02-Binary Arithmetic

Text: Unit 1

ECEGR/ISSC 201
Digital Operations and Computations
Winter 2012

Overview

- Binary Addition
- Binary Subtraction
- Binary Multiplication
- Binary Division
- Representation of negative numbers
- 2's Complement
- Binary Coded Decimal Arithmetic

Binary Arithmetic

- Adding numbers in binary is simple and similar in procedure as in decimal
  - 0 + 0 = 0
  - 1 + 0 = 1
  - 0 + 1 = 1
  - 1 + 1 = 0 (and 1 carried over to the next column)
- For example

```
 1101
+1011
11000
```


Binary Arithmetic

- Add

```
 0001
 0001
 1101
+1011
11010
```

- It can be easier to simply add the numbers in groups of two

```
  00010
+ 11000
  11010
```


Binary Arithmetic

- Binary subtraction can be more confusing
  - 0 - 0 = 0
  - 1 - 0 = 1
  - 0 - 1 = 1 (and borrow from next column)
  - 1 - 1 = 0
- Example

```
  02 borrow
 11101
-10011
  1010
```


Example

- Add the binary numbers: 111+101
- Subtract the binary numbers: 10000-11

Example

- Add the binary numbers: 111+101

```
  111
 +101
 1100
```

- Subtract the binary numbers: 10000-11

```
      2-1   2
   2-1 | 2-1 |
 0  ↓  ↓  ↓  ↓
 1  0  0  0  0
 -        1  1
 0  1  1  0  1
```


Multiplication of Binary Numbers

- 0 x 0 = 0
- 1 x 0 = 0
- 0 x 1 = 0
- 1 x 1 = 1

- Example

```
     1101
   x 1011
     1101
    1101
   0000
  1101
 10001111
```


Example

- Multiply 1001 and 0110

Example

- Multiply 1001 and 110
- Example

```
    1001
  x  110
    0000
   1001
  1001
  110110
```


Example

- You can check by converting to decimal and multiplying
  - 1001 = 9
  - 110 = 6
- 9 x 6 = 54
- 110110 = $1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
- 110110= 32 + 16 + 0 + 4 + 2 +0 = 54
- So we did it correctly

Division of Binary Numbers

- Binary division is the same as decimal division
- Only possible quotient values are 1 or 0
- Example 18 divided by 6:

```
     00011
110)10010
   - 110
     0011
      110
     -110
       00
```

Division of Binary Numbers

- Remainders are possible
- Consider 145 divided by 11 = 13.1818

```
        1101
1011)10010001
     1011
      1110
      1011    <- 1011 cannot go into 110
       1101      so move over another place
       1011
         10   Remainder is 10 (which is 2/11=.1818)
```


Negative Numbers

- Several ways of representing negative numbers
- Sign & Magnitude: make the first bit a sign bit
  - If 0, then positive
  - If 1, then negative
- Example: what is the decimal equivalent of 1110 if this convention is used?

Negative Numbers

- Several ways of representing negative numbers
- Sign & Magnitude: make the first bit a sign bit
  - If 0, then positive
  - If 1, then negative
- Example: what is the decimal equivalent of 1110 if this convention is used?
  - First bit = 1
  - Remaining bits: 110 = 6
  - 1110 = -6_{10}

Negative Numbers

- Reduces the magnitude of numbers that can be represented: 2^{n-1}
- Example: 1110
  - One sign bit
  - Three magnitude bits
  - Range of values: [+7, -7]

Negative Numbers

- Sign & magnitude method is conceptually simple, but it makes practical logic circuit design difficult

Negative Numbers

- Add 5 and -6 using sign and magnitude convention using a 4-bit word

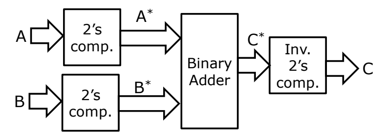
```
+5      0101
-6      1110
-1   (1)0011   Incorrect!
               Ignoring overflow, 0011 = 3
```

Negative Numbers

- Sign & magnitude method is conceptually simple, but it makes practical logic circuit design difficult
  - Circuit design for addition is simple
  - Circuit design for subtraction is more complex
- Other methods:
  - 2's complement
  - 1's complement

