Bits and Bitwise Operators

Announcements

assign0 out, due Monday 4/9

- Focus is on getting comfortable in unix
- Note instructions for the readme

Piazza

- Great student contributions. Keep it up!

Office hours

- Regular schedule starts next week

Lab signups

SCPD students: expect an email soon with info

Roadmap

Next four weeks: various aspects of C

This week: data representation

- How numbers are stored
- Computer arithmetic
- Limitations

Next week: pointers and memory

Goals for Today

Work with bits as individual units

- Bitwise operators, masks

Use bits to represent C data types

- Number bases (binary, hex)
- Integer types
- Characters

Use gdb to trace programs an inspect values

Definitions

bit (binary digit): a single 1 or 0

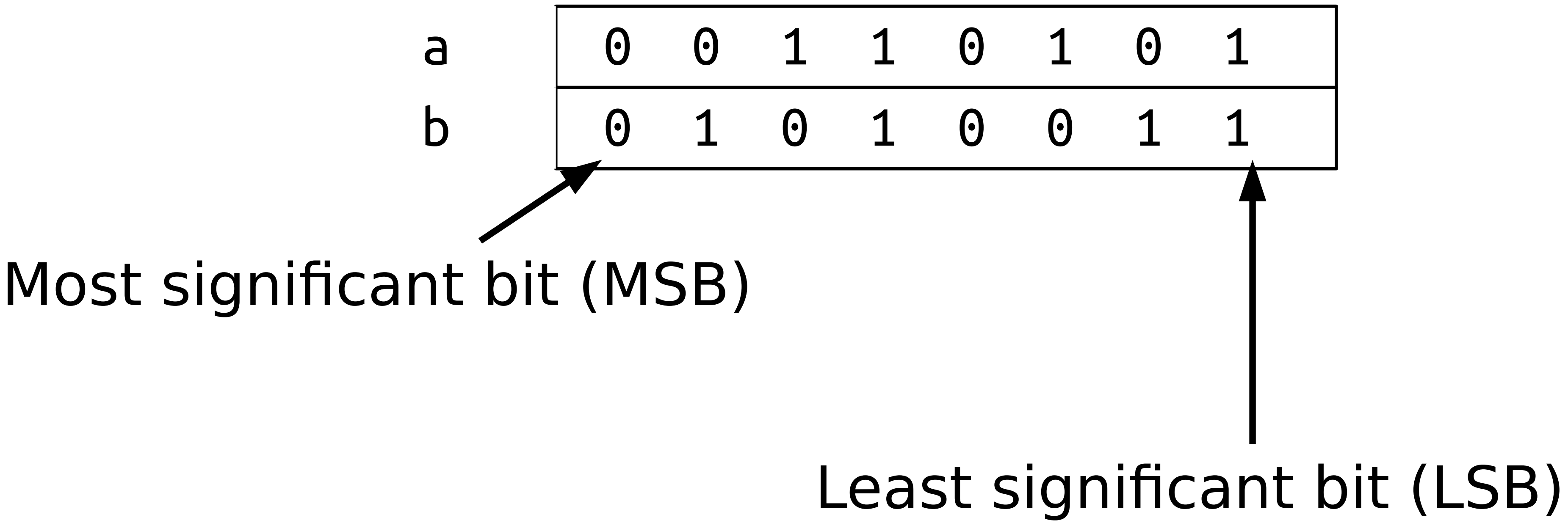
- Can think of as true or false

byte: 8 bits

- Smallest addressable unit
- In C, there’s no byte type
- But char is always one byte

Bitwise Operators

```
unsigned char a, b;
```



a	0	0	1	1	0	1	0	1
b	0	1	0	1	0	0	1	1

Most significant bit (MSB)

Least significant bit (LSB)

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1
AND	a & b	0	0	0	1	0	0	0	1

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1
AND	a & b	0	0	0	1	0	0	0	1
OR	a | b	0	1	1	1	0	1	1	1

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1
AND	a & b	0	0	0	1	0	0	0	1
OR	a | b	0	1	1	1	0	1	1	1
XOR	a ^ b	0	1	1	0	0	1	1	0

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1
AND	a & b	0	0	0	1	0	0	0	1
OR	a | b	0	1	1	1	0	1	1	1
XOR	a ^ b	0	1	1	0	0	1	1	0
NOT	~a	1	1	0	0	1	0	1	0

