Basic and Complex Objects in C

David E. Culler

CS61CL

Feb 4, 2009

Lecture 2

Computers manipulate finite representations of things

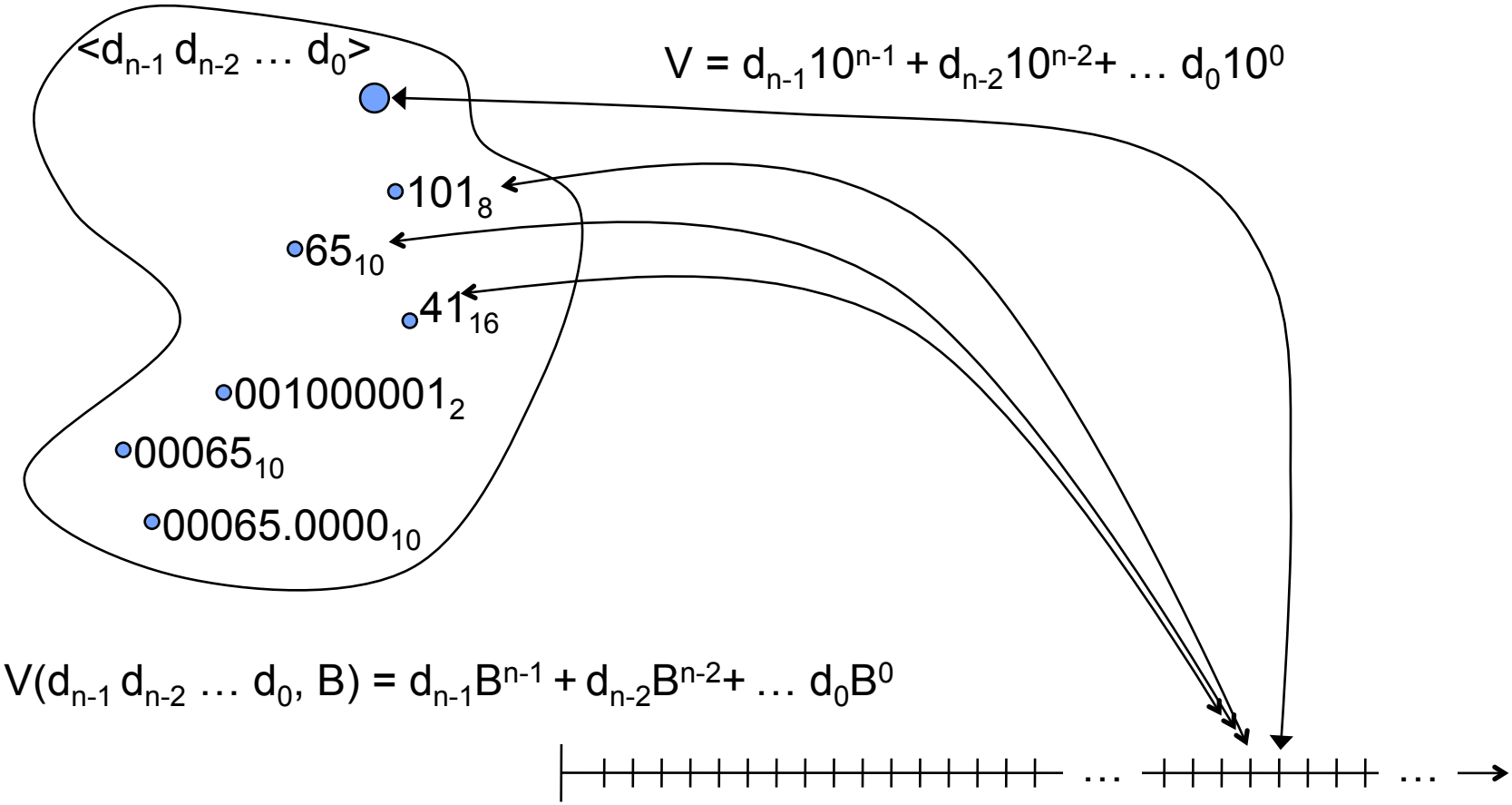
Basic Data Types in C

- char – 1 byte
  - Sufficient to represent the local character set
  - Typically ASCII
- int – signed integer values
  - Range dictated by natural word width of the machine
- float – single precision floating point number
  - A lot like real numbers
  - Specific representation defined by IEEE
- double – double precision floating point numb er
  - Even more like real numbers…

What is this?

- 101
- 0101

Representation ⇔ Meaning



- **Objects are represented as collections of symbols**
- **Their meaning is derived from what you do with them.**

Another mapping function

Regular ASCII Chart (character codes 0 - 127)

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h	sp	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	`	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah		(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch	♀	(np)	028d	1Ch	∟	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh		(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255; Codepage 850)

