

Interpreting Bits: Binary and Hexadecimal Numbers



UCONN

Caiwen Ding

Department of Computer Science and Engineering
University of Connecticut

CSE3666: Introduction to Computer Architecture

Admin

- Tue's lecture is on number systems and Thur's on logical operations.
- lab 1 on Monday (hosted in Blackboard Collaborate in HuskyCT, for all lab sessions) and it is due on Wed. The link to the lab assignment is in HuskyCT (in Labs/lab1).
 - Please get prepared for the lab. Install RARS and run the example code, for example, hello.s
 - Read the lab assignment.
 - Study the slides we discussed last week.
 - It is not enough time in a 50-minute lab session to learn the slides, figuring out issues with RARS, and then complete the lab.
- HW1 is available in HuskyCT. (Due 02/04/2022, 11:59pm EST)
- Office hours posted (HuskyCT and Discord).
- Some students have not set the section role on Discord. Please check Discord messages often. (one TA answering discord questions)

iClicker quiz will be available at HuskyCT after 5pm Jan 27. It will remain open until the end of day, Monday (Jan 31). One attempt. 20 mins.

Outline

- Binary numbers 2
 - Addition and subtraction
- Two's complement numbers $(\text{negate}) \text{Binary}$
 - Addition and subtraction
 - Negation
 - Sign extension
- Hexadecimal numbers 16
- ASCII \rightarrow 'A' 'B' 'C' 'Space'
- Practice

<https://zhijieshi.github.io/cse3666/binarynumbers/>

n	2^n
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Reading: Section 2.4, and hex to binary conversion in Section 2.5

Why Learning Binary and Hexadecimal Numbers?

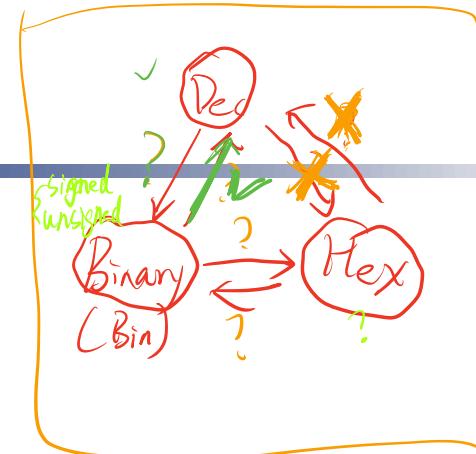
- We are using them in RARS.

Text Segment		Computer		User		Source
Bkpt	Address	Code	Basic			
	0x00400000	0x0fc10517	auipc x10, 0x00000fc10	16:	la a0, msg	# load the address of the string
	0x00400004	0x00050513	addi x10, x10, 0x00000000	17:	li a7, 4	# set the system call number. 4 for printing a string
	0x00400008	0x00400893	addi x17, x10, 0x00000004	18:	ecall	# system call
	0x0040000c	0x00000073	ecall	19:	slli t0, t1, 0	
	0x00400010	0x00631293	slli x5, x6, 0x00000006	20:	loop: add t1, t2, t0	
	0x00400014	0x00538333	add x6, x7, x5	21:	jal ra, foo	
	0x00400018	0x080000ef	jal x1, 0x00000004	22:	bne t1, zero, loop	
	0x0040001c	0xfe031ce3	bne x6, x0, 0xfffffffffc	23:	foo: jal ra, 0	
	0x00400020	0x000008067	jal x0, x1, 0x00000000	24:	exit: la a7, 10	# set the system call number
	0x00400024	0x00a00893	addi x17, x0, 0x00000000	25:	ecall	# system call
	0x00400028	0x00000073	ecall			

- Data and Address often use Hexadecimal Numbers.
 - Memory address per block is increased by 4.

Decimal number

- We are familiar with decimal numbers
 - The radix is 10
- There are ten digits: 0, 1, 2, ..., 8, 9



Example: 3 6 6 6

$$\begin{aligned} 3666 &= 3 \times \textcircled{10}^3 + 6 \times \textcircled{10}^2 + 6 \times \textcircled{10}^1 + 6 \times \textcircled{10}^0 \\ &= 3000 + 600 + 60 + 6 \end{aligned}$$

In this example, each 6 carries different values.

Each time a digit moves to left by one place, the value is increased by 10.

