Number Representations

There are 10 types of people in this world
Those who understand binary and those who don’t

Agenda

- Bits, Bytes, and Words
- Number bases and base conversion
  - Positional notation
- Binary arithmetic and data representation
  - Signed numbers
  - Arithmetic and overflow
  - Packed Decimal, ASCII, Parity…

From Lecture 1: Below Your Program

- High-level language program (in C)

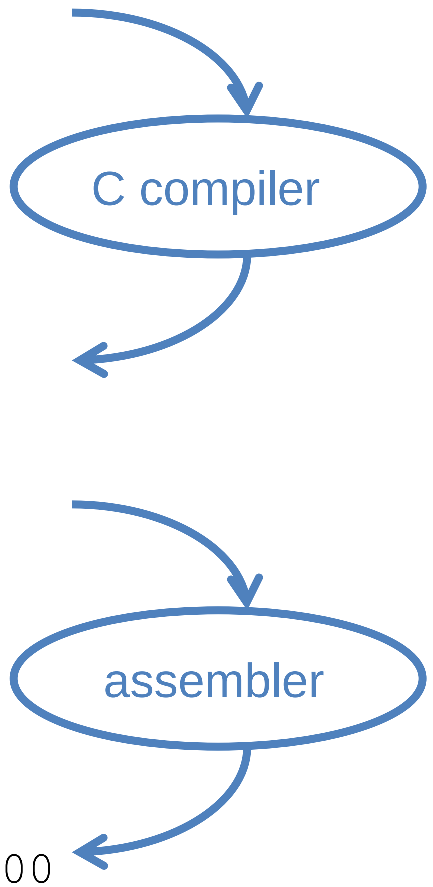
```
swap (int v[], int k) {
    int temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

- Assembly language program (for MIPS)

```
swap:  sll     $2, $5, 2
       add     $2, $4,$2
       lw      $15, 0($2)
       lw      $16, 4($2)
       sw      $16, 0($2)
       sw      $15, 4($2)
       jr      $31
```

- Machine (object) code (for MIPS)

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
 . . .
```



How do people and computers represent numbers?

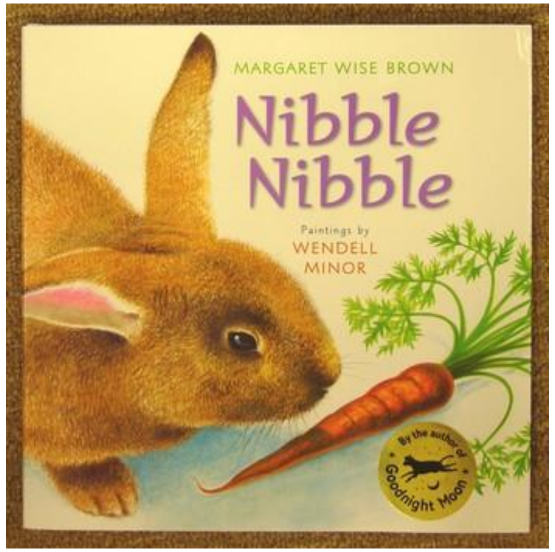
Why Base 10? Why Base 2?

- **Decimal: Base 10**, a single number from 0-9
- **Binary: Base 2**, a single digit is called a ***bit*** (binary digit)
- A bit is the smallest unit of information, and can represent…
  - 1 / 0
  - True / False
  - Yes / No
  - On / Off
  - used in a two-state (Boolean) logic

- Can represent anything with a sequence of binary bits: numbers, letters, words, the image of this pillow, programs, etc.

Nibbles to Words

- Typically store information in groups
  - a **byte** is a group of 8 bits
    - e.g. 01100101
  - a **nibble/nybble** (a small bite) is a group of 4 bits,
    - e.g. 0110
  - a **word** (MIPS) is a group of 4 bytes, or 32 bits
    - e.g. 01100101011001010110010101100101
- Least significant bit right most



Numbers and Positional Notation

- a number with n digit.