128d	80h	Ç	144d	90h	É	160d	A0h	á	176d	B0h	░	192d	C0h	└	208d	D0h	╨	224d	E0h	Ó	240d	F0h	-
129d	81h	ü	145d	91h	æ	161d	A1h	í	177d	B1h	▒	193d	C1h	┴	209d	D1h	╤	225d	E1h	ß	241d	F1h	±
130d	82h	é	146d	92h	Æ	162d	A2h	ó	178d	B2h	▓	194d	C2h	┬	210d	D2h	╥	226d	E2h	Ô	242d	F2h	‗
131d	83h	â	147d	93h	ô	163d	A3h	ú	179d	B3h	│	195d	C3h	├	211d	D3h	╙	227d	E3h	Ò	243d	F3h	¾
132d	84h	ä	148d	94h	ö	164d	A4h	ñ	180d	B4h	┤	196d	C4h	─	212d	D4h	╘	228d	E4h	õ	244d	F4h	¶
133d	85h	à	149d	95h	ò	165d	A5h	Ñ	181d	B5h	╡	197d	C5h	┼	213d	D5h	╒	229d	E5h	Õ	245d	F5h	§
134d	86h	å	150d	96h	û	166d	A6h	ª	182d	B6h	╢	198d	C6h	╞	214d	D6h	╓	230d	E6h	µ	246d	F6h	÷
135d	87h	ç	151d	97h	ù	167d	A7h	º	183d	B7h	╖	199d	C7h	╟	215d	D7h	╫	231d	E7h	þ	247d	F7h	¸
136d	88h	ê	152d	98h	ÿ	168d	A8h	¿	184d	B8h	╕	200d	C8h	╚	216d	D8h	╪	232d	E8h	Þ	248d	F8h	°
137d	89h	ë	153d	99h	Ö	169d	A9h	®	185d	B9h	╣	201d	C9h	╔	217d	D9h	┘	233d	E9h	Ú	249d	F9h	¨
138d	8Ah	è	154d	9Ah	Ü	170d	AAh	¬	186d	BAh	║	202d	CAh	╩	218d	DAh	┌	234d	EAh	Û	250d	FAh	·
139d	8Bh	ï	155d	9Bh	ø	171d	ABh	½	187d	BBh	╗	203d	CBh	╦	219d	DBh	█	235d	EBh	Ù	251d	FBh	¹
140d	8Ch	î	156d	9Ch	£	172d	ACh	¼	188d	BCh	╝	204d	CCh	╠	220d	DCh	▄	236d	ECh	ý	252d	FCh	²
141d	8Dh	ì	157d	9Dh	Ø	173d	ADh	¡	189d	BDh	¢	205d	CDh	═	221d	DDh	¦	237d	EDh	Ý	253d	FDh	³
142d	8Eh	Ä	158d	9Eh	×	174d	AEh	«	190d	BEh	¥	206d	CEh	╬	222d	DEh	Ì	238d	EEh	¯	254d	FEh	■
143d	8Fh	Å	159d	9Fh	ƒ	175d	AFh	»	191d	BFh	┐	207d	CFh	¤	223d	DFh	▀	239d	EFh	´	255d	FFh	