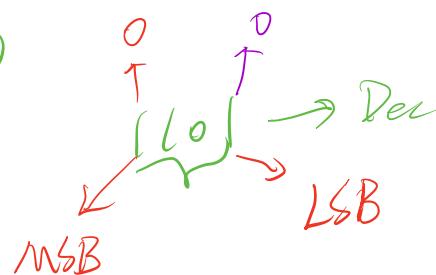
Does the radix have to be 10?
base

Binary numbers

- If the radix is 2, we have binary numbers
 - We only need two digits, 0 and 1
 - Each digit is also called **a bit** (a binary digit)

Given a sequence of bits: (bit stream)

$$b_{n-1} b_{n-2} \dots b_2 b_1 b_0$$



The value is

$$b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

b_{n-1} is the most significant bit (MSB). b_0 is the least significant bit (LSB).

Questions

- What are the decimal representations of the following binary numbers? **What can we learn from these exercises?**

Q1: 1101
3rd 1st
4th

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{\text{Dec}}$$

Q2: 11010
 $(1101) \times 2$

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2 \times (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) + 0 = (26)_{\text{Dec}}$$

Q3: 110100 $(?)_{\text{Dec}}$

$$26 \times 2 = 52$$

Can you quickly find out the answer to Q3?

Dec
80
8x10

Shift bits

Given v ,

- Shift left by one bit, the value becomes $2v$



- Shift right by one bit, the value becomes $v / 2$

(26) 11010 shift right
(13) 1101

Example: convert decimal to binary

Convert decimal number 78 to a binary number.



Find the binary representation of the value denoted by decimal number 78

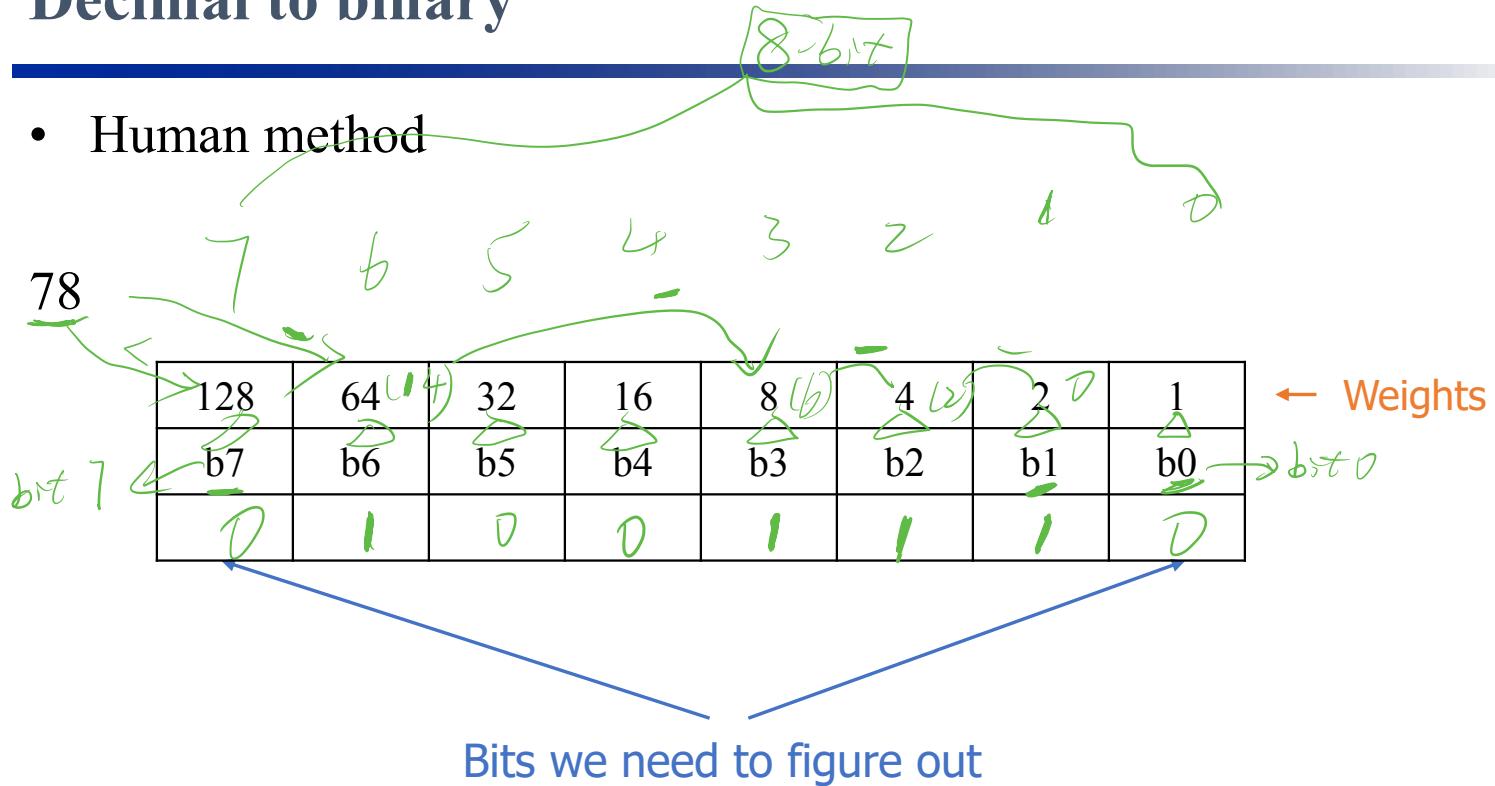
Different representations for the value.