$a_{n-1} \dots a_1\ a_0$

- The integer value in base b:

$N= a_{n-1}\ b^{n-1}+ \dots\dots +a_1 b^1 +a_0 b^0$

Examples

- 238_{10}

$$\begin{aligned} 238 &= 2 * 10^{2} + 3 * 10^{1} + 8 * 10^{0} \\ &= 200 + 30 + 8 \end{aligned}$$

- 10110_{2}

$$\begin{aligned} 11010 &= 1 * 2^{4} + 1 * 2^{3} + 0 * 2^{2} + 1 * 2^{1} + 0 * 2^{0} \\ &= 16 + 8 + 2 \\ &= 26 \end{aligned}$$

Common Bases

Name of Base	Base	Digits used
Decimal	10	0,1,2,3,4,5,6,7,8,9
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- We often write hex numbers preceded by 0x

  $1011_2 = 11_{10} = 13_8 = B_{16} = 0xB$

How many bits are needed to represent a decimal number?

- What is the largest decimal number with **n** digits?

  $10^n - 1$, why?

- What is the largest binary number with **m** digits?

  $2^m - 1$, why?

- For base **b**, the largest number is $b^k - 1$ for **k** digit number.

How many bits are needed to represent a decimal number?

- How many digits necessary for numbers up to one million?

$$1{,}000{,}000 <= 10^n - 1$$
$$1{,}000{,}001 <= 10^n$$
$$n >= \log_{10}(1{,}000{,}001) = 6.0000004$$

- What about storing one million binary?

How to Convert from one Base to Another?

Base Conversion – Decimal to Another Base

- Reverse positional notation :
  - Divide by **b**
  - Remainder of result is least significant bit
  - Repeat with the quotient.

Example: 5_{10} = 5/2= 2R1 ---> Take out reminder 1

= 2/2= 1R0 --> Take out reminder 0

= 1/2= 0R1 --> Take out reminder 1

Stop since value is 0. Result is backward

= 101_2

Base Conversion – Decimal to Another Base

- Reverse positional notation, divide by base, 16 and keep reminder.
- Example: What is 562_{10} in hexadecimal?

562_{10} = 562/16= 35R2 --> Take out 2

= 35/16 = 2R3 --> Take out 3

= 2/ 16 = 0R2 --> Take out 2

= 0X232

Base Conversion – Other Base to Decimal

- Basic Approach: Direct expansion with positional weights
  $N = \mathbf{a}_n b^n + \mathbf{a}_{n-1} b^{n-1} + \ldots\ldots + \mathbf{a}_1 b + \mathbf{a}_0$

Base Conversion – Other Base to Decimal

- Examples: What is 1010101_2 in Decimal

$1010101_2 = 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

$= 1*64 + 0*32 + 1*16 + 0*8 + 1*4 + 0*2 + 1*1 \;= 85_{10}$

Examples: What is $1AB_{16}$ in Decimal?

