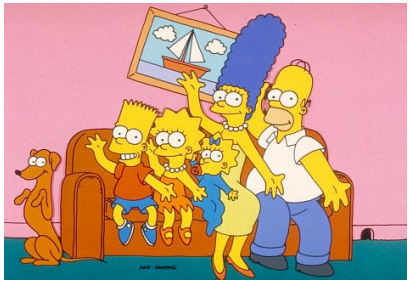
Data Representation

External vs. Internal Representation

- External representation:
  - Convenient for programmer
  - Decimal (base 10)



- Internal representation:
  - Actual representation of data in the computer's memory: Always binary (1's and 0's)

Binary representation (base 2)

- On a computer all data is stored in binary
- Only two symbols: 0 and 1
- Each position is called a *bit*
- *Bits take up space*
- 8 bits make a *byte*
- *Example of a 4-bit number*

- Actually the data is voltages
- We use the abstraction:
  - High voltage: 1 (true)
  - Low voltage: 0 (false)

Represent 15_{10} in binary using positional encoding

1024	0	0	0	0	0	0	1	0	1	(Decimal 5)
512	256	128	64	32	16	8	4	2^1	2^0	(1's place

Represent 15_{10} in binary using positional encoding

$$1111$$

Positional encoding for non-negative numbers

- Each position represents some power of the base
- Decimal (Base 10), Digits (0-9)
- Binary (Base 2), Digits (0,1)
- Hex (Base 16), Digits (0-9, A-F)

flex

X means this is a hex representation

16^3 256 16 16^0

A 0

0XA0 → 10×16 = 160_{10} 1

101_5 = ? In decimal

Using positional encoding

A. 26

B. 51

C. 126

D. 130

$1 \quad 0 \quad 1$

$(25)5^2 \quad 5^1 \quad 1$

Base 5

Converting between binary and decimal

int x = 22;
int x = 0b10110
↳ binary rep.

Binary to decimal: $1\,0\,1\,1\,0_2 = ?_{10}$

16 8 4 2 1

1*16 + 1*4 + 1*2 => 22

Decimal to binary: $34_{10} = ?_2$

10

1 0 0 0 1 0

64 32 16 8 4 2 2^0

Binary to hex

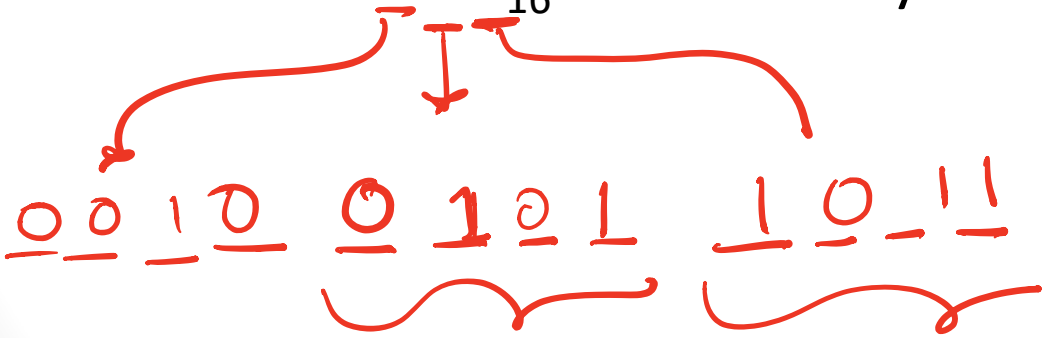
0001 0110 (Binary)

int x = 0X 1 6; (Hex)

Hex to binary

- Each hex digit corresponds directly to four binary digits
- Programmers love hex, why?