\$100 is the same as one hundred dollars, cien dolares, ...

Computers only deal with bits.

Decimal to binary

- Human method



Which bit can you figure out first?
What is the value of the bit?

Decimal to binary - 2

78

128	64	32	16	8	4	2	1
b7	b6	b5	b4	b3	b2	b1	b0
							0

The more generic method is starting from the right

The right most bit is 0 because 78 is even, or when 78 is divided by 2, the remainder is 0

$$78 = 39 * 2 + 0$$

How about the rest of the bits?

Decimal to binary: Complete steps

We repeat on the quotient we get, until it is 0.

Number	Quotient // 2	Remainder % 2	Expression
78	39	0	$78 = 39 * 2 + 0$
39	19	1	$39 = 19 * 2 + 1$
19	9	1	$19 = 9 * 2 + 1$
9	4	1	...
4	2	0	
2	1	0	
1	0	1	

The answer is 0b 100 1110. ~~XXXXXX~~

We will use 0b or a subscript of 2 for bits

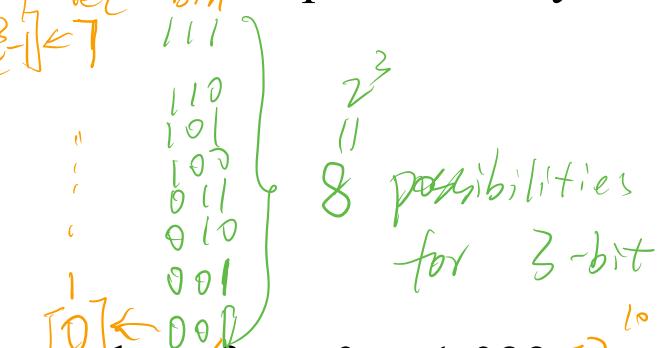
0b1001110 binary Num
 $(1001110)_2$ binary Num

n-bit binary numbers

- Very often, the number of bits available is fixed
- The range of values that can be represented by n bits is

0 to $2^n - 1$

For example:



10 bits can represent any values from 0 to 1,023 $= 2^{10} - 1$

16 bits can represent any values from 0 to 65,535 $= 2^{16} - 1$

32 bits can represent any values from 0 to 4,294,967,295 $= 2^{32} - 1$

Powers of 2:

1K: $2^{10} = 1024$ ✓ 64K: $2^{16} = 65,536$ ✗ 1M: $2^{20} = 1,048,576$ ✗

Question

- ~~At least~~, how many bits do you need to represent 100 different values?

$$\sum_{(b4)}^b \leq 100 \leq \sum_{(128)}^7$$

- A. 5
 - B. 6
 - C. 7
 - D. 8
 - E. Don't know

Addition of numbers

Decimal numbers

$$\begin{array}{r} 6 \ 5 \ 3 \ 9 \\ + 7 \ 1 \ 6 \ 3 \\ \hline 1 \ 0 \ 1 \ 1 \\ \hline 3 \ 7 \ 0 \ 2 \end{array}$$

$(10) \rightarrow (10^m)$

Carries

Binary numbers

$$\begin{array}{r} 0 \ 1 \ 0 \ 1 \\ + 0 \ 1 \ 1 \ 1 \\ \hline 0 \ 1 \ 1 \ 0 \ 0 \end{array}$$

$(2) \rightarrow (2^n)$

Subtraction of binary numbers

0b1101 – 0b0111

Borrows 

$$\begin{array}{r} 01 & 11 & 10 & 01 \\ - & 0 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & 0 \end{array}$$

Annotations: A red arrow points from the 1 in the 3rd column of the minuend to the 1 in the 4th column of the subtrahend. A green arrow points from the 0 in the 4th column of the minuend to the 1 in the 4th column of the result.