$1AB_{16}=1* 16^2+ A*16^1 + B* 16^0$

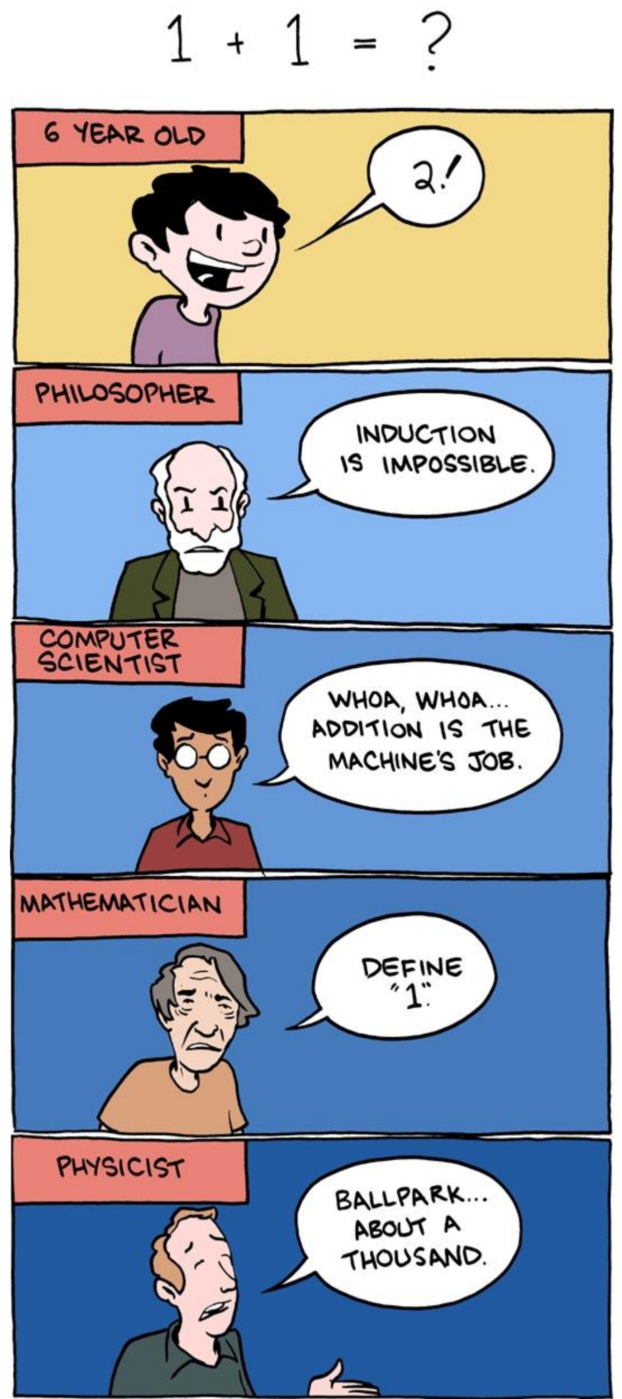
$= 1* 256+ A*16 + B$

$= 256 + 160 + 11 \;= 427_{10}$

Base Conversion – Base A to Base B

- Conversion from base a to base b
  - First convert base a to decimal
  - Then convert decimal to base b
- Special cases (easier by grouping)
  - Binary to hexadecimal and back
    - Example:
      11010101101_2
      $= 011010101101_2$
      $= 6AD_{16}$
  - Binary to octal and back
    - Example: 760_8 becomes 111110000_2

Dec	Hex	Bin
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



Questions

- What is the largest binary number with n bits?
- How do we add binary numbers?
- How do we subtract binary numbers?
- Why do programmers always mix up Halloween and Christmas?

  "Because Oct 31 (31_8) = Dec 25 (25_{10})."
- How should we represent negative numbers?

How to Represent Signed Numbers?

Sign-and-Magnitude

- The first approach
- Use the most significant bit to represent the sign

  +13= 0000 1101

  -13 = 1000 1101
- Problems
  - Two representations for zero: 0000 0000 and 1000 0000.
  - Cannot add a positive number and a negative number together.

