Binary Numbers Again

Recall than N binary digits (N bits) can represent unsigned integers from 0 to 2^N-1.

4 bits = 0 to 15
8 bits = 0 to 255
16 bits = 0 to 65535

Besides simply representation, we would like to also do arithmetic operations on numbers in binary form. Principle operations are addition and subtraction.

BR 8/99

Binary Arithmetic, Subtraction

The rules for binary arithmetic are:

- 0 + 0 = 0, carry = 0
- 1 + 0 = 1, carry = 0
- 0 + 1 = 1, carry = 0
- 1 + 1 = 0, carry = 1

The rules for binary subtraction are:

- 0 - 0 = 0, borrow = 0
- 1 - 0 = 1, borrow = 0
- 0 - 1 = 1, borrow = 1
- 1 - 1 = 0, borrow = 0

Borrows, Carries from digits to left of current of digit.

Binary subtraction, addition works just the same as decimal addition, subtraction.

BR 8/99

Binary, Decimal addition

Decimal

```
  34
+ 17
------
  51
```

from LSD to MSD:
7+4 = 1; with carry out of 1 to next column

1 (carry) + 3 + 1 = 5.
answer = 51.

Binary

```
  % 101011
+ % 000001
---------------
     101100
```

From LSB to MSB:
1+1 = 0, carry of 1
1 (carry)+1+0 = 0, carry of 1
1 (carry)+0 + 0 = 1, no carry
1 +0 = 1
0 + 0 = 0
1 + 0 = 1
answer = % 101100

BR 8/99

Subtraction

Decimal

900
- 001

899

0-1 = 9; with borrow of 1 from next column
0 -1 (borrow) - 0 = 9, with borrow of 1
9 - 1 (borrow) - 0 = 8.
Answer = 899.

Binary

% 100
- % 001

011

0-1 = 1; with borrow of 1 from next column
0 -1 (borrow) - 0 = 1, with borrow of 1
1 - 1 (borrow) - 0 = 0.
Answer = % 011.

BR 8/99

Hex Addition

$ 3A
+ $ 28

$ 62

A+8 = 2; with carry out of 1 to next column

1 (carry) + 3 + 2 = 6.
answer = $ 62.

Decimal check.

$ 3A = 3 x 16 + 10
= 58
$28 = 2 x 16 + 8
= 40
58 + 40 = 98

$ 62 = 6 x 16 + 2
= 96 + 2 = 98!!

BR 8/99

Hex addition again

Why is $A + $8 = 2 with a carry out of 1?

The carry out has a weight equal to the BASE (in this case 16). The digit that gets left is the excess (BASE - sum).

$A + $8 = 10 + 8 = 18.

18 is GREATER than 16 (BASE), so need a carry out!

Excess is 18 - BASE = 18 - 16 = 2, so '2' is digit.

Exactly the same thing happens in Decimal.
5 + 7 = 2, carry of 1.
5 + 7 = 12, this is greater than 10!.
So excess is 12 - 10 = 2, carry of 1.

BR 8/99

Hex Subtraction

$ 34
- $ 27

$ 0D

4-7 = D; with borrow of 1 from next column

3 - 1 (borrow) - 2 = 0.
answer = $ 0D.

Decimal check.

$ 34 = 3 x 16 + 4
= 52
$27 = 2 x 16 + 7
= 39
52 - 39 = 13

$ 0D = 13 !!

BR 8/99

Hex subtraction again

Why is $4 - $7 = $D with a borrow of 1?

The borrow has a weight equal to the BASE (in this case 16).

BORROW +$4 - $7 = 16 + 4 -7 = 20 -7 = 13 = $D.

$D is the result of the subtraction with the borrow.

Exactly the same thing happens in decimal.
3 - 8 = 5 with borrow of 1
borrow + 3 - 8 = 10 + 3 - 8 = 13 - 8 = 5.

BR 8/99

Fixed Precision

With paper and pencil, I can write a number with as many digits as I want:

1,027,80,032,034,532,002,391,030,300,209,399,302,992,092,920

A microprocessor or computing system usually uses FIXED PRECISION for integers; they limit the numbers to a fixed number of bits:

$	AF4500239DEFA231	64 bit number, 16 hex digits
$	9DEFA231	32 bit number, 8 hex digits
$	A231	16 bit number, 4 hex digits
$	31	8 bit number, 2 hex digits

High end microprocessors use 64 or 32 bit precision; low end microprocessors use 16 or 8 bit precision.

BR 8/99

Unsigned Overflow

In this class I will use 8 bit precision most of the time, 16 bit occassionally.

Overflow occurs when I add or subtract two numbers, and the correct result is a number that is outside of the range of allowable numbers for that precision. I can have both unsigned and signed overflow (more on signed numbers later)

8 bits -- unsigned integers 0 to 2^8 -1 or 0 to 255.

