CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #3c

Data Representation and Number Systems

NUS National University of Singapore | School of Computing

Questions?

Ask at https://app.sli.do/event/qVCWNryB45Bnh6p2HRfnFG

OR

Scan and ask your questions here!
(May be obscured in some slides)

10.6 2s Complement on Addition/Subtraction (1/4)

- Algorithm for addition of integers, A + B:
  1. Perform binary addition on the two numbers.
  2. Ignore the carry out of the MSB.
  3. Check for overflow. Overflow occurs if the 'carry in' and 'carry out' of the MSB are different, or if result is opposite sign of A and B.

- Algorithm for subtraction of integers, A – B:

  A – B = A + (-B)

  1. Take 2s-complement of B.
  2. Add the 2s-complement of B to A.

10.6 Overflow (2/4)

- Signed numbers are of a fixed range.
- If the result of addition/subtraction goes beyond this range, an **overflow** occurs.
- Overflow can be easily detected:
  - *positive* add *positive* → *negative*
  - *negative* add *negative* → *positive*
- Example: 4-bit 2s-complement system
  - Range of value: -8_{10} to 7_{10}
  - $0101_{2s} + 0110_{2s} = 1011_{2s}$
    $5_{10} + 6_{10} = -5_{10}$?! (overflow!)
  - $1001_{2s} + 1101_{2s} = \underline{1}0110_{2s}$ (discard end-carry) $= 0110_{2s}$
    $-7_{10} + -3_{10} = 6_{10}$?! (overflow!)

10.6 2s Complement Addition (3/4)

- Examples: 4-bit system

```
  +3      0011
+ +4    + 0100
----    -------
  +7      0111
----    -------
```

No overflow

```
  -2      1110
+ -6    + 1010
----    -------
  -8     11000
----    -------
```

No overflow

```
  +6      0110
+ -3    + 1101
----    -------
  +3     10011
----    -------
```

No overflow

```
  +4      0100
+ -7    + 1001
----    -------
  -3      1101
----    -------
```

No overflow

```
  -3      1101
+ -6    + 1010
----    -------
  -9     10111
----    -------
```

Overflow!

```
  +5      0101
+ +6    + 0110
----    -------
 +11      1011
----    -------
```

Overflow!

- Which of the above is/are overflow(s)?

10.6 2s Complement Subtraction (4/4)

- Examples: 4-bit system
  - 4 – 7
  - Convert it to 4 + (-7)

```
  +4      0100
+ -7    + 1001
----    -------
  -3      1101
----    -------
```

No overflow

  - 6 – 1
  - Convert it to 6 + (-1)

```
  +6      0110
+ -1    + 1111
----    -------
  +5     10101
----    -------
```

No overflow

  - -5 – 4
  - Convert it to -5 + (-4)

```
  -5      1011
+ -4    + 1100
----    -------
  -9     10111
----    -------
```

Overflow!

- Which of the above is/are overflow(s)?

10.7 1s Complement on Addition/Subtraction (1/2)

- Algorithm for addition of integers, A + B:
  1. Perform binary addition on the two numbers.
  2. If there is a carry out of the MSB, add 1 to the result.
  3. Check for overflow. Overflow occurs if result is opposite sign of A and B.

- Algorithm for subtraction of integers, A – B:

  A – B = A + (-B)
  1. Take 1s-complement of B.
  2. Add the 1s-complement of B to A.

10.7 1s Complement Addition (2/2)

- Examples: 4-bit system

```
  +3      0011
+ +4    + 0100
----    -------
  +7      0111
----    -------
```

No overflow

```
  +5      0101
+ -5    + 1010
----    -------
  -0      1111
----    -------
```

No overflow

```
  -2      1101
+ -5    + 1010
----    -------
  -7     10111
----    +    1
        -------
         1000
        -------
```

No overflow

```
  -3      1100
+ -7    + 1000
----    -------
 -10     10100
----    +    1
        -------
         0101
        -------
```

Overflow!

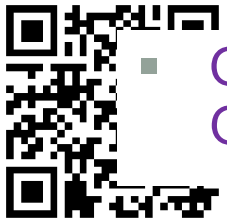
Any overflow?

DLD page 42 – 43 Quick Review Questions
Questions 2-13 to 2-18.

10.8 Excess Representation (1/2)

- Besides sign-and-magnitude and complement schemes, the **excess representation** is another scheme.
- It allows the range of values to be distributed evenly between the positive and negative values, by a simple translation (addition/subtraction).
- Example: Excess-4 representation on 3-bit numbers. See table on the right.