We will learn another way to do binary subtraction.

Question

- Find the sum and difference of the following two binary numbers.

10110
01011

10101

Enter the lower four bits of the difference on you iClicker.

How about negative numbers?

Suppose we have 3 bits.

We just learned how to convert each number to decimal.

How can we represent negative numbers with bits?

Bits	In decimal	?
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4 or -1
101	5	-3 or -2
110	6	-2 or -3
111	7	-1 or -4

+1
(000)

?

Two's complement numbers

✓
two's complement

- The most popular method is **two's complement numbers**
 - There are other schemes. Almost all processors now use 2's complement
- Use half of bit patterns for negative values (**which half?**)

Bits	As binary	As 2's complement
000	0	
001	1	
010	2	
011	3	
100	4	-4
101	5	-3
110	6	-2
111	7	-1



111

Reading two's complement numbers

Given an ***n-bit*** 2's complement number

$$b_{n-1} b_{n-2} \dots b_2 b_1 b_0$$

The value is

$$-b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

Other ways:

If the sign is 0, the value is the same as the unsigned.

If the sign is 1, the value is (the unsigned value $- 2^n$).

4-bit two's complement number:

$$\cancel{0b1001} = \cancel{-8} + \cancel{0} + \cancel{0} + \cancel{1} = (-7)_{\text{Dec}}$$

$$\cancel{0b1100} = \cancel{-8} + \cancel{4} + \cancel{0} + \cancel{0} = (-4)_{\text{Dec}}$$

Example: 3-bit binary numbers

We have two ways to interpret the bits

unsigned: binary numbers (without sign)

signed: ← two's complement numbers

Bits	Unsigned	Signed
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

1111111 two's complement

2's complement numbers: range of values

- When dealing with 2's complement numbers, we need to know the number of bits
 - For example, 4-bit, 8-bit, 16-bit, etc., 2's complement number.
 - We should always write leading 0s

Given an ***n*-bit** 2's complement number:

$$b_{n-1} b_{n-2} \dots b_2 b_1 b_0$$

$$-b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

What are the smallest and largest values can be represented by these bits?

Example: Range of values

For n -bit two's complement numbers:

-2^{n-1} can be represented but 2^{n-1} cannot

$$2^{n-1}$$

$$(-2^{n-1})$$

$$2^{n-1} = 3$$

$$[2^{n-1}]$$

$$3$$

Examples

Number of bits	Smallest	Largest
8	$-2^8 = -128$	$2^8 - 1 = 127$
12	$-2^{12} = -4096$	$2^{12} - 1 = 4095$
16	$-2^{16} = -65536$	$2^{16} - 1 = 65535$
32	$-2^{32} = -4,294,967,296$	$2^{32} - 1 = 4,294,967,295$

Why

What are the bits for the smallest values?

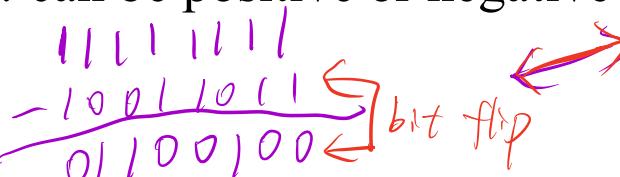
What are the bits for the largest values?

Negate 2's complement numbers

$$\begin{array}{r} 011 \rightarrow 3 \\ ? \leftarrow -3 \end{array}$$

Given the bits representing x , find out the bits for $-x$

x can be positive or negative

Steps: 

why $(-1 -x)$ is the same as bit flip x

- 1. Complement all the bits in x , i.e., $1 \rightarrow 0$ and $0 \rightarrow 1$
- 2. Add 1 to the complemented bits

flip every bit

Explanation:

$$\begin{aligned} \text{The two steps do: } 0 - x &= (-1 + 1) - x = (-1 - x) + 1 \\ \text{Bit pattern of } -1 \text{ is } 111\dots111 & \end{aligned}$$

"Subtract x from -1 " is the same as "flip the bits in x "

Example: negate 2's complement numbers

From $+2$ to -2

~~0000 0010
1111 1101~~ where we start
~~1~~ flip bits
~~1111 1110~~ add 1

From -2 to $+2$

~~1111 1110
0000 0001~~ (1)
~~+1~~ (2)

From -2 to ~~$+2$~~ (2)

~~1111 1110
0000 0001~~ where we start
~~1~~ flip bits
~~0000 0010~~ add 1

→ not the common case

Question

- What is the value of the following 2's complement numbers?

