CS2100 COMPUTER ORGANISATION

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

Lecture #3b

Data Representation and Number Systems

NUS National University of Singapore | School of Computing

Questions?

IMPORTANT: DO NOT SCAN THE QR CODE IN THE VIDEO RECORDINGS. THEY NO LONGER WORK

Ask at
https://sets.netlify.app/module/676ca3a07d7f5ffc1741dc65

OR

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

9. ASCII Code (1/3)

- ASCII code and Unicode are used to represent characters ('a', 'C', '?', '\0', etc.)
- ASCII
  - American Standard Code for Information Interchange
  - 7 bits, plus 1 parity bit (odd or even parity)

Character	ASCII Code
0	0110000
1	0110001
. . .	. . .
9	0111001
:	0111010
A	1000001
B	1000010
. . .	. . .
Z	1011010
[	1011011
\	1011100

9. ASCII Code (2/3)

- ASCII table

‘A’: 1000001 (or 65_{10})

	MSBs							
LSBs	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC$_1$	!	1	A	Q	a	q
0010	STX	DC$_2$	“	2	B	R	b	r
0011	ETX	DC$_3$	#	3	C	S	c	s
0100	EOT	DC$_4$	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	O	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

9. ASCII Code (3/3)

(Slide 5 in lecture 3a)

01000110

As an 'int', it is 70

As a 'char', it is 'F'

- Integers (0 to 127) and characters are 'somewhat' interchangeable in C

CharAndInt.c

```
int num = 65;
char ch = 'F';

printf("num (in %%d) = %d\n", num);
printf("num (in %%c) = %c\n", num);
printf("\n");

printf("ch (in %%c) = %c\n", ch);
printf("ch (in %%d) = %d\n", ch);
```

```
num (in %d) = 65
num (in %c) = A

ch (in %c) = F
ch (in %d) = 70
```

Past-Year's Exam Question!

PastYearQn.c

```
int i, n = 2147483640;
for (i=1; i<=10; i++) {
    n = n + 1;
}
printf("n = %d\n", n);
```

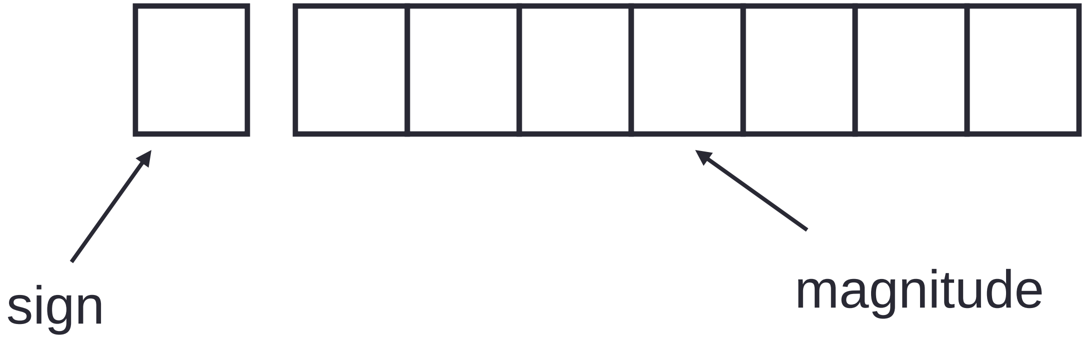
- What is the output of the above code when run on sunfire?
- Is it 2147483650?

10. Negative Numbers

- Unsigned numbers: only non-negative values
- Signed numbers: include all values (positive and negative)
- There are 3 common representations for signed binary numbers:
  - Sign-and-Magnitude
  - 1s Complement
  - 2s Complement

10.1 Sign-and-Magnitude (1/3)

- The sign is represented by a 'sign bit'
  - 0 for +
  - 1 for -
- Eg: a 1-bit sign and 7-bit magnitude format.



- 00110100 ➔ $+110100_2$ = $+52_{10}$
- 10010011 ➔ -10011_2 = -19_{10}

10.1 Sign-and-Magnitude (2/3)

- Largest value: 01111111 = $+127_{10}$
- Smallest value: 11111111 = -127_{10}
- Zeros: 00000000 = $+0_{10}$
  10000000 = -0_{10}
- Range (for 8-bit): -127_{10} to $+127_{10}$
- Question:
  - For an *n*-bit sign-and-magnitude representation, what is the range of values that can be represented?

