Would you rather

A) all trafic lights turn green for you

B) never have to stand in line again

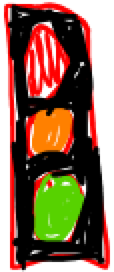
CS 340

Storing Data Types

clicker.cs.illinois.edu

Q1

~Code~
340



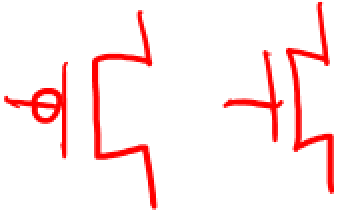
Updates

1. MP2 - due today!
2. HW 3 due Thursday 1:59pm
3. Exam 1 - September 23rd
   a. **Sign up now! By the 18th.**
   b. **Study guide & Practice Exam out**
   c. **No class next Tuesday (23rd)**
   d. **Submit DRES to CBTF Directly**
4. MP3 - out today! Due Tuesday in 2 weeks

no class

	Tues		thur	
	Today MP2	mp3 out	lect Hw3	
	Exam		yes class	
	class MP3		class	

Big Picture



→ logic

calculations

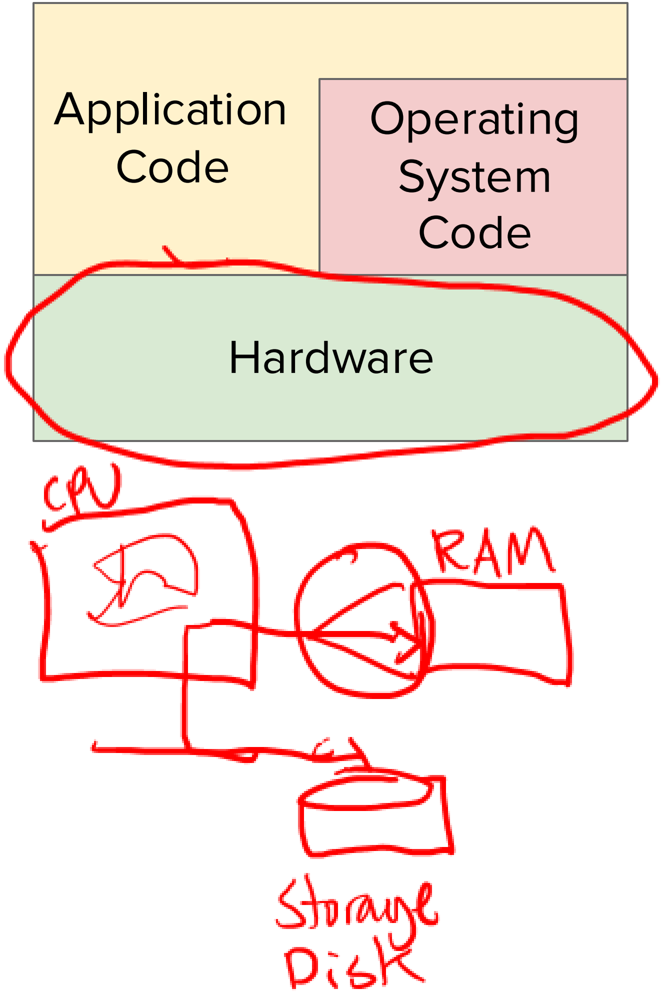
selection

storing

cache

storing bytes

- Hexadecimal
- int, char



Application Code

Operating System Code

Hardware

Storing Data Types

Today’s LGs:

- Be able to go between hex, decimal, and binary
- Understand how bits and bytes are stored in a computer
    - Little and big endianness ←
- Understand implications in C of things are stored in a computer

Bytes

Bytes

Idea 1 - Bytes are a unit indicating 8 bits (1’s and 0’s)

Idea 2 - The base-2 (binary) number system is a way of interpreting 1’s and 0’s to represent bigger numbers