8 bits

1111 0010

(-14)

1110 1010

(-22)

① ②

equation

0000 1101 0

+1

0000 1110

14

✓ (-14)

12 bits

1111 1111 0010

(-14)

1111 1110 1010

(-22)

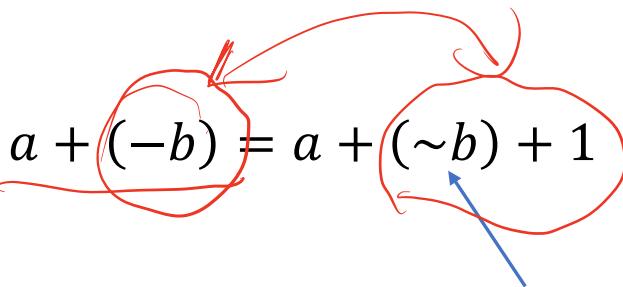
(-14)

How do we know
a number is unsigned or signed?

Addition and subtraction of 2's complement numbers

- **Addition:** Same methods as (unsigned) binary numbers

- **Subtraction:**

$$a - b = a + (-b) = a + (\sim b) + 1$$


We just need an adder!

Flip bits in b

Exercises

~~1~~ ~~4~~

Find the sum and differences of 1111_2 and 0100_2 .

Keep the lower 4 bits of the results.

$$1111 + 0100 \quad (3)_{\text{Dec}}$$

$$1111 - 0100 \quad (-5)_{\text{Dec}}$$

Sign Extension

- Representing a 2's complement number with more bits
 - And preserve the value!
- Replicate the sign bit to the left
 - Compared with unsigned values where we just extend with 0s

~~Examples: 8-bit to 16-bit~~

~~Signed~~ $\{ \begin{array}{l} \text{if sign} = 0, \text{ pad } 0 \\ \text{if sign} = 1, \text{ pad } 1 \end{array} \}$

~~0000 0010 => 0000 0000 0000 0010 (same result for signed and unsigned)~~

3-bit (signed)
 $(111)_{\text{bin}} \leftrightarrow (-1)_{\text{dec}}$
12-bit
 $\leftrightarrow (-1)$

$1111\ 1110 \Rightarrow 1111\ 1111\ 1111\ 1110$ (sign extension for signed)
 $1111\ 1110 \Rightarrow 0000\ 0000\ 1111\ 1110$ (0 extension for unsigned)

Sign extension or 0 extension? Depends on how we interpret bits

Hexadecimal (hex)

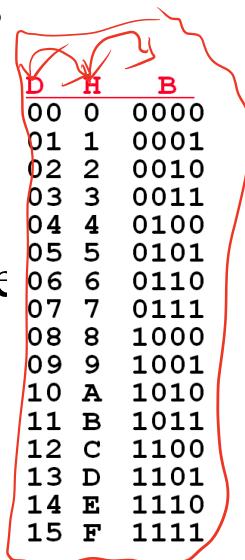
- The radix is 16
- There are 16 digits: 0-9, and a - f

Hex digits	0 - 9	a	b	c	d	e	f
Decimal value	0 - 9	10	11	12	13	14	15

- Use 0x or a subscript of 16 to indicate hexadecimal numbers
- 0xABCD or ABCD₁₆

Hexadecimal numbers are shorter than binary numbers, easier for human to read/write

Computer only know bits!



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

Example: conversion between hex and decimal

Convert 0x3666 to decimal

$$3 \times 16^3 + 6 \times 16^2 + 6 \times 16^1 + 6 \times 16^0 = (13926)_{\text{dec}}$$

Convert 78 to hexadecimal

Number	Quotient // 16	Remainder % 16	Expression
78	4	14 (E)	$78 = 4 * 16 -$
4	0	4 (4)	
	0	0	

So $78 = 0x4E$.