2's Complement

- Idea: use a convention that allows subtraction by using the same circuit as addition
- A – B = A + (-B) = C



2's Complement

1. Define number of bits, n, to use
2. Determine the binary n-bit 2's complement representation of A_{10}, denoted A^*
3. Determine the binary n-bit 2's complement representation of B_{10} , denoted B^*
4. Add A^* and B^* using binary addition, ignoring carries to the n + 1 digit
5. Convert the resulting 2's complement number, C^*, to decimal

2's Complement

- The representation of A_{10} and B_{10} using 2's complement notation depends on if they are negative or not
- If positive (or zero)
  - Set first bit to zero
  - Remaining bits are the magnitude (as in the sign & magnitude convention)
  - Note that if n bits are used, the largest value of A_{10} that can be used is 2^{n-1} since the first bit must be 0
  - Example: largest positive number in a 4-bit system is 0111 = 7

2's Complement

- If negative
  - First compute: $N^*_{10} = 2^n + N_{10}$
  - Then convert to binary: $N^* = N^*_{10}$
  - The first bit of the result must be a 1
- Example: let B_{10} = -3 in a 4-bit system
  - $B^*_{10} = 2^4 + -3 = 16 - 3 = 13$
  - B^* = 1101
- Note that if n bits are used, the smallest value of N_{10} that can be used is -2^{n-1} since the first bit must be 1

Example

- Use a 4-bit word to represent the following
  - -2
  - -8
  - 8
- Use a 3-bit word to represent the following
  - -1
  - 0
  - -2

Example

- Use a 4-bit word to represent the following
  - -2: $N^* = 2^n + N_{10} \Rightarrow 16_{10} - 2_{10} = 14_{10} = 1110$
  - -8: => 16 – 8 = 8 = 1000
  - 8 : cannot be represented (1000 is -8)
- Use a 3-bit word to represent the following
  - -1: $8_{10} - 1_{10} = 7_{10} = 111$
  - 0: 0
  - -2: $8_{10} - 2_{10} = 6_{10} = 110$

2's Complement

- Compute 7 – 4 using 2's complement addition using a 4-bit word
  - $A_{10} = 7$
  - $B_{10} = -4$
  - A_{10} is non-negative, so $A^* = 0111$
  - B_{10} is negative so $B^*_{10} = 2^n + B_{10} = 16 - 4 = 12$
  - B^* is the binary representation of 12: 1100
  - Adding A^* and B^*

```
   0111   Ignore the carry since it is a 4-bit system
   1100   Answer is 0011. Since the first digit is a 0,
(1)0011   the answer is 3
```


Inverse 2's Complement

- Compute 4 – 7 using 2's complement addition using a 4-bit word
  - $A_{10} = 4$
  - $B_{10} = -7$
  - A_{10} is non-negative, so $A^* = 0100$
  - B_{10} is negative so $B^*_{10} = 2^n + B_{10} = 16 - 7 = 9$
  - B^* is the binary representation of 9: 1001
  - Adding A^* and B^*

```
0100   Answer is 1101. Since the first digit is a 1,
1001   the answer is negative and we must apply
1101   inverse 2's complement
```


Inverse 2's Complement

- Use $C^*_{10} = 2^n + C_{10}$ to compute inverse 2's complement to solve for C_{10}
  - $C^*_{10} = 1101 = 13$
  - $13 = 16 + C_{10}$
  - $C_{10} = -3$ (correct solution!)

2's Complement

- For a 3-bit system
  - 3 = 011
  - 2 = 010
  - 1 = 001
  - 0 = 000
  - -1 = 111
  - -2 = 110
  - -3 = 101
  - -4 = 100
- For a 4-bit system
  - 7 = 0111
  - 6 = 0110
  - 5 = 0101
  - 4 = 0100
  - 3 = 0011
  - 2 = 0010
  - 1 = 0001
  - 0 = 0000
  - -1 = 1111
  - -2 = 1110
  - -3 = 1101
  - -4 = 1100
  - -5 = 1011
  - -6 = 1010
  - -7 = 1001
  - -8 = 1000

