Converting To Binary

1. KNOW your powers of 2:

$2^0 - 1$	$2^3 - 8$	$2^6 - 64$	$2^9 - 512$
$2^1 - 2$	$2^4 - 16$	$2^7 - 128$	$2^{10} - 1024$
$2^2 - 4$	$2^5 - 32$	$2^8 - 256$	

2. Decimal (base 10) to BINARY (Base 2)

ex) 72_{10}

-64 — 2^6

8

-8 — 2^3

0

a. FIND THE LARGEST POWER OF 2 THAT IS less than OR equal to your #.

64 (2^6)

b. SUBTRACT THAT power OF 2 from your #.

c. Repeat steps a + b until you reach 0.

d. NOTE THE powers of 2 that you used to complete a-c.

2^6, 2^3

e. SET UP a set of blanks:
Specifically one more than the largest power of 2 from step d.

(6 + 1 = 7 blanks)

each blank is marked w/ a power of 2 - starting w/ 0 on the RIGHT most blank.

1	0	0	1	0	0	0
2^6	2^5	2^4	2^3	2^2	2^1	2^0

f. For each power of 2 from step d - mark a 1 on the blank. For all other blanks - mark 0.

conclusion:

$72_{10} = 1001000_2$

Converting From Binary

1. Binary to Decimal

$1\emptyset\emptyset1\emptyset\emptyset\emptyset_2$

a. Find the powers of 2 that belong to the 1's in your binary #.

1	~~0~~	0	1	0	0	0
2^6	2^5	2^4	2^3	2^2	2^1	2^0

$2^6 \quad 2^3$

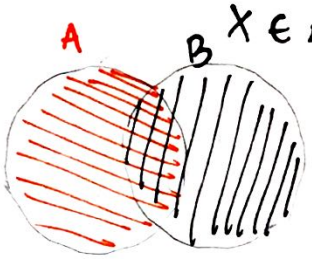
$64 + 8 = \boxed{72_{10}}$

* essentially we are checking which powers of 2 are "used" in the decimal number

INTRO TO SETS

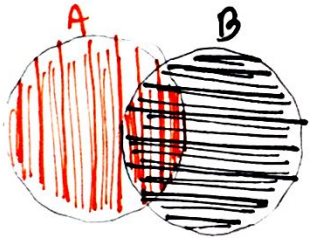
1. set: we say x is an element of X $(x \in X)$ to mean x is an element of the set X.

a. $A \cup B$: A union B → $x \in A \cup B$ IF and ONLY IF (IFF) $x \in A$ OR $x \in B$



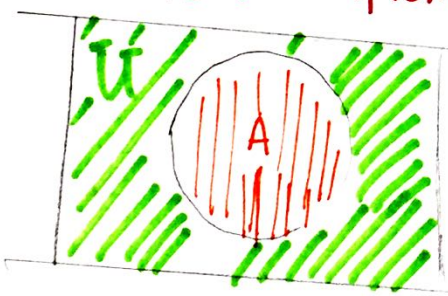
→ anything SHADED is in the set $A \cup B$

b. $A \cap B$: A INTERSECT B → $x \in A \cap B$ IFF $x \in A$ and $x \in B$



The area that is both ORANGE and BLACK is part of the set $A \cap B$

c. $x \in \bar{A}$ ~(A complement) IFF x is not an element of A, in a given universe $(x \notin A)$



anything SHADED GREEN is $\in \bar{A}$.

→ universe: the set of all numbers is dictated by this rule.

ex) $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$

→ this means the only #s in this universe is 1 - 8

d. Null set: $\emptyset$

The empty set - no elements

* $\{\emptyset\}$ is not the null set.

e. cardinality of a set: is the # of $\in$'s in that set.

Propositions + Truth Tables

1. a proposition: must be either true or false.

propositions:

1. $3 * 2 = 10$ (False)
2. It is raining (True)

NOT propositions:

1. $3 * 2$ (not an equation!) (no = sign)
2. shut the door

2. IF p and q are propositions - then we can make other propositions by combining p & q using the logical connections:

a. $\neg p$ or $\sim p$ → not p

b. $p \wedge q$ → p and q

c. $p \vee q$ → p or q

d. $p \rightarrow q$ → if p then q.

e. p xor q → exclusive or

3. Truth tables:

p	¬p
T	F
F	T

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

p	q	p xor q
T	T	F
T	F	T
F	T	T
F	F	F

TRUTH TABLES II

a. $p \to q$: conditional statement

IF p, then q : IF a condition is met, then something else will happen.

