 1001 0011 1111
Carry Out	1110 0111 1000 0000

Overflow not **occurs** in this case, since the most significate carry in and carry out bit is 1.

4. For addition of the following 16-bit pair of **unsigned** binary values, show the values of carry in, sum, and carry out for each pair of bits in the operands. Assuming that the result must fit in 16 bits, indicate whether there is overflow for this sum. Explain briefly how *the hardware (the CPU) can determine* whether or not there is overflow.

Carry In	0000 1111 0111 1110
	1010 0011 1001 1001
	1001 0101 1010 0111
Sum	0011 1001 0100 0000
Carry Out	1000 0111 1011 1111

Overflow **occurs** in this case, since the carry out from the most significate carry out bit is 1 which is different than carry in bit is 0.

5. Given the following binary value: 1000 1100 0100 0110, convert it to a decimal value for each of the specified integer formats. Show/explain your work.

a. Integer unsigned (B2U)

i. $1 * 2^{15} + 0 * 2^{14} + 0 * 2^{13} + 0 * 2^{12} + 1 * 2^{11} + 1 * 2^{10} + 0 * 2^{9} + 0 * 2^{8} + 0 * 2^{7} + 1 * 2^{6} + 0 * 2^{5} + 0 * 2^{4} + 0 * 2^{3} + 1 * 2^{2} + 1 * 2^{1} + 0 * 2^{0} = 35910$

b. Integer signed (B2T)

i. 1000 1100 0100 0110 (negative, since first bit is 1)

ii. 0111 0011 1011 1001 (flip)

iii. $2^{14} + 2^{13} + 2^{12} + 2^{9} + 2^{8} + 2^{7} + 2^{5} + 2^{4} + 2^{3} + 2^{0} = 29625$

iv. $29625 + 1$=29626

v. -29626

c. Integer signed (B2O)

i. 1000 1100 0100 0110 (negative, since first bit is 1)

ii. 0111 0011 1011 1001 (flip)

iii. $2^{14} + 2^{13} + 2^{12} + 2^{9} + 2^{8} + 2^{7} + 2^{5} + 2^{4} + 2^{3} + 2^{0}$=29625

iv. -29625

d. Integer signed (B2S)

i. 1000 1100 0100 0110 (negative, since first bit is 1)

ii. 000 1100 0100 0110 (remove the sign bit)

iii. $2^{11} + 2^{10} + 2^{6} + 2^{2} + 2^{1} = 3142$

iv. -3142

e. ASCII (what ASCII values are represented?)

i. ŒF

6. Given the decimal value 567, convert it to the equivalent hexadecimal value for each of the specified formats. If the conversion cannot be done, explain why not. Show/explain your work.

a. Integer unsigned (B2U)

i. 567/16=35…7

ii. 35/16=2…3

iii. 2/16=0…2

iv. 0x237

b. Integer signed (B2T)

i. 567/16=35…7

ii. 35/16=2…3
iii. 2/16=0…2
iv. 0x237

c. Integer signed (B2O)
i. 567/16=35…7
ii. 35/16=2…3
iii. 2/16=0…2
iv. 0x237

d. Integer signed (B2S)
i. 567/16=35…7
ii. 35/16=2…3
iii. 2/16=0…2
iv. 0x237

7. Given the decimal value -2459, convert it to the equivalent hexadecimal value for each of the specified formats. If the conversion cannot be done, explain why not. Show/explain your work.

a. Integer unsigned (B2U)
i. Unable to convert a negative value

b. Integer signed (B2T)
i. It is a negative value
ii. 2459/2=1229…1
iii. 1229/2=614…1
iv. 614/2=307…0
v. 307/2=153…1
vi. 153/2=76…1
vii. 76/2=38…0
viii. 38/2=19…0
ix. 19/2=9…1
x. 9/2=4…1
xi. 4/2=2…0
xii. 2/2=1…0
xiii. 1/2=0…1
xiv. 1001 1001 1011
xv. 0000 1001 1001 1011 (Extend to 16 bits)
xvi. 1111 0110 0110 0100 (Flip)
xvii. 1111 0110 0110 0101 (Negative value, add 1)
xviii. F 6 6 5
xix. 0xF665

c. Integer signed (B2O)
i. It is a negative value
ii. 2459/2=1229…1
iii. 1229/2=614…1
iv. 614/2=307…0
v. 307/2=153…1

vi. 153/2=76…1
vii. 76/2=38…0
viii. 38/2=19…0
ix. 19/2=9…1
x. 9/2=4…1
xi. 4/2=2…0
xii. 2/2=1…0
xiii. 1/2=0…1
xiv. 1001 1001 1011
xv. 0000 1001 1001 1011 (Extend to 16 bits)
xvi. 1111 0110 0110 0100 (Flip)
xvii. F 6 6 4
xviii. 0xF664

d. Integer signed (B2S)

i. It is a negative value
ii. 2459/2=1229…1
iii. 1229/2=614…1
iv. 614/2=307…0
v. 307/2=153…1
vi. 153/2=76…1
vii. 76/2=38…0
viii. 38/2=19…0
ix. 19/2=9…1
x. 9/2=4…1
xi. 4/2=2…0
xii. 2/2=1…0
xiii. 1/2=0…1
xiv. 1001 1001 1011
xv. 0000 1001 1001 1011 (Extend to 16 bits)
xvi. 1000 1001 1001 1011 (Change the first bit to 1)
xvii. 8 9 9 B
xviii. 0x899B