Hexadecimal to 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

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

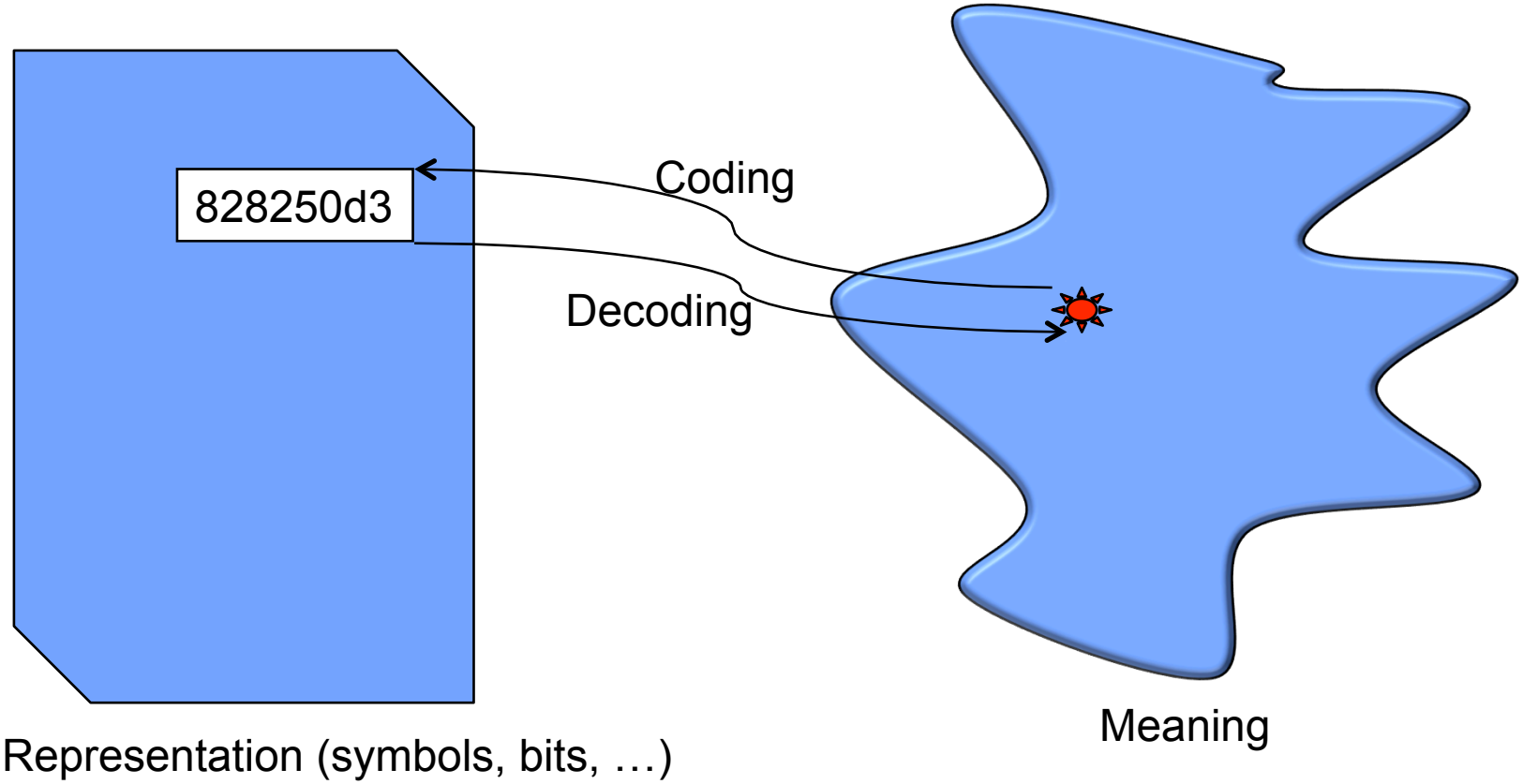
Michael Goerz, Sept 04 2000

What's this?

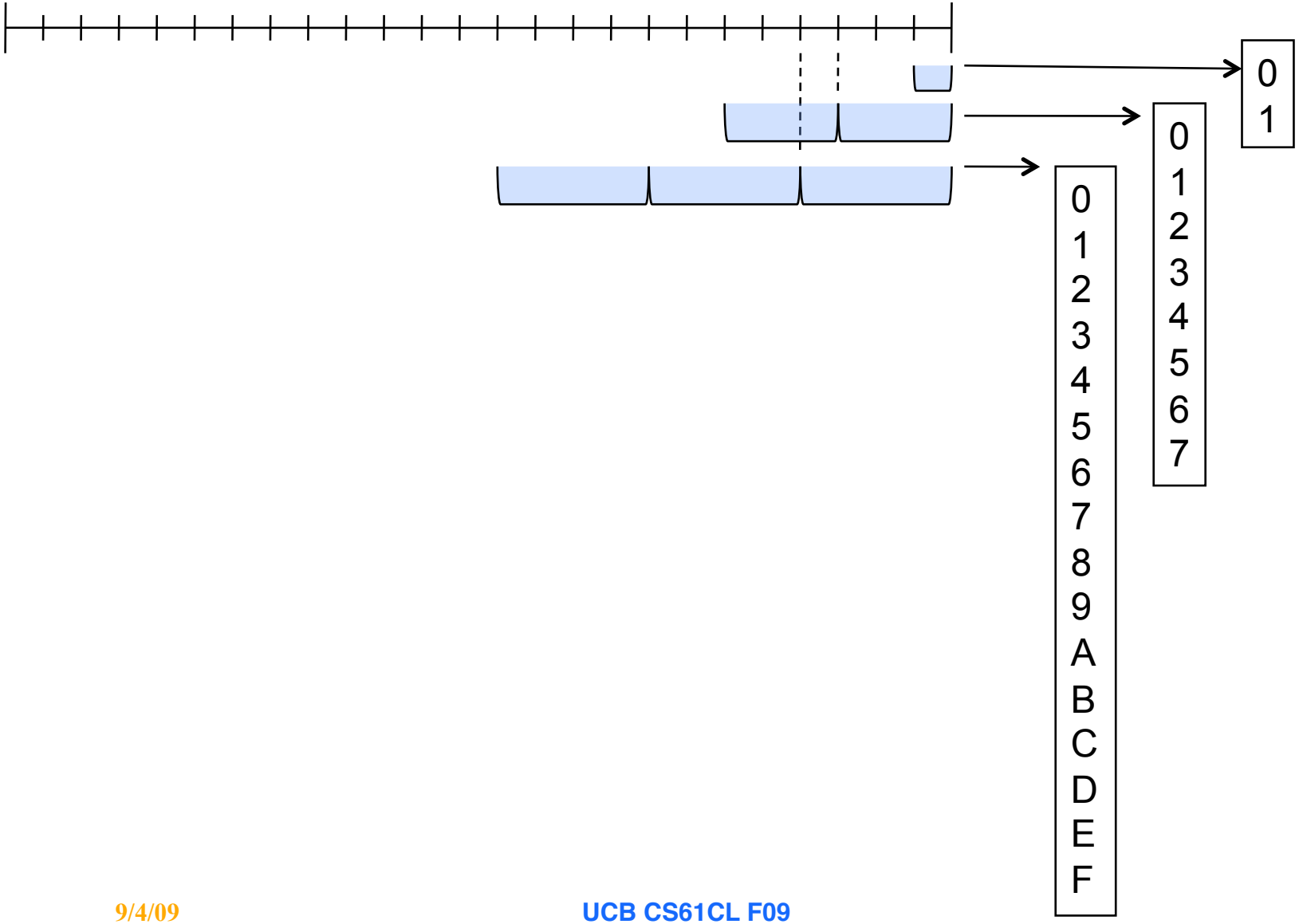
And this?

And this?