One's Complement

- ***Invert each bit!***

  +13= 0000 1101

  -13 = 1111 0010

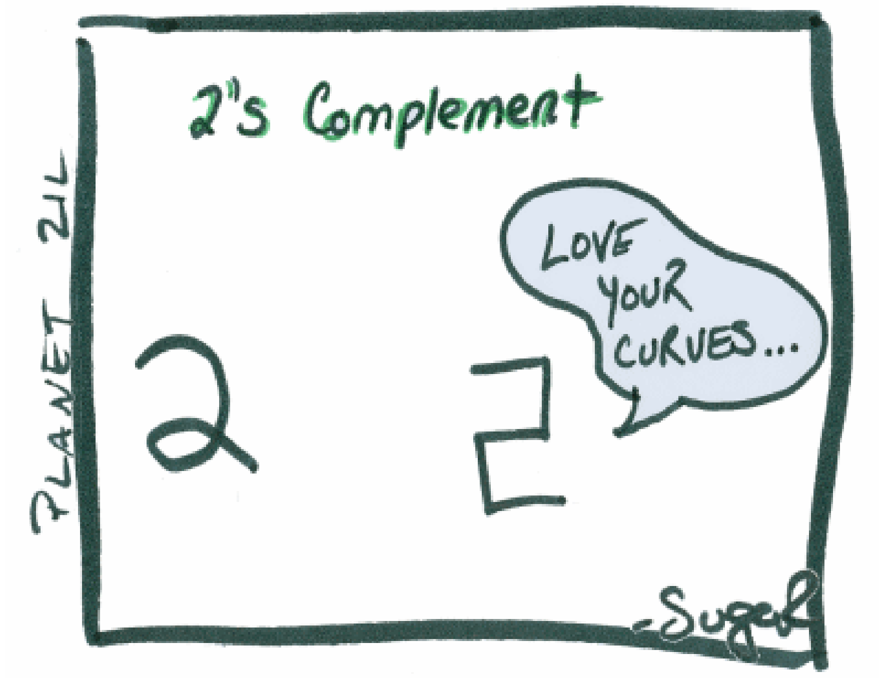
- ***Problems:***
  - *Still two representations for zero* 0000 0000 and 1111 1111
  - *Answer is off by 1*
  - Incorrect overflow
    - What is 16 + (-13)?

Two's Complement

- **The gold standard**
- **Invert the bits and add one!**
  +13 = 0000 1101
  What is –13?

- Unique zero
  0000 0000

- MSB represents the sign.
  - 1 if negative
  - 0 if positive
- -1 becomes: 0000 0001=1111 1110=1111 1111



Two's Complement

Example

Decimal	Binary
0	000
1	001
2	010
3	011
-4	100
-3	101
-2	110
-1	111

- Easily implemented in hardware
- Range from -2^{n-1} to $+2^{n-1} - 1$
- Negative numbers are defined as
  $-N = B^n - N$
- The complement of complement of Y is Y, i.e., -(-Y) = Y.

Two's Complement

- Why does inverting bits and adding one work?

  $x + \overline{x} = -1$ (i.e., it is all ones)

  $-x = \overline{x} + 1$ (basic algebra)

Where $\overline{x}$ is the one's complement of x

Two's Complement

- 16 - 13 = ?

  = 16 + (-13)

```
   0001 0000
   1111 0011
   =========
 1 0000 0011
```

Binary Arithmetic

Addition	Subtraction	Multiplication
0 + 0 = 0	0 - 0 = 0	0 * 0 = 0
0 + 1 = 1	0 - 1 = 1 borrow 1	0 * 1 = 0
1 + 0 = 1	1 - 0 = 1	1 * 0 = 0
1 + 1 = 0 carry 1	1 - 1 = 0	1 * 1 = 1

- Rules in base 10 are also valid in any other base
- Subtraction often done using addition and 2's complement
- Multiplication and division are similar. We will learn in a few lectures

Arithmetic overflow

- Typically use a fixed # of bits to represent numbers!
- *Arithmetic overflow* can occur during two's complement addition/subtraction

Arithmetic overflow

- Example: $6_{10} + 5_{10}$ using 4 bits

Example in 4 bits:

$6_{10} = 0110_2$
$5_{10} = 0101_2$

```
  0110
+0101
--------
  1011
```

11_{10} if interpreted as an unsigned number
-5_{10} if interpreted as a two's complement number (signed) **(But the result is actually 11_{10} as we are summing two positive numbers --> Carry change the sign bit)**

Arithmetic overflow

- What do you notice:
  - Carry changes the sign bit
    - Add 2 positive numbers and get a negative result
    - Add 2 negative numbers and get a positive result.
    - A minus B with A<0 and B>0 and getting positive result.
    - A minus B with A>0 and B<0 and getting negative result'
  - Carry goes beyond the boundary of the size (unsigned numbers)
    - (the answer is smaller than it should have been since we "lost" a bit)

  - Need to take extra care when implement it on the circuits

Packed Decimal
(Binary Coded Decimal, BCD)

