CS 241: Systems Programming
Lecture 12. Bits and Bytes 1

Fall 2019

Prof. Stephen Checkoway

Computers use binary

Everything in a computer is stored and manipulated as a collection of bits

- The bits mean something only in how they are used, **not** what they are

Example with 32-bits: `01000011010100110100001101001001`

- As a integer: 1129530185
- As a (single-precision) floating point number: 211.262833
- As a sequence of four ASCII characters: CSCI
- As 32-bit x86 instructions:

```
inc   ebx
push  ebx
inc   ebx
dec   ecx
```

Base 10 review

Base 10 review

Given a decimal (base 10) number 1253_{10}

Base 10 review

Given a decimal (base 10) number 1253_{10}

- 3 ones (10^0)

Base 10 review

Given a decimal (base 10) number 1253_{10}

- 3 ones (10^0)
- 5 tens (10^1)

Base 10 review

Given a decimal (base 10) number 1253_{10}

- 3 ones (10^0)
- 5 tens (10^1)
- 2 hundreds (10^2)

Base 10 review

Given a decimal (base 10) number 1253_{10}

- 3 ones (10^0)
- 5 tens (10^1)
- 2 hundreds (10^2)
- 1 thousand (10^3)

10^3	10^2	10^1	10^0
1	2	5	3

Base 8 (Octal)

Base 8 (Octal)

Only uses the digits 0-7

- In C, literal starts with leading `0`

Base 8 (Octal)

Only uses the digits 0-7

- In C, literal starts with leading `0`

Given a octal (base 8) number 02345

- 5 ones (8^0)
- 4 eights (8^1)
- 3 sixty-fours (8^2)
- 2 five hundred twelves (8^3)

Base 8 (Octal)

Only uses the digits 0-7

- In C, literal starts with leading `0`

Given a octal (base 8) number 02345

- 5 ones (8^0)
- 4 eights (8^1)
- 3 sixty-fours (8^2)
- 2 five hundred twelves (8^3)

8^3	8^2	8^1	8^0
2	3	4	5

Base 16 (Hexadecimal)

Base 16 (Hexadecimal)

Single place has values of 0-15

- Need digits larger than 9. Use A=10, B=11, …, F=15
- In C, starts with a leading **`0x`** or **`0X`**

Base 16 (Hexadecimal)

Single place has values of 0-15

- Need digits larger than 9. Use A=10, B=11, …, F=15
- In C, starts with a leading **`0x`** or **`0X`**

Given a hexadecimal (base 16) number `0x04E5`

- 5 ones (16^0)
- 14 sixteens (16^1)
- 4 two hundred fifty-sixes (16^2)
- 0 four thousand ninety-sixes (16^3)

Base 16 (Hexadecimal)

Single place has values of 0-15

- Need digits larger than 9. Use A=10, B=11, …, F=15
- In C, starts with a leading **`0x`** or **`0X`**

Given a hexadecimal (base 16) number `0x04E5`

- 5 ones (16^0)
- 14 sixteens (16^1)
- 4 two hundred fifty-sixes (16^2)
- 0 four thousand ninety-sixes (16^3)

16^3 16^2 16^1 16^0

Base 16 (Hexadecimal)

Single place has values of 0-15

- Need digits larger than 9. Use A=10, B=11, …, F=15
- In C, starts with a leading `0x` or `0X`

Given a hexadecimal (base 16) number `0x04E5`

- 5 ones (16^0)
- 14 sixteens (16^1)
- 4 two hundred fifty-sixes (16^2)
- 0 four thousand ninety-sixes (16^3)

16^3	16^2	16^1	16^0
0	4	E	5

Base 2 (Binary)

Base 2 (Binary)

Only uses the digits 0 and 1

Base 2 (Binary)

Only uses the digits 0 and 1

Given a binary number `0b0000010011100101`

- 1 2^0, 0 2^1, 1 2^2, 0 2^3
- 0 2^4, 1 2^5, 1 2^6, 1 2^7
- 0 2^8, 0 2^9, 1 2^{10}, 0 2^{11}
- 0 2^{12}, 0 2^{13}, 0 2^{14}, 0 2^{15}

Base 2 (Binary)

Only uses the digits 0 and 1

Given a binary number `0b0000010011100101`

- $1\ 2^0$, $0\ 2^1$, $1\ 2^2$, $0\ 2^3$
- $0\ 2^4$, $1\ 2^5$, $1\ 2^6$, $1\ 2^7$
- $0\ 2^8$, $0\ 2^9$, $1\ 2^{10}$, $0\ 2^{11}$
- $0\ 2^{12}$, $0\ 2^{13}$, $0\ 2^{14}$, $0\ 2^{15}$

$2^{15..12}$ $2^{11..8}$ $2^{7..4}$ $2^{3..0}$

Base 2 (Binary)

Only uses the digits 0 and 1

Given a binary number `0b0000010011100101`

- $1\ 2^0$, $0\ 2^1$, $1\ 2^2$, $0\ 2^3$
- $0\ 2^4$, $1\ 2^5$, $1\ 2^6$, $1\ 2^7$
- $0\ 2^8$, $0\ 2^9$, $1\ 2^{10}$, $0\ 2^{11}$
- $0\ 2^{12}$, $0\ 2^{13}$, $0\ 2^{14}$, $0\ 2^{15}$