Coding / Decoding



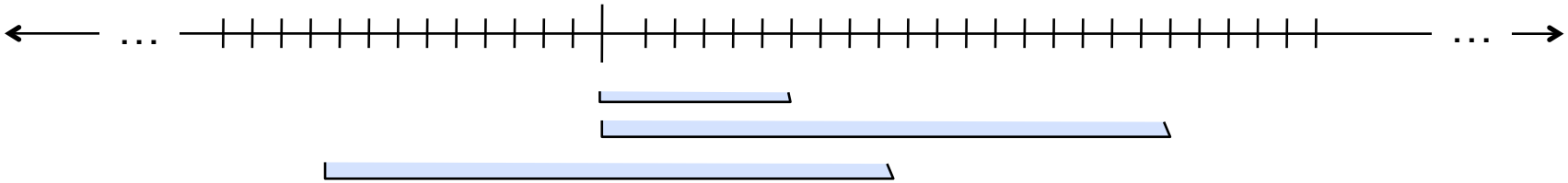
Trivial Coding/Decodng



The Number of Numbers (Symbols)

- Finite Representations
- N digits in base 10 can represent 10^N values
  - 000, 001, …, 999
- N digits in base 2 can represent 2^N values
  - 0000, 0001, … , 1111
- N digits in base B can represent B^N values
- N symbols in an alphabet of k characters can represent k^N distinct "words"

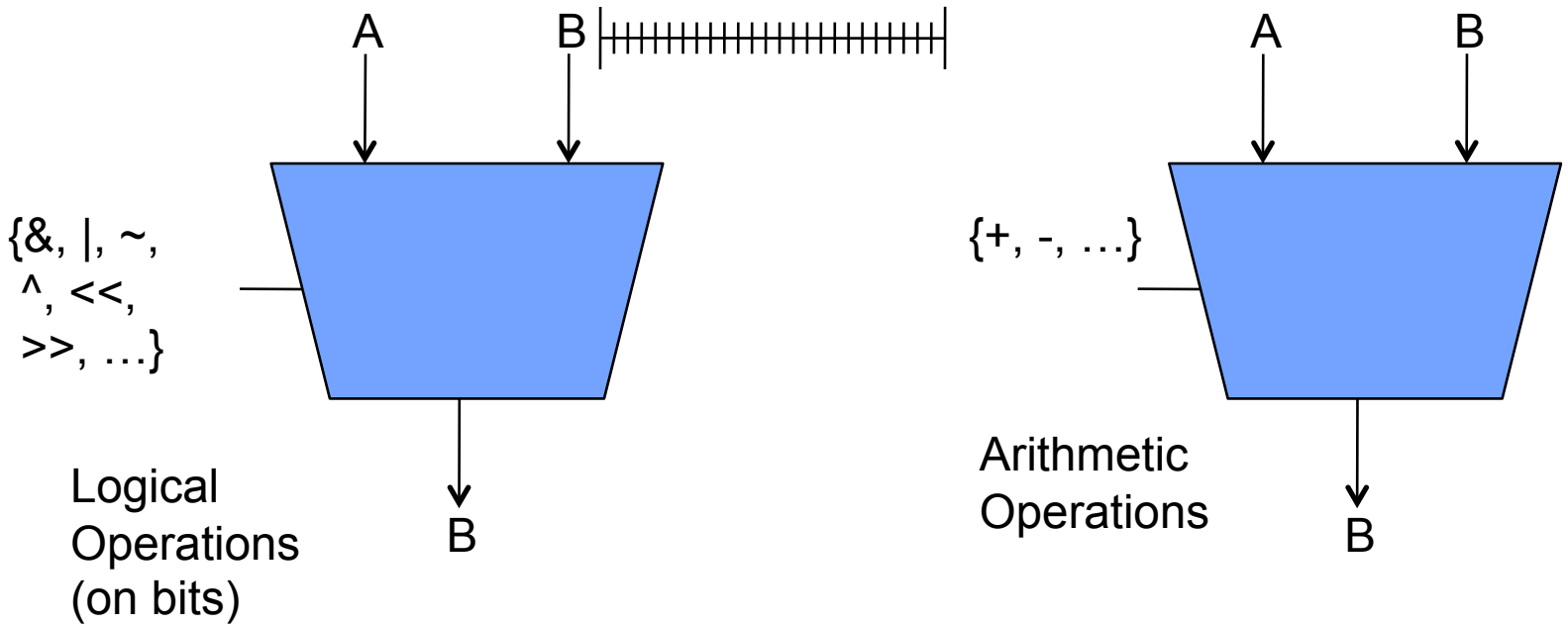
Which ones?



- **char : 1 byte, 0..255**
- **unsigned int**
  - **32 bits => 0 .. $2^{32} - 1$ (about 4 billion, $2^{10} \sim 10^3$)**
  - **64 bits => 0 .. $2^{64} - 1$ (about 16 quintillion)**
- **int**
  - **32 bits => -2^{31}…0... $2^{31} - 1$ (about -2 billion to +billion)**
  - **64 bits => -2^{63}…0 .. $2^{63} - 1$**
  - **long long int x;**

Operations (on the representation)

- **Computers provide direct hardware support for manipulating certain basic objects**
- ***Word* of bits**



