ENGR (XMUT) 101
Engineering Technology

A/Prof. Pawel Dmochowski

School of Engineering and Computer Science
Victoria University of Wellington

Week 11 Lecture 1

- Main topics
  - Number system – part 3

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range:

  $$-2^{n-1} <= N <= 2^{n-1}-1$$

  where n is the number of bits available for representing N.

- Note that $2^{n-1}-1 = (011..11)_2$ and

  $$-2^{n-1} = (100..00)_2$$

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range

  $$-2^{n-1} <= N <= 2^{n-1}-1$$

  where n is the number of bits available for representing N. Note that $2^{n-1}-1 = (011..11)_2$ and $-2^{n-1} = (100..00)_2$

- For 2's complement there are more negative numbers than positive.

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range

  $$-2^{n-1} <= N <= 2^{n-1}-1$$

  where n is the number of bits available for representing N. Note that $2^{n-1}-1 = (011..11)_2$ and $-2^{n-1} = (100..00)_2$
- For 2's complement more negative numbers than positive.
- For 1's complement there are two representations for zero.

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range

  $-2^{n-1} <= N <= 2^{n-1}-1$

  where n is the number of bits available for representing N. Note that $2^{n-1}-1 = (011..11)_2$ and $-2^{n-1} = (100..00)_2$
- For 2's complement more negative numbers than positive.
- For 1's complement two representations for zero.
- For a n bit number in base (radix) z there are z^n different unsigned values.

  $(0, 1, \ldots z^{n-1})$

1's Complement Addition

- Adding 1's complement numbers is easy.

1's Complement Addition

- Adding 1's complement numbers is easy.

- For example, to add $+(1100)_2$ and $+(0001)_2$.

1's Complement Addition

- Adding 1's complement numbers is easy.
- For example, to add $+(1100)_2$ and $+(0001)_2$.
- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1's comp.
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 1's comp.

1's Complement Addition

- Adding 1's complement numbers is easy.
- For example, to add $+(1100)_2$ and $+(0001)_2$.

- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1's comp.
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 1's comp.

Step 1: Add binary numbers

Step 2: Add carry to low-order bit

```
    0 1 1 0 0
+   0 0 0 0 1
------------
    0 1 1 0 1
            0
-------------
    0 1 1 0 1
```

1’s Complement Subtraction

- Subtracting 1’s complement numbers is also easy.

1's Complement Subtraction

- Subtracting 1's complement numbers is also easy.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1's complement
  - $(-1)_{10} = -(0001)_2 = 11110_2$ in 1's complement

1's Complement Subtraction

- Subtracting 1's complement numbers is also easy.
- For example, subtract $+(0001)_2$ from $+(1100)_2$.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10}$ = $+(1100)_2$ = 01100_2 in 1's comp.
  - $(-1)_{10}$ = $-(0001)_2$ = 11110_2 in 1's comp.

```
    0 1 1 0 0
 -  0 0 0 0 1
 --------------
    0 1 1 0 0
 +  1 1 1 1 0
 --------------
```

1's comp

Step 1: Take 1's complement of 2nd operand.

1’s Complement Subtraction

- Subtracting 1’s complement numbers is also easy.
- For example, subtract $+(0001)_2$ from $+(1100)_2$.
- Let’s compute $(12)_{10} - (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1’s comp.
  - $(-1)_{10} = -(0001)_2 = 11110_2$ in 1’s comp.

```
      0 1 1 0 0
  -   0 0 0 0 1
  --------------
```

Step 1: Take 1’s complement of 2nd operand.

Step 2: Add the binary numbers.

```
Add      0 1 1 0 0
     +   1 1 1 1 0
     --------------
       1 0 1 0 1 0
```

1's Complement Subtraction

- Subtracting 1's complement numbers is also easy.
- For example, subtract $+(0001)_2$ from $+(1100)_2$.
- Let's compute $(12)_{10} - (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1's comp.
  - $(-1)_{10} = -(0001)_2 = 11110_2$ in 1's comp.

```
      0 1 1 0 0
  -   0 0 0 0 1
  -------------
1's comp

      0 1 1 0 0
Add + 1 1 1 1 0
  -------------
    1 0 1 0 1 0
Add carry     1
  -------------
Final Result  0 1 0 1 1
```

Step 1: Take 1's complement of 2nd operand

Step 2: Add binary numbers

Step 3: Add carry to low order bit

2’s Complement Addition

- Adding 2’s complement numbers is easy.

2's Complement Addition

- Adding 2's complement numbers is easy.
- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 2's complement
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 2's complement

2's Complement Addition

- Adding 2's complement numbers is easy.
- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 2's comp.
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 2's comp.