10.1 Sign-and-Magnitude (3/3)

- To negate a number, just invert the sign bit.
- Examples:
  - How to negate 00100001_{sm} (decimal 33)?
    Answer: 10100001_{sm} (decimal -33)
  - How to negate 10000101_{sm} (decimal -5)?
    Answer: 00000101_{sm} (decimal +5)

10.2 1s Complement (1/3)

- Given a number ***x*** which can be expressed as an *n*-bit binary number, its negated value can be obtained in **1s-complement** representation using:

$$-x = 2^n - x - 1$$

- Example: With an 8-bit number 00001100 (or 12_{10}), its negated value expressed in 1s-complement is:

$$\begin{aligned} -00001100_2 &= 2^8 - 12 - 1 \text{ (calculation done in decimal)} \\ &= 243 \\ &= 11110011_{1s} \end{aligned}$$

(This means that -12_{10} is written as 11110011 in 1s-complement representation.)

10.2 1s Complement (2/3)

- Technique to negate a value: invert all the bits.
- Largest value: $01111111 = +127_{10}$
- Smallest value: $10000000 = -127_{10}$
- Zeros: $00000000 = +0_{10}$
  $11111111 = -0_{10}$
- Range (for 8 bits): -127_{10} to $+127_{10}$
- Range (for *n* bits): $-(2^{n-1} – 1)$ to $2^{n-1} – 1$
- The most significant bit (MSB) still represents the sign: 0 for positive, 1 for negative.

10.2 1s Complement (3/3)

- Examples (assuming 8-bit):

$(14)_{10} = (00001110)_2 = (00001110)_{1s}$

$-(14)_{10} = -(00001110)_2 = (11110001)_{1s}$

$-(80)_{10} = -(\ ?\ )_2 = (\ ?\ )_{1s}$

10.3 2s Complement (1/3)

- Given a number ***x*** which can be expressed as an *n*-bit binary number, its negated value can be obtained in **2s-complement** representation using:

$$-x = 2^n - x$$

- Example: With an 8-bit number 00001100 (or 12_{10}), its negated value expressed in 2s-complement is:

$$\begin{aligned} -00001100_2 &= 2^8 - 12 \text{ (calculation done in decimal)} \\ &= 244 \\ &= 11110100_{2s} \end{aligned}$$

(This means that -12_{10} is written as 11110100 in 2s-complement representation.)

10.3 2s Complement (2/3)

- Technique to negate a value: invert all the bits, then add 1.
- Largest value: 01111111 = $+127_{10}$
- Smallest value: 10000000 = -128_{10}
- Zero: 00000000 = $+0_{10}$
- Range (for 8 bits): -128_{10} to $+127_{10}$
- Range (for *n* bits): -2^{n-1} to $2^{n-1} - 1$
- The most significant bit (MSB) still represents the sign: 0 for positive, 1 for negative.

10.3 2s Complement (3/3)

- Examples (assuming 8-bit):

$$(14)_{10} = (00001110)_2 = (00001110)_{2s}$$

$$-(14)_{10} = -(00001110)_2 = (11110010)_{2s}$$

$$-(80)_{10} = -(\ ?\ )_2 = (\ ?\ )_{2s}$$

Compare with slide 13.

- 1s complement:

$$(14)_{10} = (00001110)_2 = (00001110)_{1s}$$

$$-(14)_{10} = -(00001110)_2 = (11110001)_{1s}$$

10.4 Comparisons

Important!