8. Assume that each of the following five hexadecimal values below is stored as a 32-bit (4-byte) integer value and that these 5 values are being **stored sequentially in memory starting at address 0x3A60**. Show the hex representation of the bytes in memory for both the big endian and little endian byte addressing schemes. Be sure to extend the "**Address**" column specifying the correct address values for each byte of memory needed to store these 5 values. HINT: Assume that integers are stored in 32 bits; 32 bits is 4 bytes, so 5 values at 4 bytes each would be a total of 20 bytes of memory.

a. 0x**00**EC31C2
b. 0x5576DBAE
c. 0x**000**B5219
d. 0x**000000**AA
e. 0x**0**4A2E191

The ***beginning*** of your table might look like this:

Address	Big Endian	Little Endian
0x3A60	00	C2
0x3A61	EC	31
0x3A62	31	EC
0x3A63	C2	00
0x3A64	55	AE
0x3A65	76	DB
0x3A66	DB	76
0x3A67	AE	55
0x3A68	00	19
0x3A69	0B	52
0x3A6A	52	0B
0x3A6B	19	00
0x3A6C	00	AA
0x3A6D	00	00
0x3A6E	00	00
0x3A6F	AA	00
0x3A70	04	91
0x3A71	A2	E1
0x3A72	E1	A2
0x3A73	91	04

9. Assuming that 12 bits are available to store the relevant operand and the result (Show all of your work for full credit):

a. Show how the compiler can use **subtraction* and shifting *only*** (that is, no use of multiplication) to determine the result of 15*60, if 15 is the value of a variable, x, and **60 is a constant** which is known to the compiler at compilation time. Show the bit patterns of both operands in the multiplication, and explain clearly the steps you follow to determine the result. Hint: See the slides on "multiplying by constants".

i. $60 = 2^6 - 2^2$
ii. $x = 15 = 0000\ 0000\ 1111$
iii. $x * 60 = x * 2^6 - x * 2^2 = (x \ll 6) - (x \ll 2)$
iv. 0011 1100 0000
v. -0000 0011 1100
vi. ----------
vii. 0011 1000 0100
viii. $0011\ 1000\ 0100 = 900$
ix. $15 * 60 = 900$

b. Show how the compiler can use **subtraction* and shifting *only*** (that is, no use of multiplication) to determine the result of 34*-56, if 34 is the value of a variable, x, and **-56 is a constant** which is known to the compiler at compilation time. Show the bit patterns of both operands in the multiplication and explain clearly the steps you follow to determine the result. Hint: See the slides on "multiplying by constants" when the constant is ***negative***.

i. $-56 = 2^3 - 2^6$
ii. $x = 34 = 0000\ 0010\ 0010$
iii. $x * (-56) = x * 2^3 - x * 2^6 = (x \ll 3) - (x \ll 6)$
iv. 0001 0001 0000
v. -1000 1000 0000
vi. ----------
vii. 1000 1001 0000
viii. $1000\ 1001\ 0000 = -1904$
ix. $34 * (-56) = -1904$

*Using negation and addition rather than subtraction is an acceptable alternative.

10. Given x = -39 (i.e. a signed integer encoded in B2T 1101 1001), represented in 8 bits, and $y = 2^k$,

determine the values as designated by each column heading (see the signed integer division by 2^k slides).

k	y	Binary of x right shifted by k	Decimal (x>>k)	x/y (decimal)
0	1	1101 1001	217	217
1	2	0110 1100	108	108.5
2	4	0011 0110	54	54.25
3	8	0001 1011	27	27.125
4	16	0000 1101	13	13.5625
5	32	0000 0110	6	6.78125

11. Based on the decimal value given in the first column, determine the result for each of the different rounding options specified when rounding to the nearest whole number.