Step 1: Add binary numbers

Step 2: Ignore carry bit

```
                    0 1 1 0 0
Add          +      0 0 0 0 1
             -----------------
Final Result    0 | 0 1 1 0 1 |
                ^
              Ignore
```

2's Complement Subtraction

- Follow the 3 steps for subtraction.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10}$ = $+(1100)_2$ = 01100_2 in 2's comp.
  - $(-1)_{10}$ = $-(0001)_2$ = 11111_2 in 2's comp.

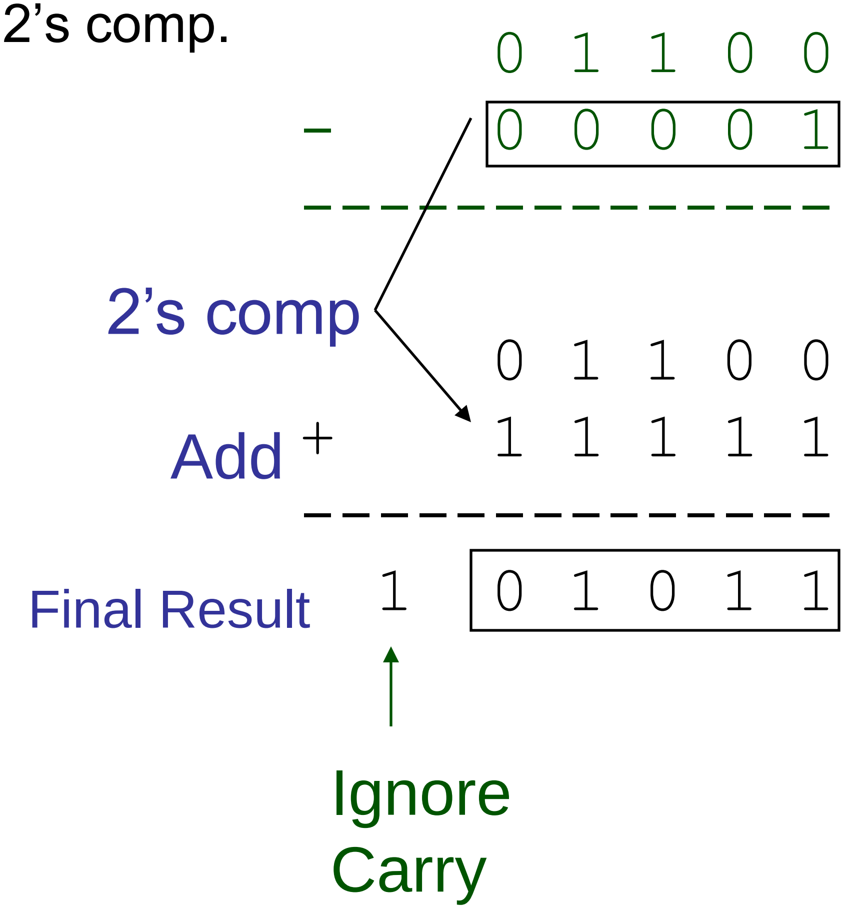
2's Complement Subtraction

- Follow the 3 steps for subtraction.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10}$ = $+(1100)_2$ = 01100_2 in 2's comp.
  - $(-1)_{10}$ = $-(0001)_2$ = 11111_2 in 2's comp.

Step 1: Take 2's complement of 2nd operand

Step 2: Add binary numbers

Step 3: Ignore carry bit



2's Complement Subtraction: Exercise 1

Compute $(13)_{10} - (5)_{10}$ using the 2s complement form.

5 minutes

to complete this exercise!!

2's Complement Subtraction: Exercise 1

- Let's compute $(13)_{10} - (5)_{10}$.

$(13)_{10} = +(1101)_2 = (01101)_2$
$(-5)_{10} = -(0101)_2 = (11011)_2$

- Adding these two 5-bit codes…

```
                     0  1  1  0  1
            +        1  1  0  1  1
                   ----------------
   carry  ------> 1  0  1  0  0  0
```

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

$(01000)_2 = +(1000)_2 = +(8)_{10}$

2's Complement Subtraction: Exercise 2

Compute $(5)_{10} - (12)_{10}$

5 minutes

to complete this exercise!!