- Replace each digit with its 4-bit equivalent

  5372_{10} in BCD is 0101 0011 0111 0010
- Good
  - User friendly? Yes!
  - BCD is easier for humans to parse
- Bad
  - Wastes storage space
  - BCD is harder to implement in hardware

Parity

- Used to check for corrupt data in storage or transmission, with two kinds: even and odd
  - Total # of bits with 1 must be even for even parity
  - Total # of bits with 1 must be odd for odd parity
- Examples:

Even Parity
0010101001
0000000000
1010101011

Odd Parity
1110101000
0011110111
0000000001

- Advantage of odd parity? Detects if a line goes dead (000000000<u>0</u>)
- Detecting multiple errors? Correcting errors?

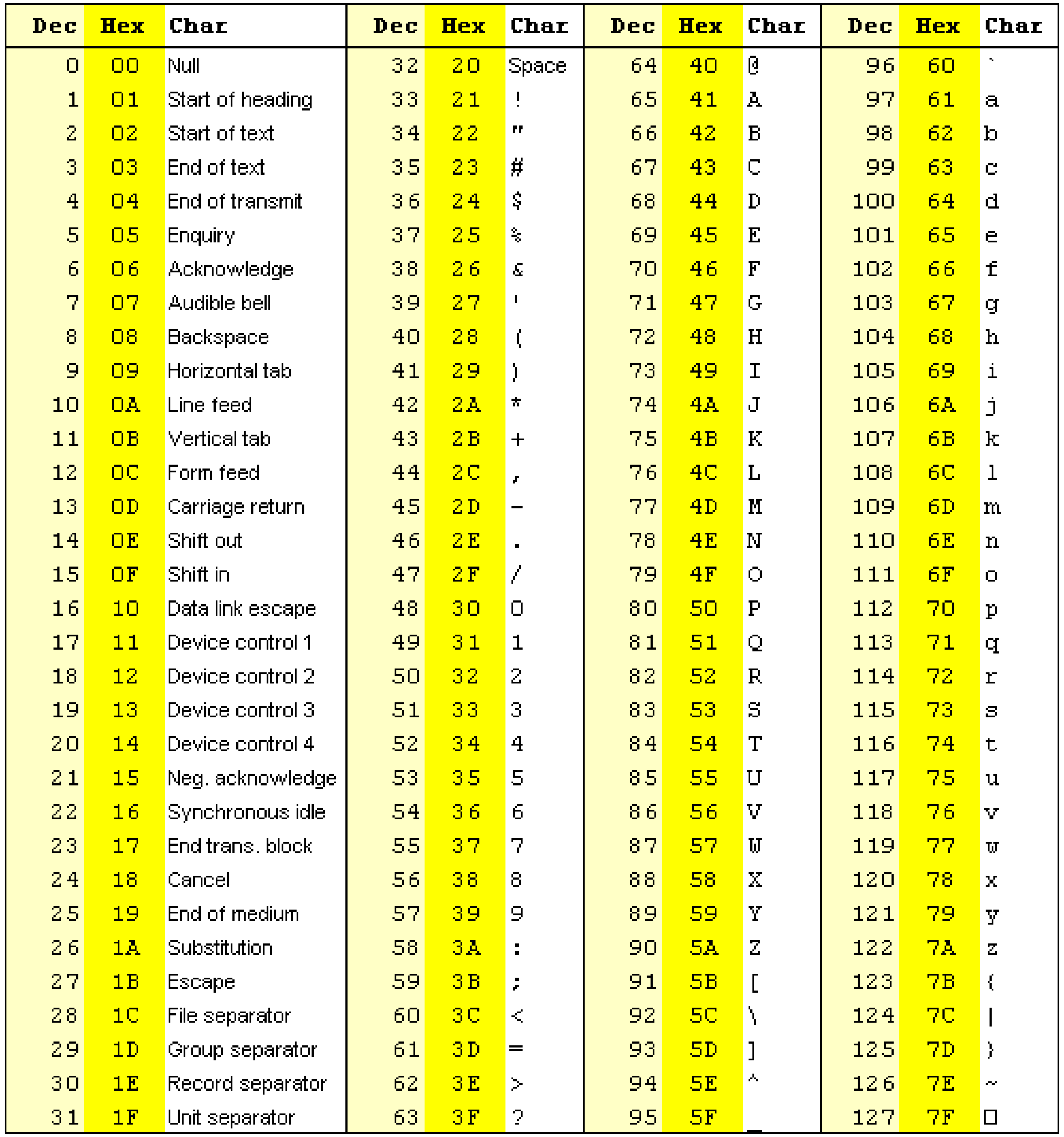
How to Represent Characters?