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

p: math dept gets additional $60K

q: math dept hires 1 new faculty

p | q

T | T : if math dept gets 60k, they hire ✓ T

T | F : math dept got money, didn't hire ✗ F

F | T : math dept got money, hired faculty

* Tricky → managed to hire faculty thru other means!

→ old faculty retires.

* so this is considered TRUE

F | F math dept doesn't get $, don't hire faculty ✓ True

b. p xor q : exclusive OR

p OR q but not both

p: prisoner executed by hangman

q: prisoner executed by firing squad

p	q	p xor q
T	T	F
T	F	T
F	T	T
F	F	F

p | q

T | T → p xor q is false b/c can't die twice

T | F → ok b/c died by hanging

F | T → ok b/c died by squad

F | F → both false, so overall false.

Systems in Depth

a. We STUDY # systems b/c computers only understand binary signals
   → binary signals are best translated into binary numbers.

b. In machine language programming:
   → use hexadecimal
   → it shortens the binary representation.
   → hex
      → base 16 $\quad \emptyset - 9, A - F$
   ex) $578_{10} =$
   $1\emptyset\emptyset1\emptyset\emptyset\emptyset\emptyset1\emptyset_2 =$ → compressed!
   242_{16}

c. Base-2 # systems will have 2 digits to work with
   ex) · base-2 has 2 digits: $\emptyset, 1$
   · base-8 has 8 digits: $\emptyset, 1, 2, 3, 4, 5, 6, 7$
   · base-10 has 10 digits: $\emptyset - \cdots 1\emptyset$

d. UNDERSTANDING Base-10
   · 54_{10} → $(4 * 10^{\emptyset}) + (5 * 10^{1})$
   ten's place ← 5 → ones place
   $4 + 5\emptyset = 54$

   · 281_{10} → $(1 * 10^{\emptyset}) + (8 * 10^{1}) + (2 * 10^{2})$
   100's ← 2 → 10's ← 8 → 1's
   $1 + 8\emptyset + 2\emptyset\emptyset = 281$

e. understanding Base-2
   $1\emptyset11_2$
   2^3 ← 1 → 2^2 ← $\emptyset$ → 2^1 ← 1 → $2^{\emptyset}$
   $(1 * 2^{\emptyset}) + (1 * 2^{1}) + (0 * 2^{2}) + (1 * 2^{3})$
   $1 + 2 + \emptyset + 8 =$
   11_{10}

(OCTAL TOO)

BINARY System (cont'd

→ Binary digits → BITS : 0 or 1

→ counting in Binary:

0 , 1 , 10 , 11 , 100 , 101 , 110 , 111 ...

0 → one BIT; 10 → 2 BITS; 100 → 3 BITS

*conclusion: with n bits, we can represent 2^n Binary #s.

n = 1 bit

$0_2 = 0_{10}$

$1_2 = 1_{10}$

n = 2 bits

$0_{10} = 00_2$

$1_{10} = 01_2$

$2_{10} = 10_2$

$3_{10} = 11_2$

n = 3 bits

$0_{10} = 000_2$

$1_{10} = 001_2$

$2_{10} = 010_2$

$3_{10} = 011_2$

$4_{10} = 100_2$

$5_{10} = 101_2$

$6_{10} = 110_2$

$7_{10} = 111_2$

Observations:

- lowest #: all 0's
- largest #: all 1's
- $1\,1\,1$ ($2^2\ 2^1\ 2^0$) → $(1*2^0) + (1*2^1) + (1*2^2)$

$1 + 2 + 4 = 7_{10}$

→ Convert FROM BINARY TO OCTAL:

base 8	base 2
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

① OCTAL to Binary:

345_8

3 → 011_2; 4 → 100_2; 5 → 101_2

$011100101_2 = 345_8$

* leading 0's can be dropped!

② Binary to OCTAL →

$011\,100\,101_2$

3 4 5

345_8

SYSTEMS CONT'D

a. OCTAL:

$345_8 \rightarrow (5 * 8^0) + (4 * 8^1) + 3(8^2)$

$(5 * 1) + (4 * 8) + (3 * 64)$

$5 + 32 + 192 = 229_{10}$

b. Why can we just substitute binary bits FOR OCTAL?