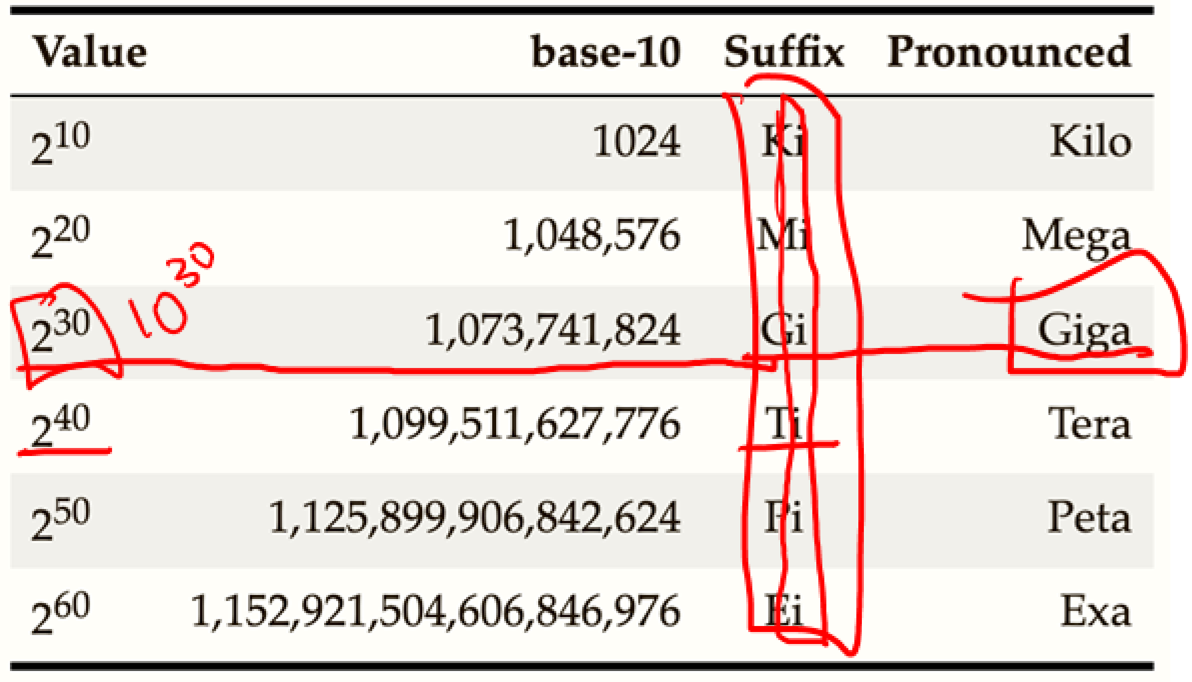
0b0101 = 5

4 2 1

How do we talk about big numbers?

RAM

My computer's memory holds… 64 GB



Value	base-10	Suffix	Pronounced
2^{10}	1024	Ki	Kilo
2^{20}	1,048,576	Mi	Mega
2^{30}	1,073,741,824	Gi	Giga
2^{40}	1,099,511,627,776	Ti	Tera
2^{50}	1,125,899,906,842,624	Pi	Peta
2^{60}	1,152,921,504,606,846,976	Ei	Exa