Character Data Codes

Name	bits per symbol	# of symbols	Comments
IBM-BCD	6	64	Capital letters: A-Z, 0-9, $, etc. Not to confuse with Packed Decimal.
ASCII	7	128	All letters: a-z, A-Z, 0-9, $, BEL, TAB, etc. See link below.
USACII	8	256	Includes parity (even).
EBCDIC	8	256	On IBM mainframes (odd parity)
UNICODE	16	65,536	Can represent the letters of all languages.

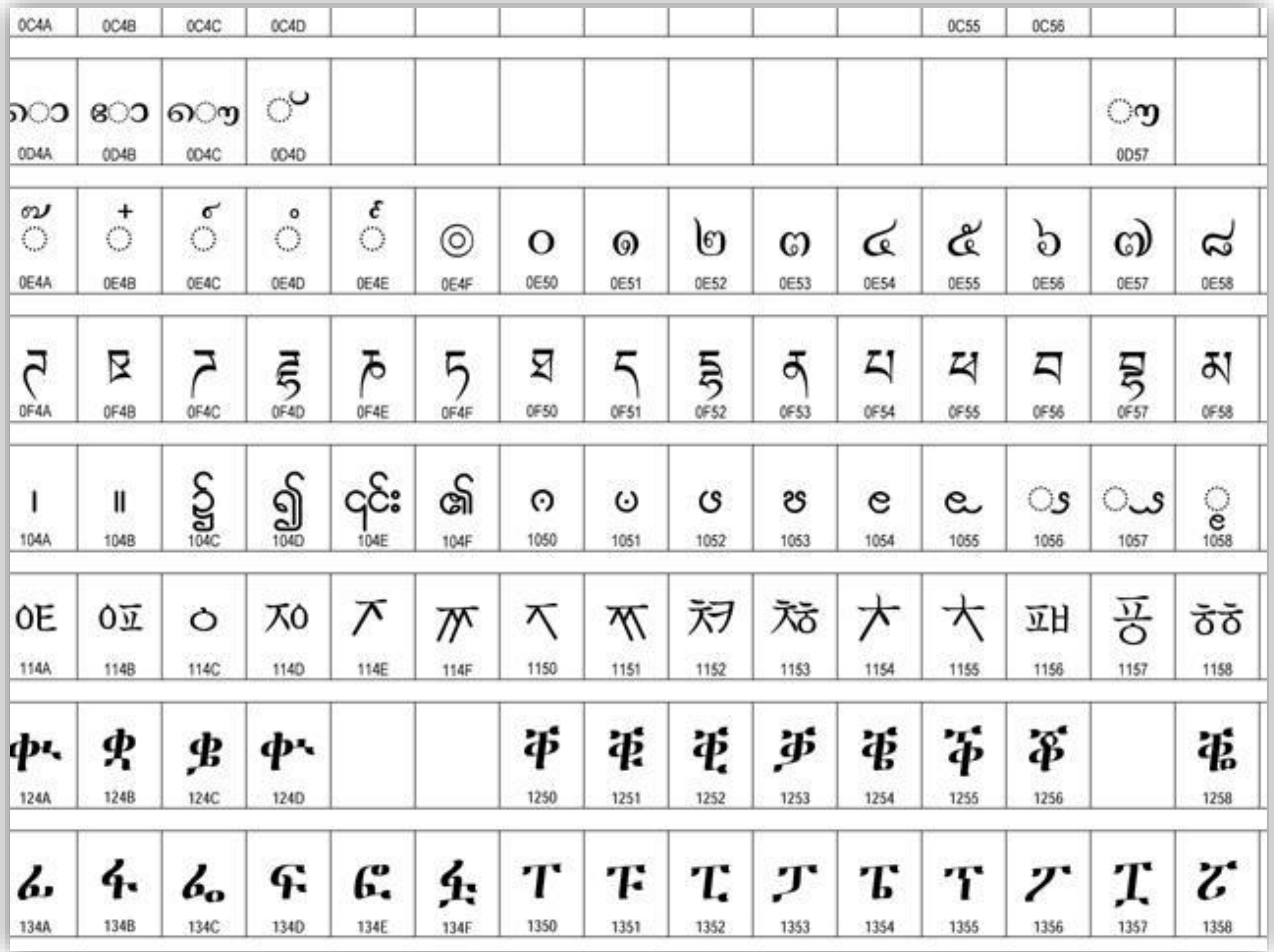
ASCII

American
Standard
Code for
Information
Interchange



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

UNICODE

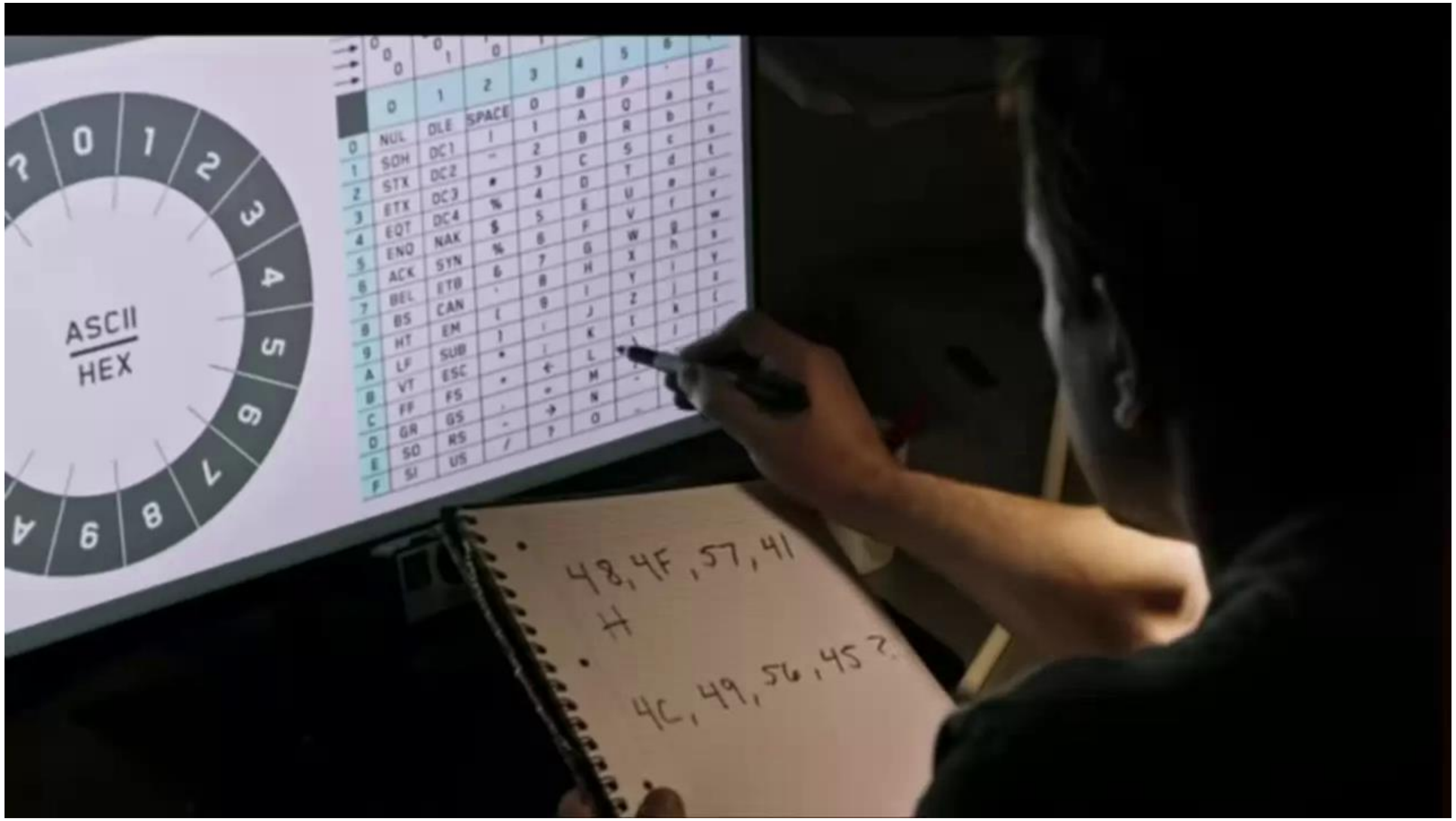


0C4A	0C4B	0C4C	0C4D								0C55	0C56		
◌ൊ 0D4A	◌ോ 0D4B	◌ൌ 0D4C	◌് 0D4D										◌ൗ 0D57	
◌๊ 0E4A	◌๋ 0E4B	◌์ 0E4C	◌ํ 0E4D	◌๎ 0E4E	๏ 0E4F	๐ 0E50	๑ 0E51	๒ 0E52	๓ 0E53	๔ 0E54	๕ 0E55	๖ 0E56	๗ 0E57	๘ 0E58