16 bits -- unsigned integers 0 to 2^{16}-1 or 0 to 65535

BR 8/99

Unsigned Overflow Example

Assume 8 bit precision; ie. I can't store any more than 8 bits for each number.

Lets add 255 + 1 = 256. The number 256 is OUTSIDE the range of 0 to 255! What happens during the addition?

```
  255  =   $ FF
+   1  =   $ 01
-------------------
  256  /=   $00
```

/= means Not Equal

$F + 1 = 0, carry out
$F + 1 (carry) + 0 = 0, carry out
Carry out of MSB falls off end, No place to put it!!!
Final answer is WRONG because could not store carry out.

BR 8/99

Unsigned Overflow

A carry out of the Most Significant Digit (MSD) or Most Significant Bit (MSB) is an OVERFLOW indicator for addition of UNSIGNED numbers.

The correct result has overflowed the number range for that precision, and thus the result is incorrect.

If we could STORE the carry out of the MSD, then the answer would be correct. But we are assuming it is discarded because of fixed precision, so the bits we have left are the incorrect answer.

BR 8/99

Signed Integer Representation

We have been ignoring large sets of numbers so far; ie. the sets of signed integers, fractional numbers, and floating point numbers.

We will not talk about fractional number representation (10.3456) or floating point representation (i.e. 9.23×10^{13}).

We WILL talk about signed integer representation.

The *PROBLEM* with signed integers (- 45, + 27, -99) is the SIGN! How do we encode the sign?

The sign is an extra piece of information that has to be encoded in addition to the magnitude. Hmmmmm, what can we do??

BR 8/99

Signed Magnitude Representation

Signed Magnitude (SM) is a method for encoding signed integers.

The Most Significant Bit is used to represent the sign. '1' is used for a '-' (negative sign), a '0' for a '+' (positive sign).

The format of a SM number in 8 bits is:

% smmmmmmm

where 's' is the sign bit and the other 7 bits represent the magnitude.

NOTE: for positive numbers, the result is the same as the unsigned binary representation.

BR 8/99

Signed Magnitude Examples (8 bits)

-5 = % 1 0000101 = $ 85
+5 = % 0 0000101 = $ 05
+127 = % 0 1111111 = $ 7F
-127 = % 1 1111111 = $ FF
+ 0 = % 0 0000000 = $ 00
- 0 = % 1 0000000 = % 80

For 8 bits, can represent the signed integers -127 to +127.

For N bits, can represent the signed integers

$$-2^{(N-1)} - 1 \quad \text{to} \quad +2^{(N-1)} - 1$$

BR 8/99

Signed Magnitude comments

Signed magnitude easy to understand and encode. Is used today in some applications.

One problem is that it has two ways of representing 0 (-0, and +0) . Mathematically speaking, no such thing as two representations for zeros.

Another problem is that addition of K + (-K) does not give Zero!

-5 + 5 = $ 85 + $ 05 = $8A = -10 !!!

Have to consider the sign when doing arithmetic for signed magnitude representation.

BR 8/99

Ones Complement Representation

Ones complement is another way to represent signed integers.

To encode a negative number, get the binary representation of its magnitude, then COMPLEMENT each bit. Complementing each bit mean that 1s are replaced with 0s, 0s are replaced with 1s.

What is -5 in Ones Complement, 8 bits?

The magnitude 5 in 8-bits is % 00000101 = $ 05

Now complement each bit: % 11111010 = $FA
$FA is the 8-bit, ones complement number of -5.

NOTE: positive numbers in 1s complement are simply their binary representation.

BR 8/99

Ones Complement Examples

-5 = % 11111010 = $ FA
+5 = % 00000101 = $ 05
+127 = % 01111111 = $ 7F
-127 = % 10000000 = $ 80
+ 0 = % 00000000 = $ 00
- 0 = % 11111111 = $ FF

For 8 bits, can represent the signed integers -127 to +127.

For N bits, can represent the signed integers

$$-2^{(N-1)} - 1 \quad \text{to} \quad +2^{(N-1)} - 1$$

BR 8/99

Ones Complement Comments

Still have the problem that there are two ways of representing 0 (-0, and +0) . Mathematically speaking, no such thing as two representations for zeros.

However, addition of K + (-K) now gives Zero!

-5 + 5 = $ FA + $ 05 = $FF = -0 !!!

Unfortunately, K + 0 = K only works if we use +0, does not work if we use -0.

5 + (+0) = $05 + $00 = $05 = 5 (ok)

5 + (-0) = $05 + $FF = $04 = 4 !!! (wrong)

BR 8/99

Twos Complement Representation

Twos complement is another way to represent signed integers.

