Alexander Miller
252
Homework 1

1. Convert the following decimal numbers to 16-bit 2's Complement Binary Numbers:

1k. 7544

7544 - 4096 = 3448
3448 - 2048 = 1400
1400 - 1024 = 376
376 - 256 = 120
120 - 64 = 56
56 - 32 = 24
24 - 16 = 8
8 - 8 = 0
Therefore, 7544 = 4096 + 2048 + 1024 + 256+ 64 + 32 + 16 + 8
Therefore, 7544 = 2^12 + 2^11 + 2^10 + 2^8 + 2^6 + 2^5 + 2^4 + 2^3
Therefore 7544 = 0001_1101_0111_1000

1l. 2974

2974 - 2048 = 926
926 - 512 = 414
414 - 256 = 158
158 - 128 = 30
30 - 16 = 14
14 - 8 = 6
6 - 4 = 2
2 - 2 = 0
Therefore, 2974 = 2048 + 512 + 256 + 128 + 16 + 8 + 4 + 2
Therefore, 2974 = 2^11 + 2^9 + 2^8 + 2^7 + 2^4 + 2^3 + 2^2 + 2^1
Therefore 2974 = 0000_1011_1001_1110

1m. -671

First we find 671 in binary:
671 - 512 = 159
159 - 128 = 31
31 - 16 = 15
15 - 8 = 7
7 - 4 = 3
3 - 2 = 1
1 - 1 = 0
Therefore, 671 = 512 + 128 + 16 + 8 + 4 + 2 + 1
Therefore, 671 = 2^9 + 2^7 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0
Therefore, 671 = 0000_0010_1001_1111
-671 = ~671 + 1
~671 = 1111_1101_0110_0000
Therefore, -671 = 1111_1101_0110_0001

2. For the following numbers, convert them each to hexadecimal [assume unsigned], octal [assume unsigned], and decimal [assume signed], compute a+b and a-b in binary and indicate overflow and carryout [assume signed]

2j.
a = 0000_1001_0010_1111
b = 0101_1100_0001_1100

a+b (signed)
a+b = 0110_1101_0100_1011 (hard to show work on this when typing)
No overflow (2 positive operands yields a positive result)
No carryout

a-b (signed)
~b = 1010_0011_1110_0011
~b + 1 = -b
Therefore, -b = 1010_0011_1110_0100
a + (-b) = 1010_1101_1111_0111

decimal (signed)
a = 2^0 + 2^1 + 2^2 + 2^3 + 2^5 + 2^8 + 2^11
a = 1 + 2 + 4 + 8 + 32 + 256 + 2048
a = 2351
b = 2^2 + 2^3 + 2^4 + 2^10 + 2^11 + 2^12 + 2^14
b = 4 + 8 + 16 + 1024 + 2048 + 4096 + 16384
b = 23580

hexadecimal (unsigned)
a:
0000 -> 0, 1001 -> 9, 0010 -> 2, 1111 -> 15
So, a = 0x92F
b: 0101 -> 5, 1100 -> 12, 0001 -> 1, 1100 -> 12
So, b = 0x5C1C

octal (unsigned)
a: 100_100_101_111
100 -> 4, 100 -> 4, 101 -> 5, 111->7
So, a = 4457 (base 8)

b: 101_110_000_011_100
101 -> 5, 110 -> 6, 000-> 0, 011->3, 100->4
So, b = 5604 (base 8)

2k.
a = 0011_1001_1110_1111
b = 0011_1101_0010_0010

a+b (signed)
a + b = 0111_0111_0011_0011
No carryout.
No overflow. (2 positive operands yield positive result)

a-b (signed)
~b = 1100_0010_1101_1101
So -b = ~b + 1 = 1100_0010_1101_1110
So a + -b = 1111_1100_1100_1101
No carryout.
No overflow. (Difference of two positive operands will be less than or equal to them in magnitude)

decimal (signed)
a = 2^0 + 2^1 + 2^2 + 2^3 + 2^5 + 2^6 + 2^7 + 2^8 + 2^11 + 2^12 + 2^13
So a = 1+2+4+8+32+64+128+256+2048+4096+8192
So a = 14831

b = 2^1 + 2^5 + 2^8 + 2^10 + 2^11 + 2^12 + 2^13
So b = 2+32+256+1024+2048+4096+8192
So b = 15650

hexadecimal (unsigned)
a = 0011_1001_1110_1111
0011 -> 3, 1001 -> 9, 1110 -> 14, 1111 -> 15
So a = 0x39EF

b = 0011_1101_0010_0010
0011 -> 3, 1101->13, 0010-> 2, 0010->2
So b = 0x3D22

octal (unsigned)
a = 0_011_100_111_101_111
0->0, 011->3, 100->4, 111->7, 101->5, 111->7
So a = 34757 [base 8]

b = 0_011_110_100_100_010
0->0, 011->3, 110->6, 100->4, 100->4, 010->2
So b = 36442 [base 8]

2l.

a = 1111_0010_0010_0111
b = 0010_0011_0011_0011

a+b (signed)
a+b = 0001_0111_0101_1010
Carryout: yes.
Overflow: no [the magnitude of a sum of negative and positive number is necessarily less than or equal to the magnitude of the operands]

a-b (signed)
~b = 1101_1100_1100_1100
~b + 1 = -b = 1101_1100_1100_1101
So a + -b = 1100_1110_1111_0100
Carryout: no.
Overflow: no [a negative number minus a positive number yields a negative number]

decimal (signed)
a = 2^0 + 2^1 + 2^2 + 2^5 + 2^9 + 2^12 + 2^13 + 2^14 - 2^15
So a = 1+2+4+32+512+4096+8192+16384-32768
a = -3545

b = 2^0 + 2^1 + 2^4 + 2^5 + 2^8 + 2^9 + 2^13
So b = 1+2+16+32+256+512+8192
b = 9011

hexadecimal (unsigned)
a = 1111_0010_0010_0111
1111->15, 0010->2, 0010->2, 0111->7
So a = 0xF227

b = 0010_0011_0011_0011
0010->2, 0011->3, 0011->3, 0011->3
So b = 0x2333

octal (unsigned)
a = 1_111_001_000_100_111
1->1, 111->7, 001->1, 100->4, 111->7
So, a = 17147 [base 8]

b = 0_010_001_100_110_011
0->0, 010->2, 001->1, 100->4, 110->6, 011->3
So, b = 21463 [base 8]

3i.

Put tonto + hermit into the variable ribbon

```
add $s7, $s3, $s4       # $s7 = tonto + hermit
la $t0, ribbon          # $t0 = address of ribbon
sw $s7, 0($t0)          # store tonto + hermit in ribbon
```

3j.

If (clear == falls), put 1 into register $s3, else put 0 in it

```
la $t0, falls           #find address of falls ($t0)
lw $s7, 0($t0)          # load value of falls ($s7)
bne $s5, $s7, L1        # if clear != falls, skip down to L1
addi $s3, $zero, 1      #put 1 in $s3
j L2                    # skip line L1
L1: add $s3, $zero, $zero   # put 0 in $s3
L2:
```

3k.

Put tonto*3 + kaibab - creek into register $t7

```
add $s3, $s3, $s3       # $s3 = 2*tonto
add $s3, $s3, $s3       # $s3 = 3*tonto
la $t0, kaibab          # get kaibab address
lw $s7, 0($t0)          # read in value of kaibab
add $t7, $s3, $s7       # $t7 = tonto*3 + kaibab
sub $t7, $t7, $s6       # $t7 = tonto*3 - creek
```