$a + 0 = a$

$a - a = 0$

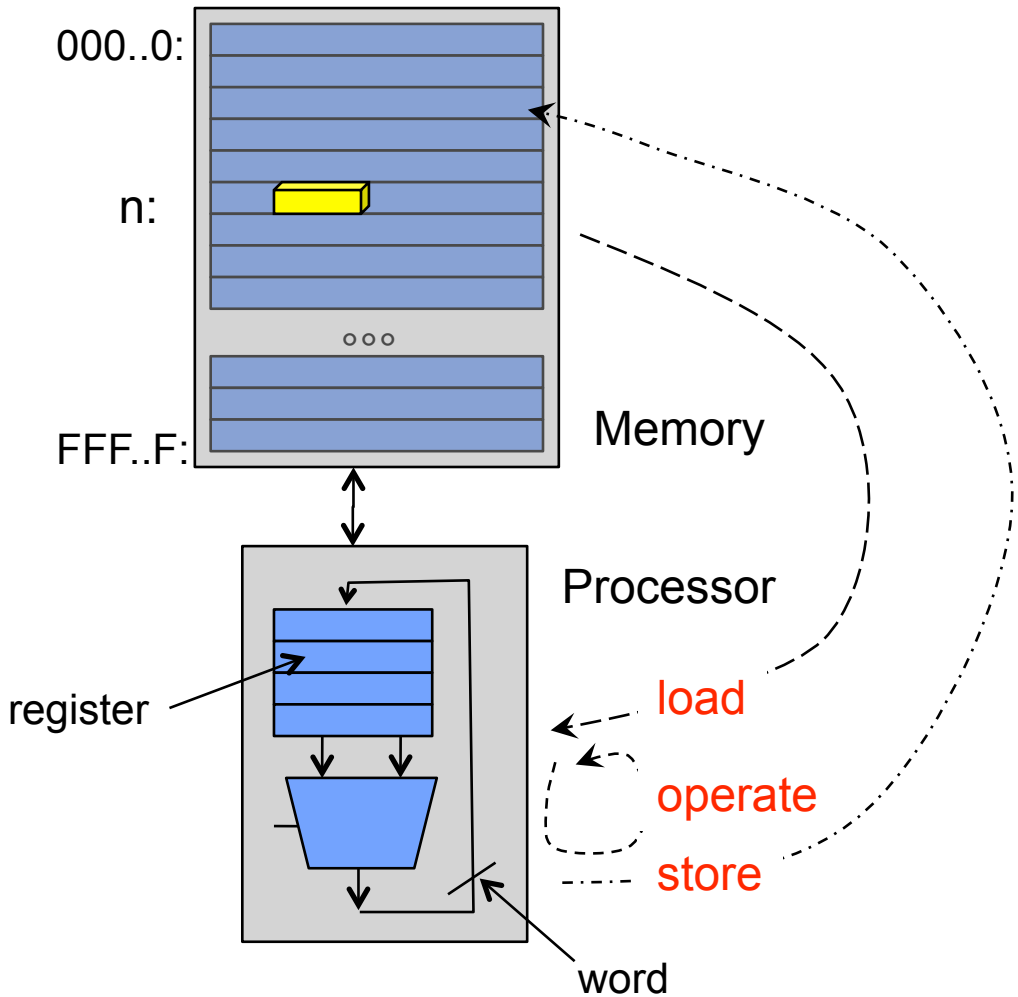
$a + b = b + a$

$a \neq b \Leftrightarrow a - b \neq 0$ (???)

Manipulating bits

- In C we can work with what the object represents
  - x = x + y;
  - Z = x * 2;
  - Q = x / 2;
  - putchar(c);
  - If (A && !B) { … }
- Or with its underlying representation
  - Z = x << 1;
  - Q = x >> 1;
  - D = x & 1;
  - E = x & 0xFFFFFFFE;
  - G = x & ~1
  - M = (y & 0x74) >> 3

Where do Objects live and work?

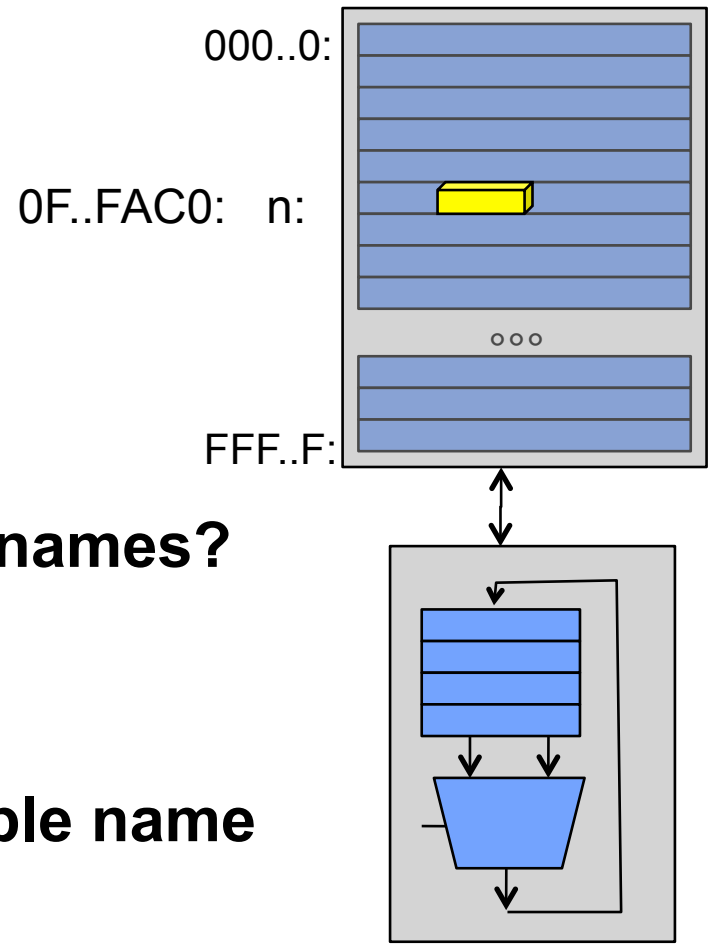


Finite Representation

- When the limitations of the representation are exceeded,
- the illusion that the “thing” and “its representation” the same
- breaks down
  - x = x << 32;
  - x = x * x;