10^{30}

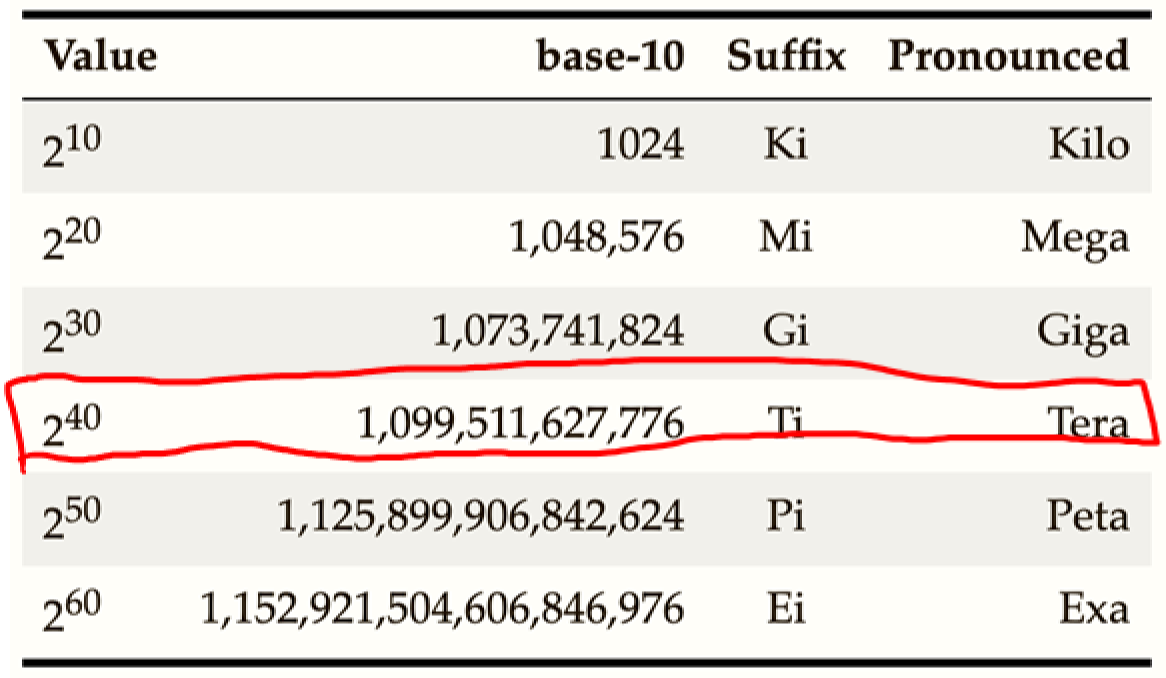
$2^{30} \cdot 2^{6} = 2^{36}$

$2\text{TB} = 2^{40} \cdot 2$

2^{41}

Translate the following, 32TB

2^(45) Bytes = $\overset{32}{\boxed{2^{5}}}\boxed{2^{40}}$? TB



Value	base-10	Suffix	Pronounced
2^{10}	1024	Ki	Kilo
2^{20}	1,048,576	Mi	Mega
2^{30}	1,073,741,824	Gi	Giga
2^{40}	1,099,511,627,776	Ti	Tera
2^{50}	1,125,899,906,842,624	Pi	Peta
2^{60}	1,152,921,504,606,846,976	Ei	Exa

clicker.cs.illinois.edu

Q2

~Code~
340

How do we talk about small numbers?

1 byte

base-2 → $\boxed{1010\ 1000}$ X 1010110001 0 1001 00 0 1000

base-10 → 183 183 75 6 200

base-256 → 256 digits

base-16

What's the biggest value in decimal we can represent with 1 byte?

clicker.cs.illinois.edu

Q3

~Code~
340

1 1 1 1 1 1 1 1

128 64 32 16 8 4 2 1

256

?

255

How many bits can 1 digit in hexadecimal represent?

clicker.cs.illinois.edu

Q4

~Code~
340

1 base-16 digit

										10	11	12	13	14	15
0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	A,	B,	C,	D,	E,	F

4 bits

0000
8 4 2 1

1111 → 15

F

4

How many digits of hexadecimal are needed to represent 1 byte?

clicker.cs.illinois.edu

Q5

~Code~
340

1 byte = 8 bits

0-255 decimal

0b 1010 0110

8 4 2 1   4 2 1

0x A6

byte

Bytes

Idea 1 - Bytes is a unit indicating 8 bits (1's and 0's).

Idea 2 - The base-2 (binary) number system is a way of interpreting 1's and 0's to represent bigger numbers

Idea 3 - To make bytes easier to work with we represent the value the bits hold in hexadecimal instead of binary.

Idea 4 - Hexadecimal is easy to convert to binary and back.

What is 0x4F in binary?