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1
AND	a & b	0	0	0	1	0	0	0	1
OR	a | b	0	1	1	1	0	1	1	1
XOR	a ^ b	0	1	1	0	0	1	1	0
NOT	~a	1	1	0	0	1	0	1	0
Left shift	a << 2	1	1	0	1	0	1		
Right shift	a >> 3				0	0	1	1	0

Bitwise Operators

```
unsigned char a, b;
```

	a	0	0	1	1	0	1	0	1
	b	0	1	0	1	0	0	1	1

AND	a & b	0	0	0	1	0	0	0	1
OR	a | b	0	1	1	1	0	1	1	1
XOR	a ^ b	0	1	1	0	0	1	1	0

NOT	~a	1	1	0	0	1	0	1	0
Left shift	a << 2	1	1	0	1	0	1	0	0
Right shift	a >> 3	0	0	0	0	0	1	1	0

Code Example: bits.c

So Far

Work with bits as individual units

- Bitwise operators, masks

Use bits to represent C data types

- Number bases (binary, hex)
- Integer types
- Characters

Use gdb to trace programs an inspect values

Binary Polynomial

Decimal:	5	0	7
	10^2	10^1	10^0

$$5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0 = 507$$

Binary Polynomial

Decimal:	5	0	7
	10^2	10^1	10^0

$$5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0 = 507$$

Binary:	0	1	1	0	1	0	1	1
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$64 + 32 + 8 + 2 + 1 = 107$$

Number Bases

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111

Decimal:	16	17
Binary:	10000	10001

Number Bases

```
Decimal:    0    1    2    3    4    5    6    7
Binary:  0000 0001 0010 0011 0100 0101 0110 0111
Hex:        0    1    2    3    4    5    6    7

Decimal:    8    9   10   11   12   13   14   15
Binary:  1000 1001 1010 1011 1100 1101 1110 1111
Hex:        8    9    a    b    c    d    e    f

Decimal:    16    17
Binary:  10000 10001
Hex:        10    11
```

Hexadecimal (base 16)

- Compact, easy conversion to/from binary
- Use in C code with 0x prefix

Conversion

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hex:	0	1	2	3	4	5	6	7

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hex:	8	9	a	b	c	d	e	f

Binary	Hex	Polynomial	Decimal
0101 1100			

Conversion

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hex:	0	1	2	3	4	5	6	7

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hex:	8	9	a	b	c	d	e	f

Binary	Hex	Polynomial	Decimal
0101 1100	0x5c		

Conversion

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hex:	0	1	2	3	4	5	6	7

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hex:	8	9	a	b	c	d	e	f

Binary	Hex	Polynomial	Decimal
0101 1100	0x5c	64 + 16 + 8 + 4	92

Conversion

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hex:	0	1	2	3	4	5	6	7

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hex:	8	9	a	b	c	d	e	f

Binary	Hex	Polynomial	Decimal
0101 1100	0x5c	64 + 16 + 8 + 4	92
		128 + 16 + 4 + 2	150

Conversion

Decimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hex:	0	1	2	3	4	5	6	7

Decimal:	8	9	10	11	12	13	14	15
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hex:	8	9	a	b	c	d	e	f

Binary	Hex	Polynomial	Decimal
0101 1100	0x5c	64 + 16 + 8 + 4	92
1001 0110	0x96	128 + 16 + 4 + 2	150

Note: Same number, different representation

Range and Data Types

- **1 byte = 8 bits = 2 hex digits**
    - `0xff = 1111 1111 (bin) = 255`
- **C integer data types (unsigned)**
    - `char`: 1 byte, 0 to 255
    - `short`: 2 bytes, 0 to ~65,000
    - `int`: 4 bytes, 0 to ~4 billion
    - `long`: 8 bytes, 0 to [big number]

ASCII: Representing Characters

Dec	Hex	Char	Dec	Hex	Char
0	0x0	'\0'	65	0x41	'A'
	...		66	0x42	'B'
32	0x20	' '		...	
33	0x21	'!'	90	0x5a	'Z'
	...			...	
48	0x30	'0'	97	0x61	'a'
49	0x31	'1'	98	0x62	'b'
	...			...	
57	0x39	'9'	122	0x7a	'z'
	...			...	

Code and gdb: parity.c

Summary

Work with bits as individual units

- Bitwise operators, masks

Use bits to represent C data types

- Number bases (binary, hex)
- Integer types
- Characters

Use gdb to trace programs an inspect values

Next time: arithmetic and signed integers