~~0x004E~~
~~04E~~
~~4E~~

The lowest digit is $0xE$ (14).

00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Addition of hexadecimal numbers

Handwritten addition of hex numbers:

6	5	A	9	
+ 7	F	B	3	
<hr/>				
0	1	0		
E	5	C	(12)	

Annotations in red:

- Top row: A red bracket above 'A' is labeled (10) . A red bracket above 'B' is labeled (11) .
- Second row: A red bracket above 'B' is labeled (11) . A red bracket above '3' is labeled a . A red bracket above '9' is labeled b .
- Third row: A red bracket above '1' is labeled (1) . A red bracket above '0' is labeled (1) . A red bracket above the carry line is labeled $(16+1)$.
- Bottom row: A red bracket above 'C' is labeled (12) .

We (humans) convert hex digits to decimal and then convert results back

$$9 + 3 = 12 = 0xC$$

$$\begin{aligned}0xA + 0xB + 0 \\&= 10 + 11 + 1 \\&= 21 \\&= 16 + 5 \\&= 0x15\end{aligned}$$

$$\begin{aligned}0x5 + 0xF + 1 \\&= 0x5 + 0x10 \\&= 0x15\end{aligned}$$

$$1 + 6 + 7 = 14 = 0xE$$

Conversion between hexadecimal and binary

- Hexadecimal is more compact to represent bits
 - Each hex digit represents 4 bits
 - 8 hex digits for 32 bits, and 16 for 64 bits

Mapping between hex digits and bits

0	0b0000	4	0b0100	8	0b1000	C (12)	0b1100
1	0b0001	5	0b0101	9	0b1001	D (13)	0b1101
2	0b0010	6	0b0110	A (10)	0b1010	E (14)	0b1110
3	0b0011	7	0b0111	B (11)	0b1011	F (15)	0b1111

Example: to binary: convert each hex digit to 4 bits ?

~~ECA8 6420₁₆~~
= 1110 | 1100 | 1010 | 1000 | 0110 | 0100 | 0010 | 0000₂

32 binary bits
Every 4 binary bit
= one hex bit

Question

= 8 hex bits

- What is the hexadecimal representation of the following bits?
 - Note that we do not care how the bits are interpreted
 - They could be unsigned or signed

1010 1001 1010

A 9 A

1100 0111 0110

C 7 6

0011
↓
3

11 bta
↓ Hex
0011 → 3

7 11 6 } Pad to 8 } 12 8

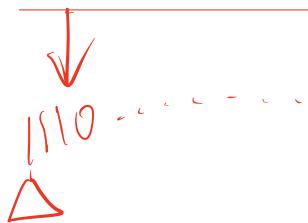
↓ convert

Why hexadecimal?

- More compact for representing the bits
 - Hexadecimal representation is shorter than binary representation
 - Four bits per hex digit
 - Easier for human to read/write/compute
- 32-bit Bin
- Easy to convert between hexadecimal digits and bits

Often, we use hexadecimal digits for short representation of bits and we have many ways to interpret bits

0xECA8 6420 can be unsigned or signed

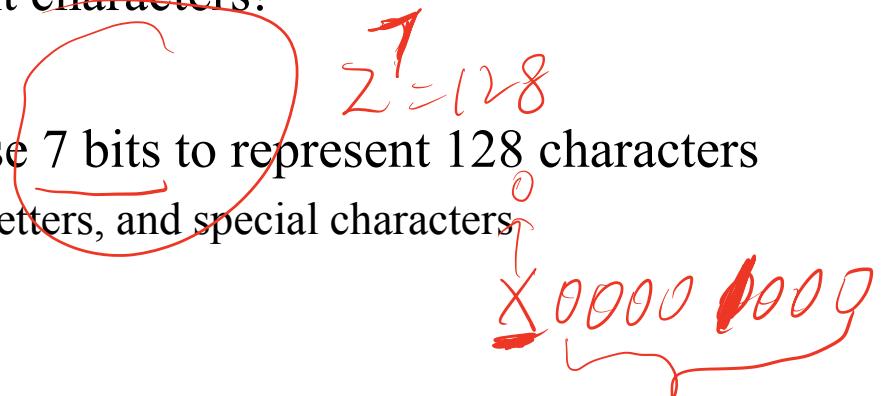


always convert to binary
to check the sign bit.

ASCII: Representing Characters

We also use bits to represent characters!