2's Complement Subtraction: Exercise 2

- Let's compute $(5)_{10} - (12)_{10}$

$(-12)_{10} = -(1100)_2 = (10100)_2$

$(5)_{10} = +(0101)_2 = (00101)_2$

- Adding these two 5-bit codes…

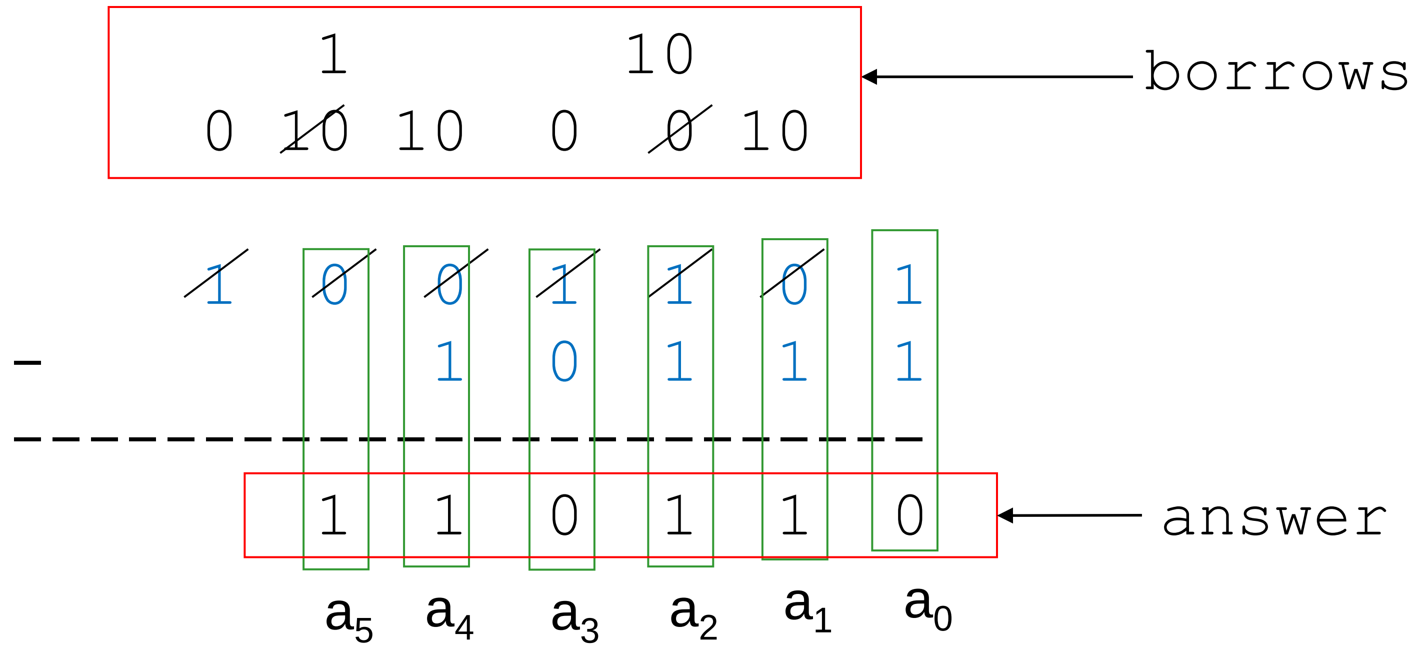
```
    0 0 1 0 1
+   1 0 1 0 0
-------------
    1 1 0 0 1
```

- Here, there is no carry bit and the sign bit is 1. This indicates a negative result, which is what we expect.

  $(11001)_2 = -(7)_{10}$

Binary Subtraction

- We can also perform subtraction (with borrows in place of carries).
- Let's subtract $(10111)_2$ from $(1001101)_2$



Binary Multiplication

What is the decimal equivalent of the 2 binary numbers?

```
   23
  X10
  230
```

```
              1  0  1  1  1
X                1  0  1  0
----------------------------
              0  0  0  0  0
           1  0  1  1  1
        0  0  0  0  0
     1  0  1  1  1
----------------------------
  1  1  1  0  0  1  1  0
```

Binary Multiplication Exercise 5.1

Multiply the 2 binary numbers:

$(1001)_2$ from $(111)_2$

```
                1   0   0   1
X                   1   1   1
-----------------------------
```

2 minutes to complete this exercise!

Binary Multiplication Exercise

Multiply the 2 binary numbers:

$(1001)_2$ from $(111)_2$

```
                  1   0   0   1
x                     1   1   1
-------------------------------
                  1   0   0   1    <- 1st row x 1
              1   0   0   1        <- 1st row x 1
          1   0   0   1            <- 1st row x 1

-------------------------------
          1   1   1   1   1   1
```

Binary Multiplication Exercise 5.1

Multiply the 2 binary numbers:

$(1001)_2$ from $(111)_2$

Self check!

```
  9
X 7
 63
```

```
                     1   0   0   1
X                        1   1   1
-----------------------------------
                     1   0   0   1
                 1   0   0   1
             1   0   0   1
-----------------------------------
             1   1   1   1   1   1
```

$2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$