  (-1 through -8: Note pattern)

2's Complement

- We performed subtraction by using an adder circuit
- Easy to create a circuit that performs the 2's complement conversion
  - 2's complement can be found by complementing a number bit-by-bit and then adding 1
- It is possible for errors to occur if there are not enough bits in the system (overflow)

2's Complement Addition

- Assume n = 4
- Case: sum < 2^{n-1}

+3	0011
+4	0100
+7	0111 is this correct?
	yes

2's Complement Addition

- Assume n = 4
- Case: sum ≥ 2^{n-1}

+5	0101
+6	0110
	1011 is this correct?

2's Complement Addition

- Assume n = 4
- Case: sum ≥ 2^{n-1}

+5	0101
+6	0110
	1011 is this correct?

- $5_{10} + 6_{10} = 11_{10}$
- 1011 = -5_{10} (using 2's complement notation)
- Answer is incorrect due to overflow!

2's Complement Addition

- Adding positive and negative numbers (negative result)

+5	0101
−6	1010
−1	1111 correct answer

2's Complement Addition

- Adding positive and negative numbers (negative result)

+5	0101
−6	1010
−1	1111 correct answer

- Adding positive and negative numbers (positive result)

−5	1011
+6	0110
1	(1)0001 correct answer when last carry is ignored

2's Complement Addition

- Adding two negative numbers, |sum| ≤ 2^{n-1}

−3	1101
−4	1100
−7	(1)1001 correct answer last carry is ignored (this is not an overflow error)

2's Complement Addition

- Adding two negative numbers, $|sum| \le 2^{n-1}$

−3	1101	
<u>−4</u>	<u>1100</u>	
−7	(1)1001	correct answer last carry is ignored (this is not an overflow error)

- Adding two negative numbers $|sum| > 2^{n-1}$

−5	1011	
<u>−6</u>	<u>1010</u>	
−11	(1)0101	incorrect answer because of overflow, -11 requires 5 bits

2's Complement Addition

- Errors are easy to detect because they result in either
  - Addition of two positive numbers results in a negative number, or
  - Addition of two negative numbers results in a positive number

1's Complement

- 1's complement (<u>N</u>):
  - A bit-by-bit complement of N
- Example:
  - N = 0101 (5_{10})
  - <u>N</u> = 1010 (-5_{10})
- Similar to 2's complement, but use:
  - <u>N</u> = $(2^n - 1) + N$
- 2's complement and 1's complement relationship:
  - $N^* = \underline{N} + 1$

Binary Coded Decimal Addition

- Recall BCD
- Example:

937.25

1001 0011 0111 . 0010 0101

- This is known as 8-4-2-1 BCD

Binary Coded Decimal Addition

- Consider the addition of two BCD numbers

+4	0100
<u>+5</u>	<u>0101</u>
+9	1001

Binary Coded Decimal Addition

- Consider the addition of two BCD numbers

+4	0100
<u>+5</u>	<u>0101</u>
+9	1001

- What happens if the sum is greater than 9?

+4	0100	
<u>+8</u>	<u>1000</u>	
+12	1100	invalid BCD digit

Binary Coded Decimal Addition

- What should the correct answer be?

+4	0100
+8	1000
+12	1100

- Since 12 > 9, it requires two BCD digits
  - 0001 0010

Binary Coded Decimal Addition

- What value can we add to 1100 to end up with the correct answer (0001 0010)?

+4	0100
+8	1000
+12	1100

Binary Coded Decimal Addition

- What value can we add to 1100 to end up with the correct answer (0001 0010)?

+4	0100
+8	1000
+12	1100

- Add 0110 (six)

1100
0110
10010

Binary Coded Decimal Addition

- Next consider:

+8	1000
+9	1001
+17	10001

- We cannot represent 17 in BCD with one digit, so it is invalid
- Add 0110 (six)

10001
0110
10111 0001 0111 in BCD: 17

Binary Coded Decimal Addition

- It can be shown that to add BCD numbers whose sum is greater than 9, we simply add 6 and carry to the next digit
- If the sum of the BCD numbers is less than or equal to 9, we can sum as usual