Byte

0b 0100 1111

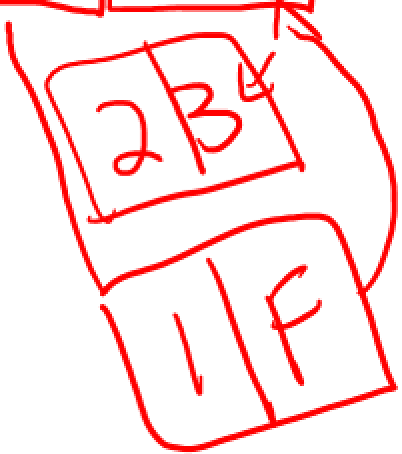
128 64 32 16 8 4 2 1

clicker.cs.illinois.edu

Q6

~Code~
340

What is 0b0001 1111 in hexadecimal?



clicker.cs.illinois.edu

Q7

~Code~
340

How many bytes do I need to store this value? 0xB9E85

0000

5 digits

= 761,733

decimal

clicker.cs.illinois.edu

Q8

~Code~
340

Why we don't use base-15 or base-17 instead of base-16?

clicker.cs.illinois.edu

Q9

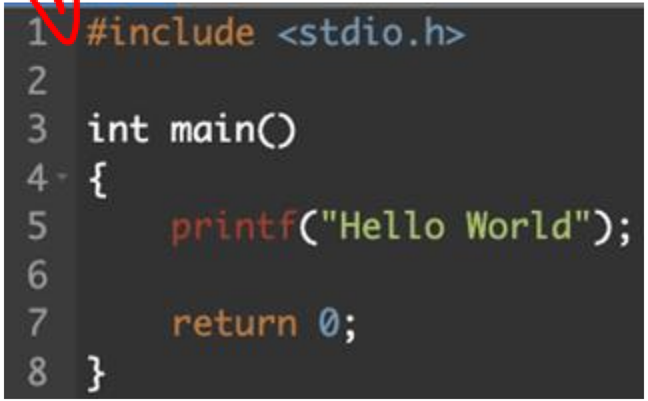
~Code~
340

Data Types

Data is stored as bytes

A data type is how we interpret the bytes

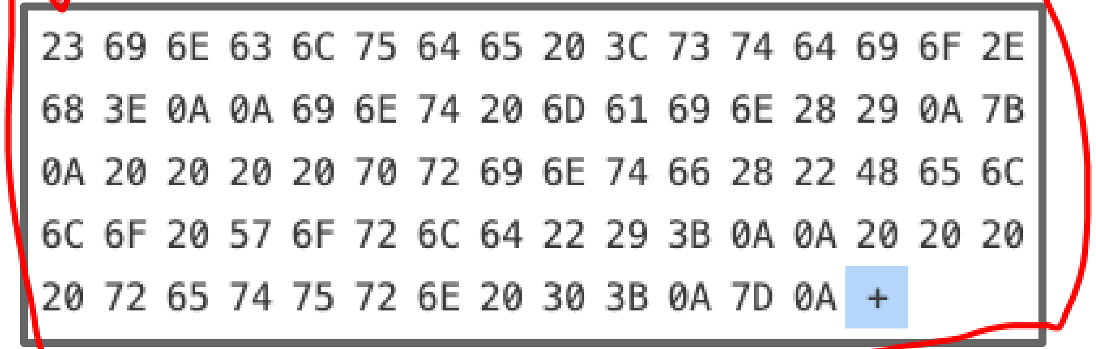
main.c



```c
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```



