

Welcome to CPSC 213!

Check in at the front!!

Randy Zhu

Lab Policy

You should come to labs!!

- Attendance mandatory
- Check-in with me at the start of the lab to get access
- Code to put into PL for attendance
 - **put code in = full marks for lab**
- Arrive no more than 15mins late
- Respectful environment/safe space

About Me

- 3rd year, BSc Honours CS
 - Working on thesis now with Dr. Caroline Lemieux @SPL
- Second time TAing for 213, 3 terms of 210
- Enjoy hiking, cycling, working on programming projects



Upcoming Deadlines

- Assignment 1 due Saturday, Jan 17th

Lab is on PrairieTest

us.prairietest.com

Exercise 1

Hexadecimal Operations

1. Use binary \leftrightarrow hexit table
2. Subtraction in two's complement (CPSC 121)
 1. Flip
 2. Add 1

L1.3. Hexadecimal Operations

1. Convert the following hexadecimal numbers to binary.

0x1010

0b

?

0x9ea5d3b

0b

?

2. Evaluate the following expressions, giving your answer as a hexadecimal number.

0x03 + 0x8a

0x

hex number

?

0xcb + 0x4e

0x

hex number

?

0x9f52 - 0x20a

0x

hex number

?

0x9f52 - 0x20a

0x9f52 - 0x20a

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$

$0x9f52 - 0x20a$

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$
 - $0x020a = 0000_0010_0000_1010$

0x9f52 - 0x20a

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$
 - $0x020a = 0000_0010_0000_1010$
 - flip -> 1111_0010_0000_1010

0x9f52 - 0x20a

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$
 - $0x020a = 0000_0010_0000_1010$
 - flip -> $1111_0010_0000_1010$
 - add1 -> $1111_0010_0000_1011 = f20b$

0x9f52 - 0x20a

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$
 - $0x020a = 0000_0010_0000_1010$
 - flip -> $1111_0010_0000_1010$
 - add1 -> $1111_0010_0000_1011 = f20b$
- Calculate $0x9f52 + 0xf20b$

0x9f52 - 0x20a

- $0x9f52 - 0x20a = 0x9f52 + (-0x020a)$
 - $0x020a = 0000_0010_0000_1010$
 - flip $\rightarrow 1111_0010_0000_1010$
 - add1 $\rightarrow 1111_0010_0000_1011 = f20b$
- Calculate $0x9f52 + 0xf20b$
 - Ignore overflows

Exercise 2

Endianness

- Strategy
 1. Determine sign extension first if RHS type is bigger
 2. Break up into bytes
 - 1 byte = 2 hexits = 8 binary digits
 3. Place by endianness

Big endian					
	0x100	0x101	0x102	0x103	
...	01	23	45	67	...

Little endian					
	0x100	0x101	0x102	0x103	
...	67	45	23	01	...