- ASCII: a standard that use 7 bits to represent 128 characters
 - Including digits, English letters, and special characters
 - And 33 control characters



Example: 65 for 'A', 66 for 'B', 110 for '^'

- An ASCII character is stored in a byte
 - Only use 7 bits. The MSB is always 0
 - Latin-1 extends ASCII to 256 characters (using all 8 bits in a byte)

ASCII Table (partial)

ASCII values are in decimal
Control characters (0 – 31) are not shown

do not memorize!

ASCII value	Character												
32	space	48	0	64	@	80	P	96	`	112	p		
33	!	49	1	65	A	81	Q	97	a	113	q		
34	"	50	2	66	B	82	R	98	b	114	r		
35	#	51	3	67	C	83	S	99	c	115	s		
36	\$	52	4	68	D	84	T	100	d	116	t		
37	%	53	5	69	E	85	U	101	e	117	u		
38	&	54	6	70	F	86	V	102	f	118	v		
39	'	55	7	71	G	87	W	103	g	119	w		
40	(56	8	72	H	88	X	104	h	120	x		
41)	57	9	73	I	89	Y	105	i	121	y		
42	*	58	:	74	J	90	Z	106	j	122	z		
43	+	59	;	75	K	91	[107	k	123	{		
44	,	60	<	76	L	92	\	108	l	124			
45	-	61	=	77	M	93]	109	m	125	}		
46	.	62	>	78	N	94	^	110	n	126	~		
47	/	63	?	79	O	95	_	111	o	127	DEL		

Representation and interpretation

- A value has different representations

$$0b1010 = 0xA = (10)_{\text{dec}}$$

- Computers only deal with bits
- You can write in any format. Compiler/assembler converts it to the same bits, if the representation is supported

- Bits can be interpreted in different ways

- E.g., unsigned numbers, 2's complement numbers
- We are going to learn a few more ways

Memorize

Powers of 2, at least to 1024

$$\begin{aligned}2^5 &= 32 \\2^7 &= 128\end{aligned}$$

$$\begin{aligned}2^{10} &= 1024 \\E &\leftrightarrow 1110 \\(F)_{\text{H}} &= 1111\end{aligned}$$

Mapping between single hex digits and 4 bits

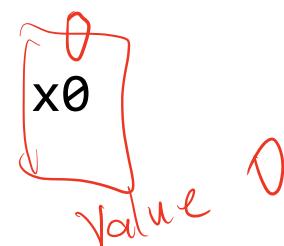
Mapping between single hex digits and numbers in [0, 15]

Example

- The immediates in addi instructions are the same!
 - Character '0' is not the same as number 0

addi s1, x0, '0' *char*
addi s2, x0, 48 *Dec*
addi s3, x0, 0x30 *hex*
s1 == s2 and s1 == s3
lower eight bits are 0011 0000

add s4, x0, x0
s4 != s1



A hand-drawn diagram of a memory location. It consists of a vertical rectangle with a horizontal top edge. Inside the rectangle, the letter 'x0' is written. Above the rectangle, there is a small circle containing the digit '0'. Below the rectangle, the word 'Value' is written in red, with an arrow pointing to the 'x0' inside the rectangle. To the right of the rectangle, the digit '0' is written again.

Study the remaining slides yourself

2's-Complement (signed) numbers

- Need to know the number of bits to read 2's complement numbers
- The left-most bit (the MSB) is the sign bit
 - 0: for non-negative numbers. The value is the same as unsigned.
 - 1: for negative numbers. The value is the unsigned value – 2^n
- Some commonly seen representations:
 - 1: 0b 1111 1111 ... 1111
 - 2: 0b 1111 1111 ... 1110
 - Most-negative: 0b 1000 0000 ... 0000
 - Most-positive: 0b 0111 1111 ... 1111

Example: counting by 5 in hexadecimal

Count by 5 in hexadecimal, starting from 0.

0x0,

0x5, 0xA, 0xF,

0x14, 0x19, 0x1E,

0x23, 0x28, 0x2D,

0x32, 0x37, ...

Do you see any patterns?