To encode a negative number, get the binary representation of its magnitude, COMPLEMENT each bit, then ADD 1. (get Ones complement, then add 1).

What is -5 in Twos Complement, 8 bits?

The magnitude 5 in 8-bits is % 00000101 = $ 05

Now complement each bit: % 11111010 = $FA
Now add one: $FA + 1 = $FB
$FB is the 8-bit, twos complement representation of -5.

NOTE: positive numbers in 2s complement are simply their binary representation.

BR 8/99

Twos Complement Examples

-5	=	% 11111011	= $ FB	
+5	=	% 00000101	= $ 05	
+127	=	% 01111111	= $ 7F	
-127	=	% 10000001	= $ 81	
-128	=	% 10000000	= $80	(note the extended range!)
+ 0	=	% 00000000	= $ 00	
- 0	=	% 00000000	= $ 00	(only 1 zero!!!)

For 8 bits, can represent the signed integers -128 to +127.

For N bits, can represent the signed integers

$$-2^{(N-1)} \quad \text{to} \quad +2^{(N-1)} - 1$$

Note that negative range extends one more than positive range.

BR 8/99

Twos Complement Comments

Twos complement is the method of choice for representing signed integers.

It has none of the drawbacks of Signed Magnitude or Ones Complement.

There is only one zero, and K + (-K) = 0.

-5 + 5 = $ FB + $ 05 = $00 = 0 !!!

Normal binary addition is used for adding numbers that represent twos complement integers.

BR 8/99

A common Question from Students

A question I get asked by students all the time is :

Given a hex number, how do I know if it is in 2's complement or 1's complement; is it already in 2's complement or do I have put it in 2's complement, etc, yadda,yadda, yadda….

If I write a HEX number, I will ask for a decimal representation if you INTERPRET the encoding as a particular method (i.e, either 2's complement, 1's complement, signed magnitude).

A Hex or binary number BY ITSELF can represent ANYTHING (unsigned number, signed number, character code, colored llamas, etc). You MUST HAVE additional information that tells you what the encoding of the bits mean.

BR 8/99

Example Conversions

$FE as an 8 bit unsigned integer = 254
$FE as an 8 bit signed magnitude integer = -126
$FE as an 8 bit ones complement integer = - 1
$FE as an 8 bit twos complement integer = -2

$7F as an 8 bit unsigned integer = 127
$7F as an 8 bit signed magnitude integer = +127
$7F as an 8 bit ones complement integer = +127
$7f as an 8 bit twos complement integer = +127

To do hex to signed decimal conversion, we need to determine sign (Step 1), determine Magnitude (step 2), combine sign and magnitude (Step 3)

BR 8/99

Hex to Signed Decimal Conversion Rules

Given a Hex number, and you are told to convert to a signed integer (either as signed magnitude, 1s complement, 2s complement)

STEP 1: Determine the sign! If the Most Significant Bit is zero, the sign is positive. If the MSB is one, the sign is negative. This is true for ALL THREE representations: SM, 1s complement, 2s complement.

$F0 = % 11110000 (MSB is '1'), so sign of result is '-'
$64 = % 01100100 (MSB is '0'), so sign of result is '+'.

If the Most Significant Hex Digit is > 7, then MSB = '1' !!!
(eg, $8,9,A,B,C,D,E,F => MSB = '1' !!!)

BR 8/99

Hex to Signed Decimal (cont)

STEP 2 (positive sign): If the sign is POSITIVE, then just convert the hex value to decimal. The representation is the same for SM, 1s complement, 2s complement.
$64 is a positive number, decimal value is
6 x 16 + 4 = 100.

Final answer is +100 regardless of whether encoding was SM, 1s complement, or 2s complement.

$64 as an 8 bit signed magnitude integer = +100
$64 as an 8 bit ones complement integer = +100
$64 as an 8 bit twos complement integer = +100

BR 8/99

Hex to Signed Decimal (cont)

STEP 2 (negative sign): If the sign is Negative, then need to compute the magnitude of the number.

We will use the trick that - (-N) = + N
i.e. Take the negative of a negative number will give you the positive number. In this case the number will be the magnitude.

If the number is SM format, set Sign bit to Zero:
$F0 = % 11110000 => % 01110000 = $70 = 112
If the number is 1s complement, complement each bit.
$F0 = % 11110000 => % 00001111 = $0F = 15
If the number is 2s complement, complement and add one.
$F0 = % 11110000 => %00001111 + 1 = %00010000 = $10 = 16

BR 8/99

Hex to Signed Decimal (cont)

STEP 3 : Just combine the sign and magnitude to get the result.