Name versus address

- The White House … vs…
- 1600 Pennsylvania Avenue NW
  Washington, DC 20500



- berkeley.edu .vs. 169.229.131.81

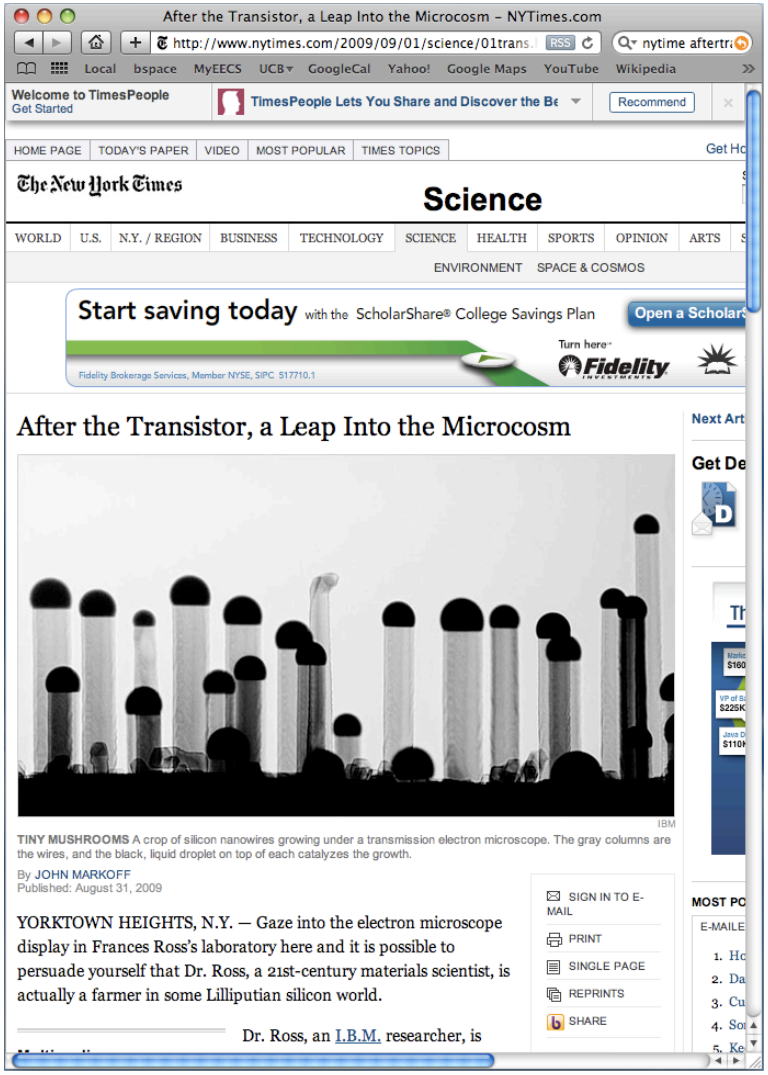
- What operations are defined on names?
- On addresses?

- In C we refer to objects by variable name
  - int x = y + 3;

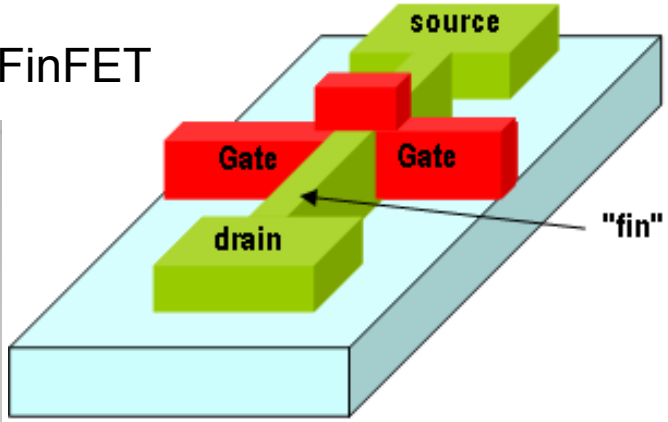
Administration

- All future CS61CL lectures will be held on wed 3-4 in 10 Evans
  - They will be webcast
  - Great for review, but still better to be there
- HW2 will be submitted using the inst tools
  - See announcement, due before section Tu/W
  - See http://inst.eecs.berkeley.edu/cgi-bin/pub.cgi?file=submit.help
  - Please resubmit your hw1 file (ala bspace) as a test
- The waitlist has been accepted into the course
  - We plan to expand existing sections with LAs and laptops
  - TA's will be finalizing the section allocation and ensuring that UCWISE section ID is correct