↳ recall: $\underset{2^1}{0}\underset{}{1}\underset{4}{1}\underset{}{0}\underset{4+1}{1}\underset{}{0}\underset{}{1}_2$

3 4 5

$= 345_8$

1. Broke into groups of 3
2. ADD leading 0's to any group w/o 3
3. Substitute the binary # for each group

↳ we can do this substitute METHOD BECAUSE:

- OCTAL - 8 bits = 2^3

There is a perfect power of 2 which matches the # of digits in the OCTAL system.

- hexadecimal - 16 bits = 2^4

↳ substitution will work here as well.

- DECIMAL - 10 bits ↳ no even power of 2 cannot substitute

systems: Hexadecimal

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

$A2DE_{16}$

1010 0010 1101 1110

$1010\ 0010\ 1101\ 1110_2 = A2DE_{16}$

↳ To convert to octal: Break into groups of 3:

$001|010|010|110|110|111|110_2$

1 2 2 6 6 7 6

$= 1221667_8 = A2DE_{16}$

$(A * 16^3) + (2 * 16^2) + (D * 16^1) + (E * 16^0)$

$(10 * 4096) + (2 * 256) + (13 \cdot 16) + (14 \cdot 1)$

$= 41694_{10}$

→ converting Binary to hex:

1. Break into groups of 4
2. Add leading 0's
3. convert the groups of 4:

$0011|0010|1101_2$

3 2 D

$= 32D_{16}$

Representing Signed Integers:

→ a complement is a number such that

$$x + (-x) = \emptyset$$

$$37 + (-37) = \emptyset$$

BUT: $101_2 + -101_2 \neq 0$ b/c ...

↳ how do you represent this!? Computer does not understand the sign, it just understands signals, binary ones.

one solu:
use ASCII (-) to store in memory, and represent (-) #'s
↳ BUT: waste of memory! (∧ 8-16 bits for it)

→ TO GET a (-) # : we will consider < 1's complement / 2's complement

RIGHT now: not converting, JUST learning the RULES OF complementing

a. 1's complement: * complement of $\emptyset$ = 1
complement of 1 = $\emptyset$

101_2
↓↓↓
$010_2 = 10_2$ ↙ 1's complement

b. 2's complement:

1. Carry out 1's complement on the binary #.
2. ADD 1

ex) 101_2
↓↓↓
010_2
$+ \quad 1$
011_2 ↙ 2's complement

ex) 10100_2
$01^{1}0^{1}11_2$
$+ \quad 1$
01100_2
$= 11\emptyset\emptyset_2$ ↙ 2's complement

* Review ADDING Binary #'s *

One's Complement:

∅	∅∅∅
1	∅∅1
2	∅1∅
3	∅11
-3	1∅∅
-2	1∅1
-1	11∅
-∅	111

1. GET One's complement TO FIND The (-) of a #.

ex) 000 = ∅ ∅∅1 = 1 ∅1∅ = 2
111 = -∅ 11∅ = -1 1∅1 = -2 ...

Observations:

1. all (-) #'s start w/ '1'
2. There are 2 codes for ∅
   → wasteful...?
3. TOTal combos - 2^n, where n is # of bits
4. (+) Decimal range: ∅ → 2^{n-1}
5. (-) Decimal range: -∅ → -2^{n-2}

** keep in MIND

In the above examples we are working in a 3 bit system!

ex) convert 111_2 to Decimal, in a 4 bit sys

$\emptyset 111_2$ (4 2 1)

leading ∅: positive $= \boxed{7_{10}}$

1. CHECK IF leading 1 or ∅
   1 - # is (-)
   ∅ - # is (+)
2. carry out 1's complement only IF the # is (-)
   → bc we only complement to get (-) #s!
3. convert to Decimal.

ex) convert 1011_2 to decimal in 4 bit binary system

1011_2

leading 1 ↓

so... 0100_2

4

$\Rightarrow 4_{10} \rightarrow \boxed{-4_{10}}$

2's Complement System:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

OBSERVATION

→ NO (-) 0
  * VERY USEFUL!

→ also 2's compliment just works better with hardware.

use 2's compliment to convert

-42_{10} to binary (in an 8 bit system)

42:
$-32 \rightarrow 2^5$
10
$-8 \rightarrow 2^3$
2 $\rightarrow 2^1$
-2
0

00101010_2

11010101_2

$+ \quad 1$

11010110_2

1. BIC # is (-), we have to complement

→ converting from binary to decimal:

11010110_2 in 8 bit 2's complement system.

1. BIC starts w/ 1 → (-) #, some have to complement it!

$00101001_2 + 1 \Rightarrow 101010$

2. now convert back to decimal

$32 + 8 + 2 = 42_{10}$

$\boxed{-42_{10}}$