$2^{15..12}$	$2^{11..8}$	$2^{7..4}$	$2^{3..0}$
0000	0100	1110	0101

Converting to decimal

Converting to decimal

Multiply and sum up the digit * $base^{position}$ value

Converting to decimal

Multiply and sum up the digit * base^{position} value

- 1253 $= 1*10^3 + 2*10^2 + 5*10^1 + 3*10^0 = 1253$

Converting to decimal

Multiply and sum up the digit * base^{position} value

- `1253` $= 1*10^3 + 2*10^2 + 5*10^1 + 3*10^0 = 1253$
- `02345` $= 2*8^3 + 3*8^2 + 4*8^1 + 5*8^0 = 1253$

Converting to decimal

Multiply and sum up the digit * base^{position} value

- `1253` $= 1*10^3 + 2*10^2 + 5*10^1 + 3*10^0 = 1253$
- `02345` $= 2*8^3 + 3*8^2 + 4*8^1 + 5*8^0 = 1253$
- `0x04E5` $= 0*16^3 + 4*16^2 + 14*16^1 + 5*16^0 = 1253$

Converting to decimal

Multiply and sum up the digit * base^{position} value

- `1253` $= 1*10^3 + 2*10^2 + 5*10^1 + 3*10^0 = 1253$
- `02345` $= 2*8^3 + 3*8^2 + 4*8^1 + 5*8^0 = 1253$
- `0x04E5` $= 0*16^3 + 4*16^2 + 14*16^1 + 5*16^0 = 1253$
- `0b0000010011100101` $= 1253$

Convert the octal value 031 to decimal

A. 7
B. 25
C. 31
D. 49
E. 248

Converting binary to hex

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`
- `0b 0000 0100 1110 0101`

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`
- `0b 0000 0100 1110 0101`

Each block of 4 bits is 0–15

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`
- `0b 0000 0100 1110 0101`

Each block of 4 bits is 0–15

- Replace each with a hex digit

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`
- `0b 0000 0100 1110 0101`

Each block of 4 bits is 0–15

- Replace each with a hex digit
- `0 4 E 5`

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to hex

Just group digits by 4s starting with LSB

- `0b0000010011100101`
- `0b 0000 0100 1110 0101`

Each block of 4 bits is 0–15

- Replace each with a hex digit
- `0 4 E 5`
- `0x04E5`

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting binary to octal

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b0000010011100101`

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b0000010011100101`
- `0b 000 000 010 011 100 101`

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b0000010011100101`
- `0b 000 000 010 011 100 101`

Each block of 3 bits is 0–7

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b000001001110 0101`
- `0b 000 000 010 011 100 101`

Each block of 3 bits is 0–7

- Replace each with a octal digit

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b0000010011100101`
- `0b 000 000 010 011 100 101`

Each block of 3 bits is 0–7

- Replace each with a octal digit
- `0 0 2 3 4 5`

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Converting binary to octal

Just group digits by 3s starting with LSB

- `0b000000010011100101`
- `0b 000 000 010 011 100 101`

Each block of 3 bits is 0–7

- Replace each with a octal digit
- `0 0 2 3 4 5`
- `0002345` (We prepended a 0 to denote octal)

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Convert the 16-bit binary number `0b11001010_11111110` to hex.

(I added an underscore to separate the two groups of 8 bits to improve readability.)

```
A. 0xBEEF

B. 0xCAFE

C. 0xDEAD

D. 0xFACE

E. 0xFEED
```

Alternative view of hex/octal

Binary is a pain to read/work with

- Consider a 64-bit number
  `0b0100110001000000110101101011000000110110110001010100011110110000`
  so long it doesn't fit on one line!

Hex (and much less commonly octal) can be viewed as a much more compact way to represent binary numbers

- **`0x4c40d6b036c547b0`**

Converting hex/octal to binary

1. Take each hexadecimal (or octal) digit
2. Convert it into binary
   - 4 places hex (e.g., A becomes 1010)
   - 3 places octal (e.g., 6 becomes 110)
3. Group them together from LSB to MSB

Converting between Hex & Octal

1. Take hexadecimal number
2. Convert to binary
3. Regroup in clusters of 3 from LSB
4. Generate Octal digits
5. Use reverse process for Octal to Hex

Converting decimal to binary

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`
- `19 / 2 =  9 r 1`

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`
- `19 / 2 =  9 r 1`
- ` 9 / 2 =  4 r 1`

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`
- `19 / 2 =  9 r 1`
- ` 9 / 2 =  4 r 1`
- ` 4 / 2 =  2 r 0`

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`
- `19 / 2 =  9 r 1`
- ` 9 / 2 =  4 r 1`
- ` 4 / 2 =  2 r 0`
- ` 2 / 2 =  1 r 0`

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

```
‣ 39 / 2 = 19 r 1
‣ 19 / 2 =  9 r 1
‣  9 / 2 =  4 r 1
‣  4 / 2 =  2 r 0
‣  2 / 2 =  1 r 0
‣  1 / 2 =  0 r 1
```

Converting decimal to binary

Repeatedly divide by 2, recording remainders

Remainders form the binary number from least to most significant

Example: 39

- `39 / 2 = 19 r 1`
- `19 / 2 =  9 r 1`
- ` 9 / 2 =  4 r 1`
- ` 4 / 2 =  2 r 0`
- ` 2 / 2 =  1 r 0`
- ` 1 / 2 =  0 r 1`
- 39 = `0b100111`

In-class exercise

https://checkoway.net/teaching/cs241/2019-fall/exercises/Lecture-12.html

Grab a laptop and a partner and try to get as much of that done as you can!