ཊ 0F4A	ཋ 0F4B	ཌ 0F4C	ཌྷ 0F4D	ཎ 0F4E	ཏ 0F4F	ཐ 0F50	ད 0F51	དྷ 0F52	ན 0F53	པ 0F54	ཕ 0F55	བ 0F56	བྷ 0F57	མ 0F58
၊ 104A	။ 104B	၌ 104C	၍ 104D	၎ 104E	၏ 104F	ၐ 1050	ၑ 1051	ၒ 1052	ၓ 1053	ၔ 1054	ၕ 1055	◌ၖ 1056	◌ၗ 1057	◌ၘ 1058
ᅊ 114A	ᅋ 114B	ᅌ 114C	ᅍ 114D	ᅎ 114E	ᅏ 114F	ᅐ 1150	ᅑ 1151	ᅒ 1152	ᅓ 1153	ᅔ 1154	ᅕ 1155	ᅖ 1156	ᅗ 1157	ᅘ 1158
ቊ 124A	ቋ 124B	ቌ 124C	ቍ 124D			ቐ 1250	ቑ 1251	ቒ 1252	ቓ 1253	ቔ 1254	ቕ 1255	ቖ 1256		ቘ 1258
ፊ 134A	ፋ 134B	ፌ 134C	ፍ 134D	ፎ 134E	ፏ 134F	ፐ 1350	ፑ 1351	ፒ 1352	ፓ 1353	ፔ 1354	ፕ 1355	ፖ 1356	ፗ 1357	ፘ 1358

The Martian

An astronaut is stranded on Mars.
The communication devices are broken.
Only one camera is working but no sound.
How can he communicate with the Earth?

48 4F 57 41 4C 49 56 45?
What was the Earth trying to say?



Answer to class question: It takes about **5 to 20 minutes** for a radio signal to travel the distance between Mars and Earth, depending on planet positions. https://mars.nasa.gov/mars2020/spacecraft/rover/communications/

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

48 4F 57 41
4C 49 56 45?

Summary

- Definitions: Bits, Nibbles, Bytes, Words
- Representations: number bases, conversion
- Signed numbers with 2’s complement
- Other data representation
  - Packed Decimal (BCD)
  - ASCII and other character data codes
  - Parity

Review and more information

- Textbook
  - Section 2.4, Signed and Unsigned Numbers

There are 10 types of people in this world…
Those who understand binary and those who don’t