Memory layout for 0x01234567

2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

Big Endian

0x1000

0x1001

0x1002

0x1003

0x12_34

Little Endian

- 34: least significant bit

0x1000

0x1001

0x1002

0x1003

2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

Big Endian



0x1000

0x1001

0x1002

0x1003

0x12_34

Little Endian

- 34: least significant bit

0x1000

0x1001

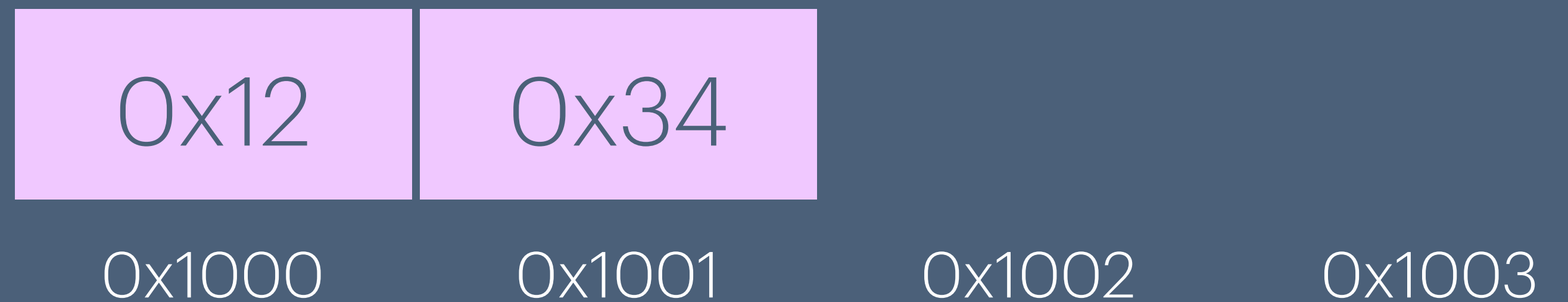
0x1002

0x1003

2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

Big Endian



0x12_34

Little Endian

- 34: least significant bit

0x1000 0x1001 0x1002 0x1003

2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

Big Endian



0x12_34

Little Endian

- 34: least significant bit

0x1000 0x1001 0x1002 0x1003

2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

0x12_34

- 34: least significant bit

Big Endian



Little Endian



2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

0x12_34

- 34: least significant bit

Big Endian



Little Endian



2.3 Layout of short a = 0x1234;

- short in Java: 16 bits, 2 bytes
- Starts at 0x1000

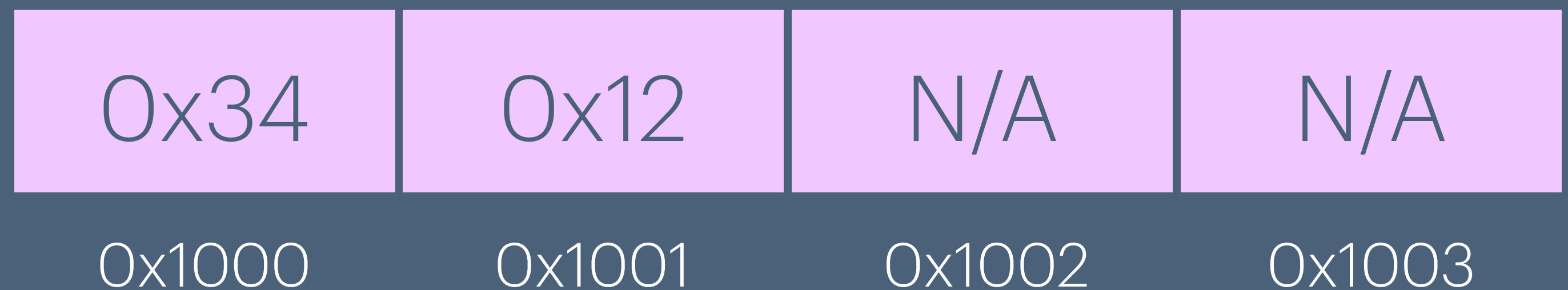
0x12_34

- 34: least significant bit

Big Endian



Little Endian



Lab Code

WZKM

Exercise 3

- All expressions below **are 32 bit (int)**
 - 0x7b
 - 0b0110
 - 42
- Widening conversion (e.g., short -> int, byte -> short) sign extends


```
int a = ((byte) 0xae) << 16;
```

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
 - 0xae = 0x00_00_00_ae

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- 0xae = 0x00_00_00_ae
- (byte) 0xae = 0xae

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- 0xae = 0x00_00_00_ae
- (byte) 0xae = 0xae
- Promote to int, sign extend


```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
 - 0xae = 0x00_00_00_ae
 - (byte) 0xae = 0xae
 - Promote to int, sign extend
 - ((byte) 0xae) = 0xff_ff_ff_ae

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- Shift right 16 bits
- 0xae = 0x00_00_00_ae
- (byte) 0xae = 0xae
- Promote to int, sign extend
- ((byte) 0xae) = 0xff_ff_ff_ae

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- Shift right 16 bits
- 0xae = 0x00_00_00_ae
- 1 hexit = 4 bits
- (byte) 0xae = 0xae
- Promote to int, sign extend
- ((byte) 0xae) = 0xff_ff_ff_ae

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- 0xae = 0x00_00_00_ae
- (byte) 0xae = 0xae
- Promote to int, sign extend
- ((byte) 0xae) = 0xff_ff_ff_ae
- Shift right 16 bits
- 1 hexit = 4 bits
- So shift 4 hexits

```
int a = ((byte) 0xae) << 16;
```

- 0xae is an **int**
- 0xae = 0x00_00_00_ae
- (byte) 0xae = 0xae
- Promote to int, sign extend
- ((byte) 0xae) = 0xff_ff_ff_ae
- Shift right 16 bits
- 1 hexit = 4 bits
- So shift 4 hexits
- **int a = 0xff_ae_00_00**