$25B_{16}$ = ? In binary



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

Hexadecimal to decimal

$25B_{16}$ = ? Decimal

256 16 1

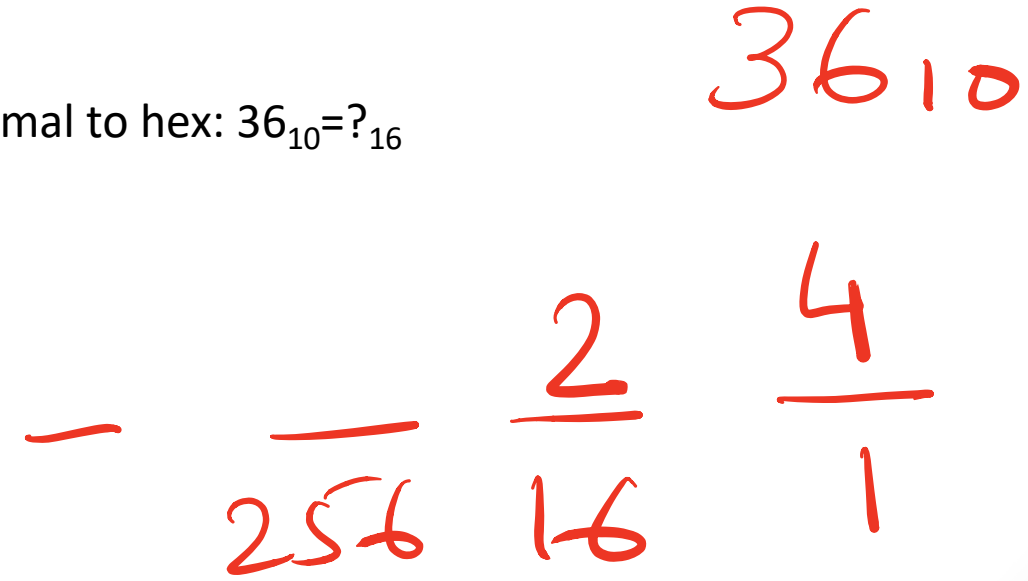
$2 * 256 + 5 * 16 + 11 * 1$

=

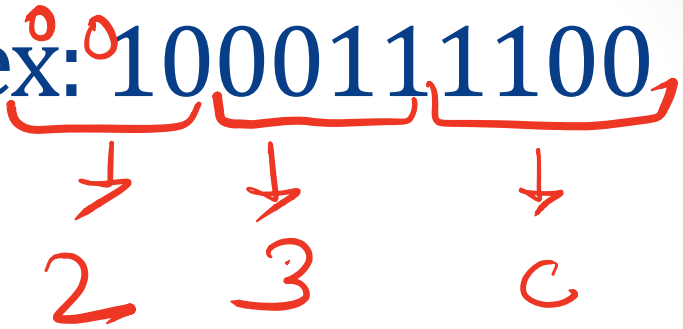
Hexadecimal to decimal

- Use polynomial expansion
- $25B_{16} = 2*256 + 5*16 + 11*1 = 512 + 80 + 11$

  $= 603$
- Decimal to hex: $36_{10}=?_{16}$



Binary to hex: 1000111100

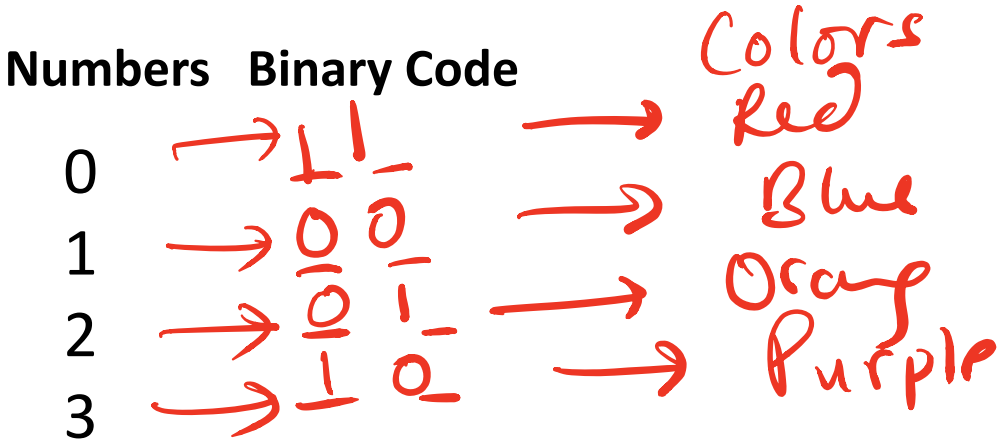


A. 8F0

B. 23C

C. None of the above

BIG IDEA: Bits can represent anything!!



How many (minimum) bits are required to represent the numbers 0 to 3?

What is the maximum positive value that can be stored in a byte?

A. 127

B. 128

C. 255

D. 256

BIG IDEA: Bits can represent anything!!

Colors	Binary code
	
	
	

How many (minimum) bits are required to represent the three colors?

BIG IDEA: Bits can represent anything!!

Characters

‘a’

‘b’

‘c’

‘d’

‘e’

N bits can represent at most 2^N things

BIG IDEA: Bits can represent anything!!

- Logical values?
  - 0 ⇒ False, 1 ⇒ True
- colors ?

- Characters?
  - 26 letters ⇒ 5 bits ($2^5 = 32$)
  - upper/lower case + punctuation
    ⇒ 7 bits (in 8) ("ASCII")
  - standard code to cover all the world's languages ⇒ 8,16,32 bits ("Unicode")
    www.unicode.com
- locations / addresses? commands?

MEMORIZE: N bits ⇔ at most 2^N things

Ascii Encoding

`char x= 'a';` Stores the ascii value of 'a' (97)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ASCII Table

Unicode

- Universal character encoding (extends ASCII to handle other languages)

```
>>> chinese = '\u4e16\u754c\u60a8\u597d!'
>>> print(chinese)
世界您好!
```

- Includes all ASCII characters using the same ascii encoding

```
>>> print('\u0048\u0049')
HI
>>>
```