Example: radix 10 to radix 1000

We often add thousands separators when writing large numbers

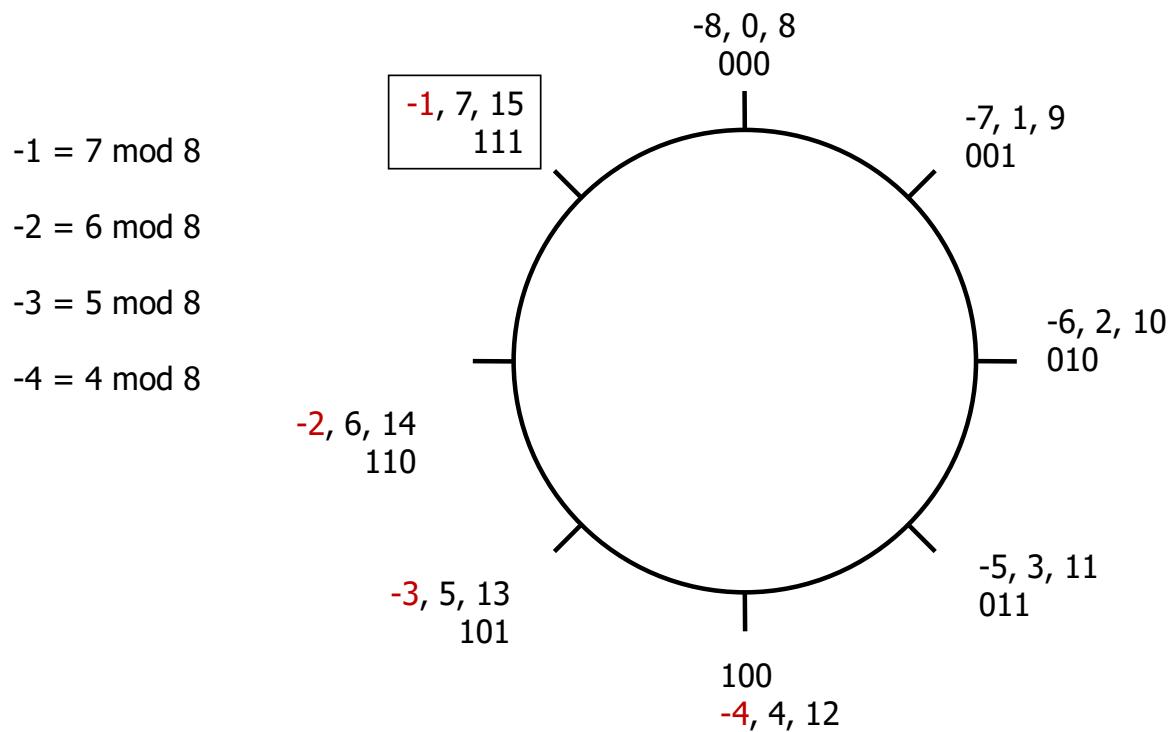
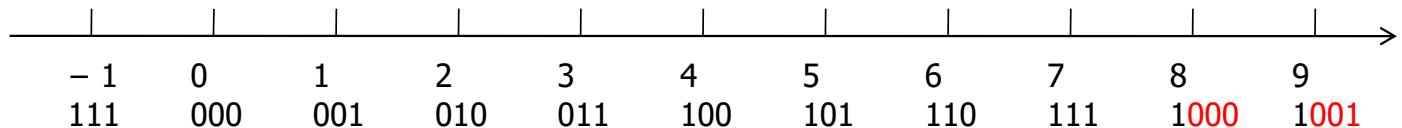
Basically, we convert decimal numbers to radix 1000 numbers

It is easy to do because $1000 = 10^3$

1 light-year = 9,460,730,472,580,800 meters



Why 2's complement number works



Why 2's complement number works

Bits	2's complement picks values in red	unsigned picks values in this column	
000	-8	0	8
001	-7	1	9
010	-6	2	10
011	-5	3	11
100	-4	4	12
101	-3	5	13
110	-2	6	14
111	-1	7	15

Question

- What is the value of the following 4-bit number?

1001

We have to agree on how to interpret the bits first.

unsigned: binary numbers (without sign)

signed: two's complement numbers

unsigned: 9

signed: -7

There will be a problem if a program writes 9 and another program reads -7

Misc

- Convert a binary number to decimal number
 - It means “find out the value of the unsigned binary number and represent the same value in decimal”.
- Convert a decimal number to an n-bit 2’s complement number
 - It means “represent the same value of the decimal number with n-bit 2’s complement number”.
 - The bits can be represented by hexadecimal digits, too
- Hexadecimal number as n-bit 2’s complement number
 - It means “ treat the bits specified by the hexadecimal digits as n-bit 2’s complement number.”