```
23 69 6E 63 6C 75 64 65 20 3C 73 74 64 69 6F 2E
68 3E 0A 0A 69 6E 74 20 6D 61 69 6E 28 29 0A 7B
0A 20 20 20 20 70 72 69 6E 74 66 28 22 48 65 6C
6C 6F 20 57 6F 72 6C 64 22 29 3B 0A 0A 20 20 20
20 72 65 74 75 72 6E 20 30 3B 0A 7D 0A +
```

Image A is displayed in ascii characters, what is image B displayed in?

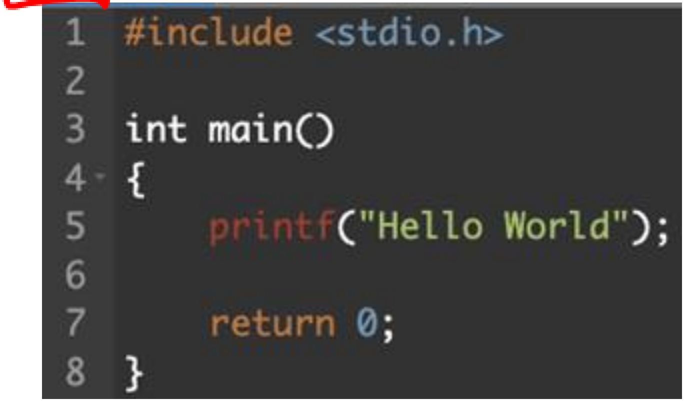
clicker.cs.illinois.edu

Q10

~Code~
340

Hex

A

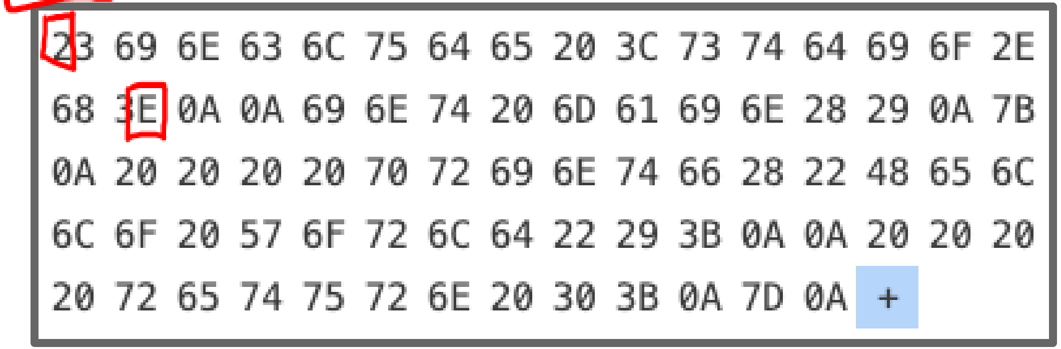


```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

B



```
23 69 6E 63 6C 75 64 65 20 3C 73 74 64 69 6F 2E
68 3E 0A 0A 69 6E 74 20 6D 61 69 6E 28 29 0A 7B
0A 20 20 20 20 70 72 69 6E 74 66 28 22 48 65 6C
6C 6F 20 57 6F 72 6C 64 22 29 3B 0A 0A 20 20 20
20 72 65 74 75 72 6E 20 30 3B 0A 7D 0A +
```

A char holds 1 byte, what's the biggest value it can hold in hexadecimal?

clicker.cs.illinois.edu

Q11

~Code~
340

How many hex digits are a byte?

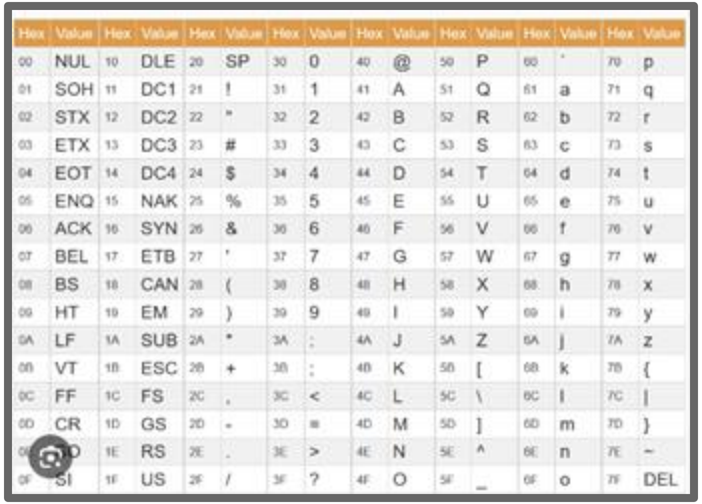
2

0xFF

Char - 1 byte

- Can print out as an ascii character

- Can hold 1 byte - 8 bits



Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Int - 4 bytes

- Can print out as negative or positive number (more on this later)
- Can hold any value between 0x00000000 and 0xFFFFFFFF

Little and Big Endian - the order the bytes are stored

Big endian - big end first

little endian - small end first

int, dec = 400 - 4 bytes = 0x00 00 01 90

Big endian [00] [00] [01] [90]

little endian [90] [01] [00] [00]