Excess-4 Representation	*Value*
000	-4
001	-3
010	-2
011	-1
100	0
101	1
110	2
111	3



- Questions: What if we use Excess-2 on 3-bit numbers? Or Excess-7?

10.8 Excess Representation (2/2)

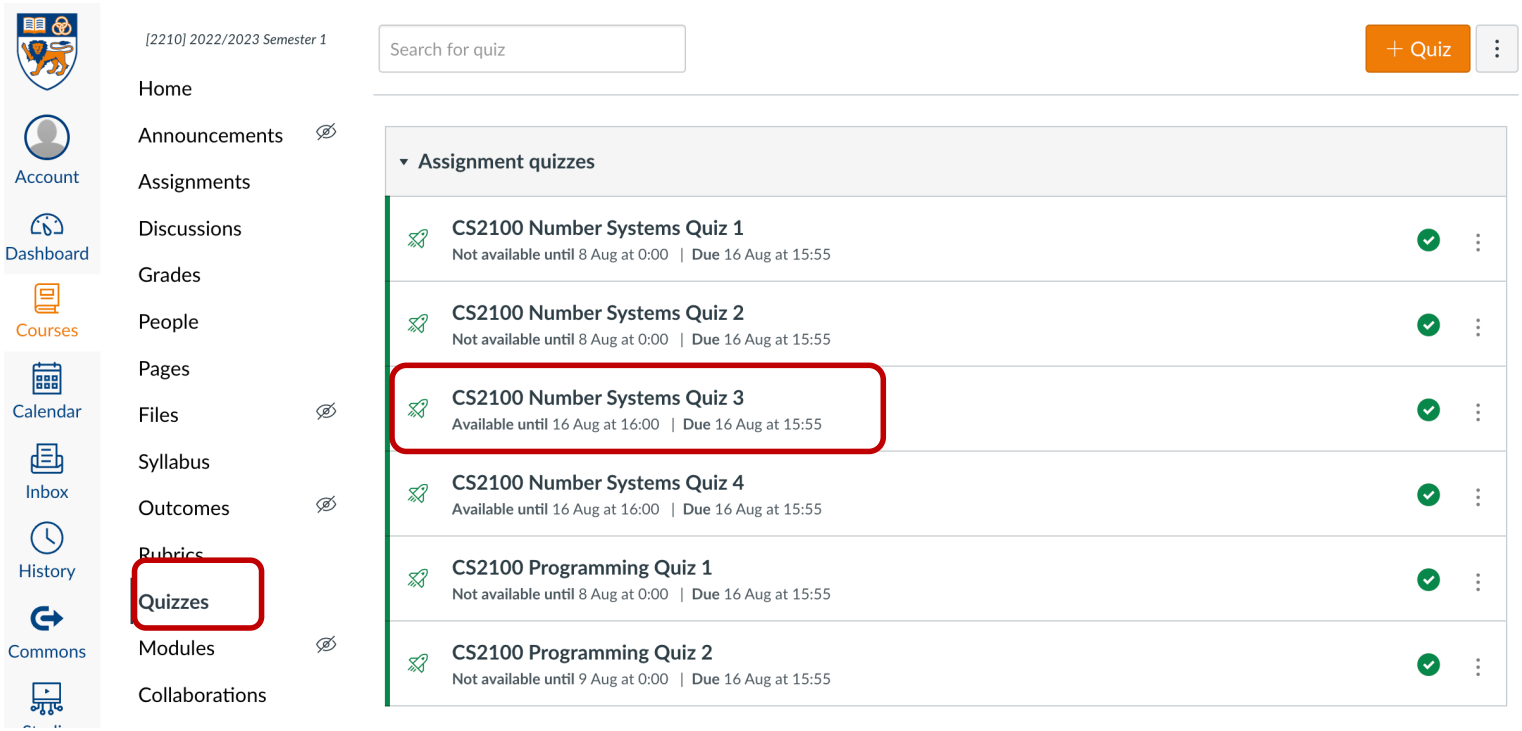
- Example: For 4-bit numbers, we may use excess-7 or excess-8. Excess-8 is shown below.

Excess-8 Representation	*Value*
0000	-8
0001	-7
0010	-6
0011	-5
0100	-4
0101	-3
0110	-2
0111	-1

Excess-8 Representation	*Value*
1000	0
1001	1
1010	2
1011	3
1100	4
1101	5
1110	6
1111	7

Quiz

- Please complete the “CS2100 C Number Systems Quiz 3” in Canvas.
  - Access via the “Quizzes” tool in the left toolbar and select the quiz on the right side of the screen.



End of File