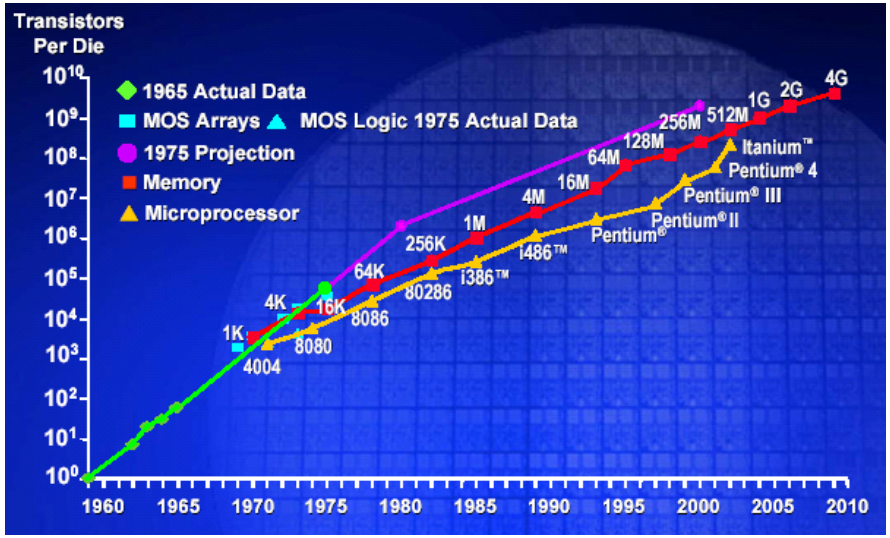
Bears in the news (…oops)



FinFET



Chenming Calvin Hu



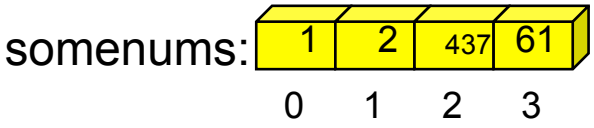
What about complex objects?

- Strings – sequences of characters
- Vectors – sequences of numbers
- Matrixes – 2D collections of numbers
- Records – finite sets of strings and numbers
- Lists
- Tables
- Sounds
- Images
- Graphs
- …

Arrays in C

- **Ordered collection of objects of homogeneous type**
  - **"this string is also a array of chars"**
  - **{1, 2, 437, 61}**

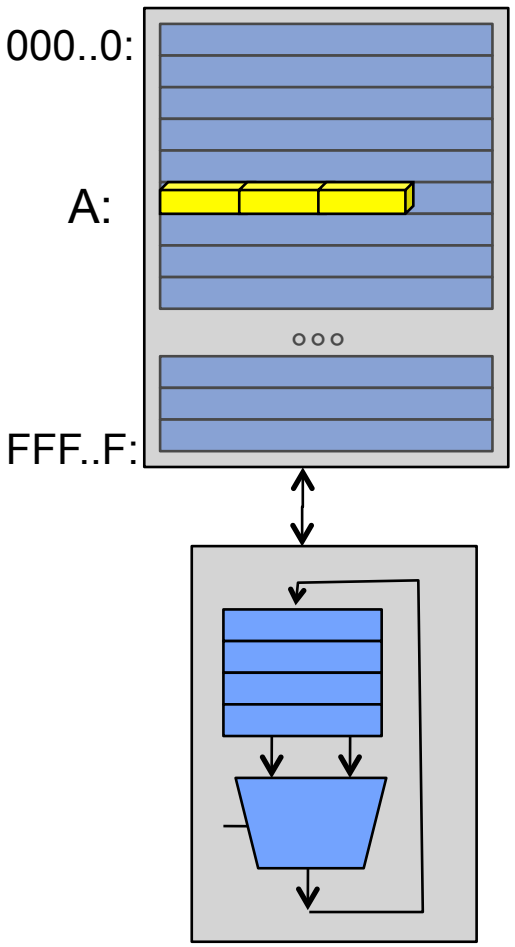
- **Can be declared and named**
  - **char class[7] = "cs61cl";**
  - **char class[] = "cs61cl";**
  - **char classname[8];**
  - **int somenums[4] = {1, 2, 437, 61};**
  - **int morenums[] = {9,8,7,6,5,4,3,2,1,0};**
  - **int numbuf[32];**



- **Can be indexed to refer to an element**
  - **char x = class[0];** **/* access or selection */**
  - **int y = somenums[2];**
  - **somenums[1] = morenums[3];** **/* assignment */**
- **Elements have a static size determined by type**

Where do complex objects reside?

- Arrays are stored in memory
- The variable (i.e., name) is associated with the location (i.e., address) of the collection
  - Just like variables of basic type
- Elements are stored consecutively
  - Can locate each of the elements
- Can operate on the indexed object just like an object of that type
  - A[2] = x + Y[i] – 3;



What can be done with a complex object?

- **Access its elements**
  - **A[i]**
- **Pass it around**
  - **Sort(A)**
  - **x = max(A, n)**

All objects have a size

- The size of their representation
- The size of static objects is given by sizeof operator

```
#include <stdio.h>
int main() {
  char c = 'a';
  int  x = 34;
  int  y[4];
  printf("sizeof(c)=%d\n",   sizeof(c) );
  printf("sizeof(char)=%d\n",sizeof(char));
  printf("sizeof(x)=%d\n",   sizeof(x) );
  printf("sizeof(int)=%d\n", sizeof(int) );
  printf("sizeof(y)=%d\n",   sizeof(y) );
  printf("sizeof(7)=%d\n",   sizeof(7) );
}
```