4-bit system

Positive values

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000

Negative values

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

Past-Year's Exam Question! (Answer)

PastYearQn.c

```
int i, n = 2147483640;
for (i=1; i<=10; i++) {
    n = n + 1;
}
printf("n = %d\n", n);
```

- What is the output of the above code when run on sunfire?
- Is it 2147483650? ✘

- int type in sunfire takes up 4 bytes (32 bits) and uses 2s complement
- Largest positive integer = $2^{31} – 1 =$ 2147483647

1st iteration: n = 2147483641

7th iteration: n = 2147483647

```
  01111 ……. 1111111111
                   + 1
  10000…….0000000000
```

8th iteration: n = -2147483648

9th iteration: n = -2147483647

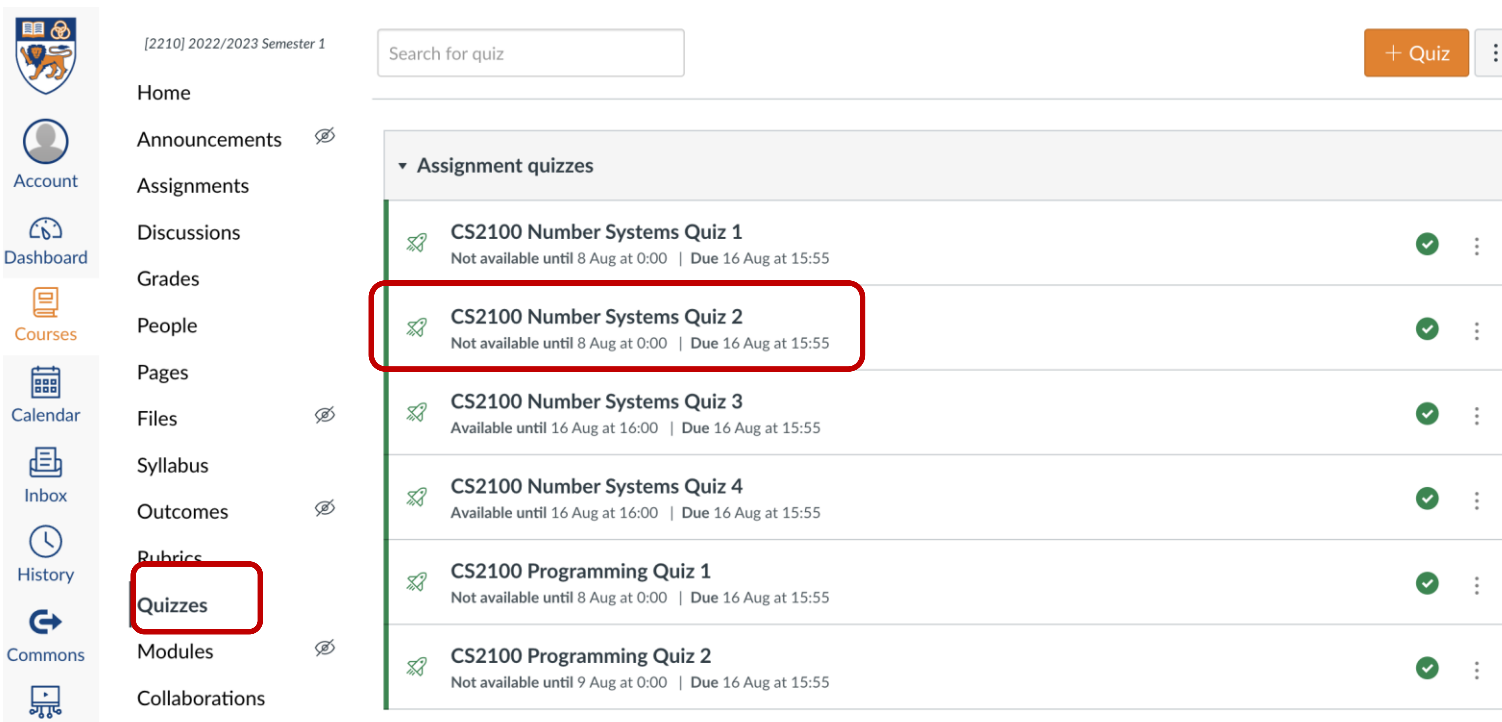
10th iteration: n = -2147483646

10.5 Complement on Fractions

- We can extend the idea of complement on fractions.
- Examples:
  - Negate 0101.01 in 1s-complement
    Answer: 1010.10
  - Negate 111000.101 in 1s-complement
    Answer: 000111.010
  - Negate 0101.01 in 2s-complement
    Answer: 1010.11

Quiz

- Please complete the “CS2100 C Number Systems Quiz 2” 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