$F0 as 8 bit Signed magnitude number is -112
$F0 as 8 bit ones complement number is -15
$F0 as 8 bit twos complement number is -16

$64 as an 8 bit signed magnitude integer = +100
$64 as an 8 bit ones complement integer = +100
$64 as an 8 bit twos complement integer = +100

BR 8/99

Signed Decimal to Hex conversion

Step 1: Know what format you are converting to!!! You must know if you are converting the signed decimal to SM, 1s complement, or 2s complement.
Convert +34 to all three formats.
Convert -20 to all three formats

Step 2: Ignore the sign, convert the magnitude of the number to binary.

34 = 2 x 16 + 2 = $ 22 = % 00100010
20 = 1 x 16 + 4 = $ 14 = % 00010100

BR 8/99

Signed Decimal to Hex conversion (cont)

Step 3 (positive decimal number): If the decimal number was positive, then you are finished no matter what the format is!

+34 as an 8 bit SM number is $ 22 = % 00100010
+34 as an 8 bit 1s complement number is $ 22 = % 00100010
+34 as an 8 bit 2s complement number is $ 22 = % 00100010

BR 8/99

Signed Decimal to Hex conversion (cont)

Step 3 (negative decimal number): Need to do more if decimal number was negative. To get the final representation, we will use the trick that:

- (+N) = -N

i.e., if you take the negative of a positive number, get Negative number.

If converting to SM format, set Sign bit to One:
20 = % 00010100 => % 10010100 = $94
If converting to 1s complement, complement each bit.
20 = % 00010100 => % 11101011 = $EB
If converting to 2s complement, complement and add one.
20 = % 00010100 => %11101011 + 1 = %11101100 = $EC

BR 8/99

Signed Decimal to Hex conversion (cont)

Final results:

+34 as an 8 bit SM number is $ 22 = % 00100010
+34 as an 8 bit 1s complement number is $ 22 = % 00100010
+34 as an 8 bit 2s complement number is $ 22 = % 00100010

-20 as an 8 bit SM number is $ 94 = % 10010100
-20 as an 8 bit 1s complement number is $ EB = % 11101011
-20 as an 8 bit 2s complement number is $ EC = % 11101100

BR 8/99

Two's Complement Overflow

Consider two 8-bit 2's complement numbers. I can represent the signed integers -128 to +127 using this representation.

What if I do (+1) + (+127) = +128. The number +128 is OUT of the RANGE that I can represent with 8 bits. What happens when I do the binary addition?

+127 = $ 7F
\+ +1 = $ 01
---
128 /= $80 (this is actually -128 as a twos complement number!!! - the wrong answer!!!)

How do I know if overflowed occurred? Added two POSITIVE numbers, and got a NEGATIVE result.

BR 8/99

Detecting Two's Complement Overflow

Two's complement overflow occurs is:

Add two POSITIVE numbers and get a NEGATIVE result
Add two NEGATIVE numbers and get a POSITIVE result

I CANNOT get two's complement overflow if I add a NEGATIVE and a POSITIVE number together.

The Carry out of the Most Significant Bit means nothing if the numbers are two's complement numbers.

BR 8/99

Some Examples

All hex numbers represent signed decimal in two's complement format.

$ FF = -1
+ $ 01 = + 1

$ 00 = 0

Note there is a carry out, but the answer is correct. Can't have 2's complement overflow when adding positive and negative number.

$ FF = -1
+ $ 80 = -128

$ 7F = +127 (incorrect!!)

Added two negative numbers, got a positive number. Twos Complement overflow.

BR 8/99

Adding Precision (unsigned)

What if we want to take an unsigned number and add more bits to it?

Just add zeros to the left.

128 = $80 (8 bits)
= $0080 (16 bits)
= $00000080 (32 bits)

BR 8/99

Adding Precision (two's complement)

What if we want to take a twos complement number and add more bits to it?
Take whatever the SIGN BIT is, and extend it to the left.

-128 = $80 = % 10000000 (8 bits)
= $FF80 = % 1111111110000000 (16 bits)
= $FFFFFF80 (32 bits)

+ 127 = $7F = % 01111111 (8 bits)
= $007F = % 0000000001111111 (16 bits)
= $0000007F (32 bits)

This is called SIGN EXTENSION. Extending the MSB to the left works for two's complement numbers and unsigned numbers.

BR 8/99

What do you need to know?

- Addition, subtraction of binary, hex numbers
- Detecting unsigned overflow
- Converting a decimal number to Signed magnitude, Ones Complement, Twos Complement
- Converting a hex number in either SM, 1s complement, 2s complement to decimal
- Number ranges for unsigned, SM, 1s complement, 2s complement
- Overflow in 2s complement
- Sign extension in 2s complement

BR 8/99