Complex Objects - really

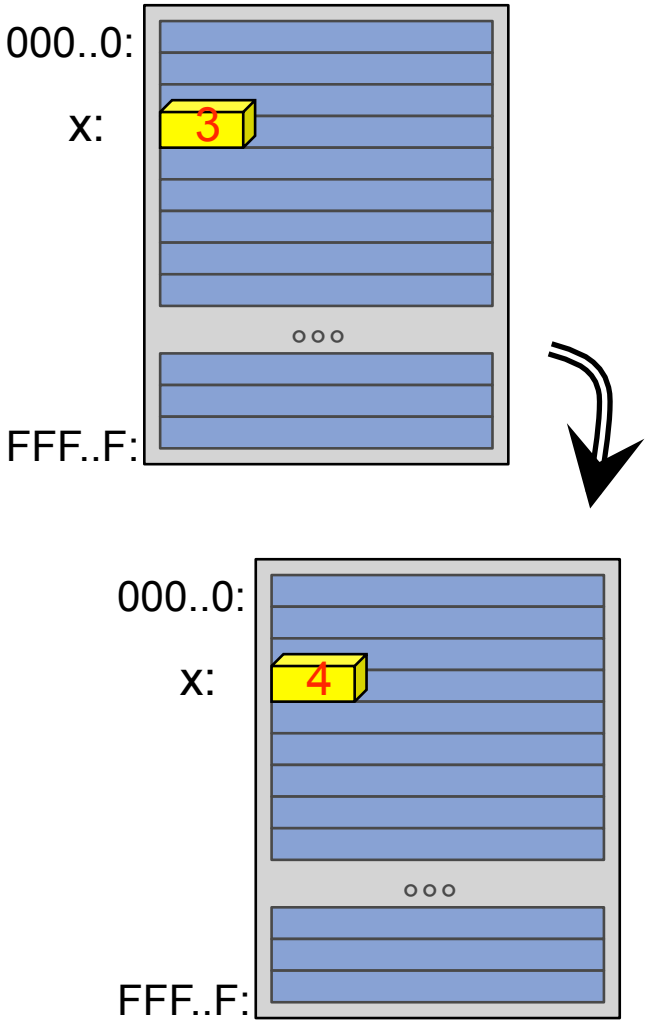
- In many cases we want to treat objects truly as an object
  - Hide its internal representation
    - » Don't know its size, …
  - Invoke methods that work on the object
  - Store it and retrieve it from other objects
- In C this is explicit in the type – pointer
  - char *text = "this is a string";
  - printf("print %s\n", text);

An object and its value…

X = X + 1;

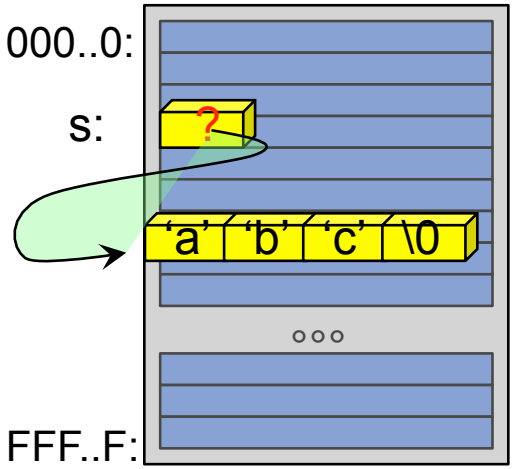
The value of variable X

The storage that holds the value X



With complex objects the distinction matters

```
char *s;
s = "abc";
```

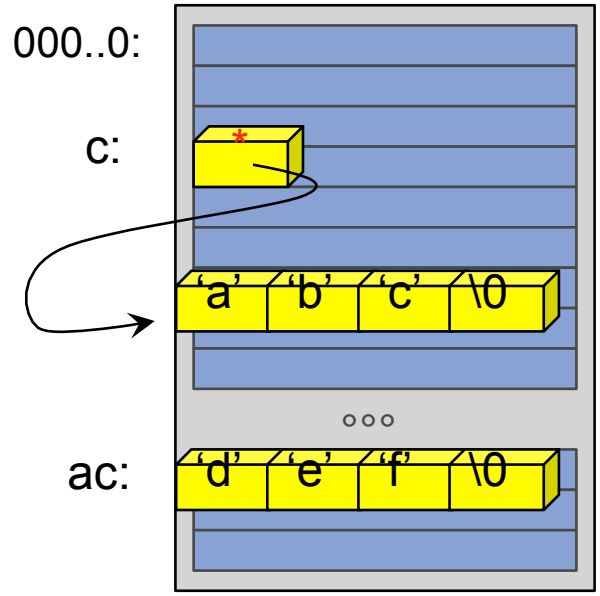


s is a reference to the string "abc"

s is a pointer to the string "abc"

Array variables are also a reference for the object

```
int main() {
  char *c = "abc";
  char ac[4] = "def";
  printf("c[1]=%c\n",c[1] );
  printf("ac[1]=%c\n",ac[1] );
}
```



- **Array name is essentially the address of (pointer to) the zeroth object in the array**
- **There are a few subtle differences**
  - **Can change what `c` refers to, but not what `ac` refers to**

Big Ideas

- Computers manipulate finite representations of things.
- A bunch of bits can represent anything, it is all a matter of what you do with it.
- Finite representations have limitations.

- An object, its value, its location, its reference

- Pointers are THE most subtle concept in C
  - Very powerful
  - Easy to misuse
  - Completely hidden in Java