Value	Round towards zero	Round away from zero	Round to nearest
-46.97	-46	-47	-47
-50.17	-50	-51	-50
-73.11	-73	-74	-73
50.41	50	51	50
76.72	76	77	77
35.42	35	36	35

12. Determine the IEEE standard 32-bit binary representation for each of the following float values: Show all of your work and include a few comments as to what you are doing at each step to get full credit.

a. 8,018.5
   i. Positive number(S=0)
   ii. 8018=1111 1010 1001 0u
   iii. 0.5*2=1.0 1
   iv. 0.0*2=0.0 0
   v. … 0
   vi. 0 0
   vii. 1111 1010 1001 0.100 0000 0000
   viii. 1. (F = 111 1010 1001 0100 0000 0000) * 10^{13}
   ix. e=13
   x. E=e+127=13+127=140=1000 1100u
   xi. S E F
   xii. 0 1000 0110 111 1010 1001 0100 0000 0000
   xiii. 0100 0101 1111 1010 1001 0100 0000 0000

b. -178.7
   i. Negative number(S=1)
   ii. 178=1011 0010u
   iii. 0.7*2=1.4 1
   iv. 0.4*2=0.8 0
   v. 0.8*2=1.6 1
   vi. 0.6*2=1.2 1
   vii. 0.2*2=0.4 0
   viii. 0.4*2=0.8 0
   ix. 0.8*2=1.6 1
   x. 0.6*2=1.2 1
   xi. 0.2*2=0.4 0
   xii. 0.4*2=0.8 0
   xiii. 0.8*2=1.6 1
   xiv. 1011 0010. 1011 0011 0011 0011
   xv. 1. (F = 011 0010 1011 0011 0011 0011) * 10^{7}
   xvi. e=7
   xvii. E=e+127=134=1000 0110u
   xviii. S E F
   xix. 1 1000 0110 011 0010 1011 0011 0011 0011
   xx. 1100 0011 0011 0010 1011 0011 0011 0011

13. Convert each of the following 32 IEEE 754 single precision bit patterns to its corresponding decimal value (the bits are separated into groups of 4 to make interpretation easier). Show all of your work and include a few comments as to what you are doing at each step.

a. 1100 0001 0001 1110 0000 0000 0000 0000
   i. Negative number (S=1)
   ii. E=1000 0010=130
   iii. e=130-127=3
   iv. F=0011 1100 0000 0000 0000 000
   v. $1.0011\ 1100\ 0000\ 0000\ 0000\ 000 * 10^3$
   vi. 1001. 1 1100 0000 0000 0000 000
   vii. $9 + 2^{-1} + 2^{-2} + 2^{-3}$
   viii. 9.875
   ix. -9.875

b. 0100 0110 0000 1101 0011 1001 1000 0000
   i. Positive number (S=0)
   ii. E=1000 1100=140
   iii. e=140-127=13
   iv. F=0001 1010 0111 0011 0000 000
   v. $1.0001\ 1010\ 0111\ 0011\ 0000\ 000 * 10^{13}$
   vi. 10001101001110. 011 0000 000
   vii. $9038 + 0 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3}$
   viii. +9038.375

14. This question focusses on multiplication of signed operands. Perform the multiplication algorithm (Deck 24, Slide 10) using left shifting of a copy of the multiplicand, and addition, to determine whether it gives the correct result for the following examples, using 4 bit operands and a 4 bit result, and show all steps in performance of the algorithm, and then answer the four accompanying questions after each problem. **None of the values below should be considered to be constants.**

a. Positive multiplicand and negative multiplier (3 * -2)
   i. Should the result fit in 4 bits?
   - Yes, $3 * -2 = 6$ is in the B2T range which is -8 to 7.

   ii. Is the correct result produced?
   - 0 0 1 1
   - * 1 1 1 0
   - ---
   - 0 0 1 1
   - \+ 0 0 1 1
   - ---
   - 0 0 1 0
   - 0 0 1 1
   - ---
   - 1 0 1 0
   - $1010 = -6$

   iii. Is there overflow from any of the addition operations (you can simply say if there is overflow from any one or more of them)? How did you determine overflow??
   - There is no overflow from any of them.

   iv. If there is overflow from any of the addition operations, does the result you get make sense? (That is, if there is overflow, would you expect to get a valid or invalid answer?) Why or why not?
   - Since there is no overflow from any of them, it is not applicable.

b. Negative multiplicand and negative multiplier (-3 * -2)
   i. Should the result fit in 4 bits?
   - Yes, $-3 * -2 = 6$ is in the B2T range which is -8 to 7.

ii. Is the correct result produced?

- 1 1 0 1
- * 1 1 1 0
- ---
- 1 1 0 1
- \+ 1 1 0 1
- ---
- 1 1 1 0
- \+ 1 1 0 1
- ---
- 0 1 1 0
- $0110 = 6$

iii. Is there overflow from any of the addition operations (you can simply say if there is overflow from any one or more of them)? How did you determine overflow??

iv. Overflow **occurs** in this case, since the most significant carry in and carry out are not same at last step which is 1.

v. If there is overflow from any of the addition operations, does the result you get make sense? (That is, if there is overflow, would you expect to get a valid or invalid answer?) Why or why not?

- Since a negative multiply a negative number, therefore, it becomes to a positive number. Hence, the overflow must occur.

Note: Do the intermediate additions one at a time; that is, every time there is a non-zero left-shifted copy of the multiplicand produced by examining the bits in the multiplier, add the left-shifted copy to the prior value of the result. Do not wait and try to add 3 or four left-shifted copies of the multiplicand in one addition operation.