Little and Big Endian - the order the bytes are stored

setup little endian

want store 0xF0AB1100 (int)

↓

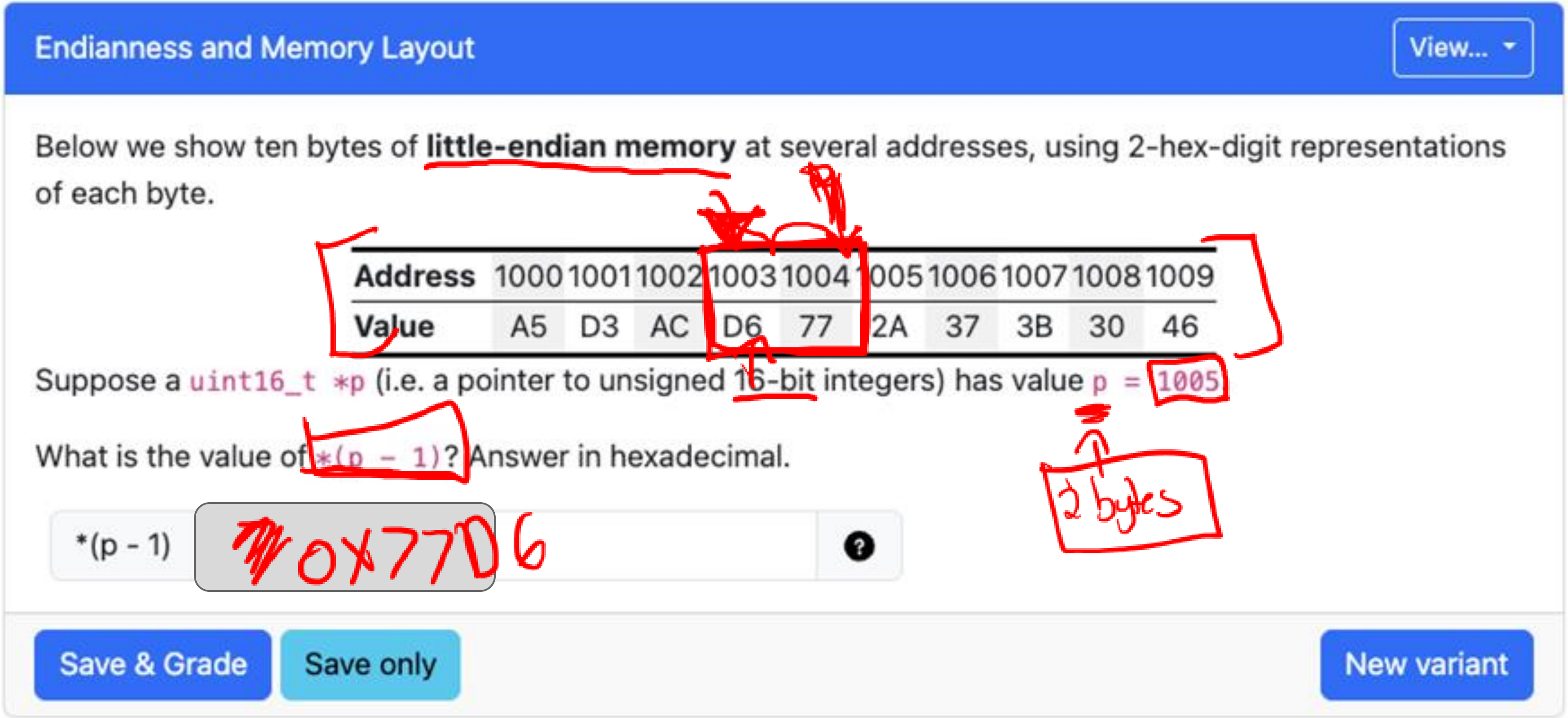
_ [00][11][AB][F0]

address

Char - 0xFA

no endianness relevant

Int - 0xAB000110



Endianness and Memory Layout

Below we show ten bytes of **little-endian memory** at several addresses, using 2-hex-digit representations of each byte.

P

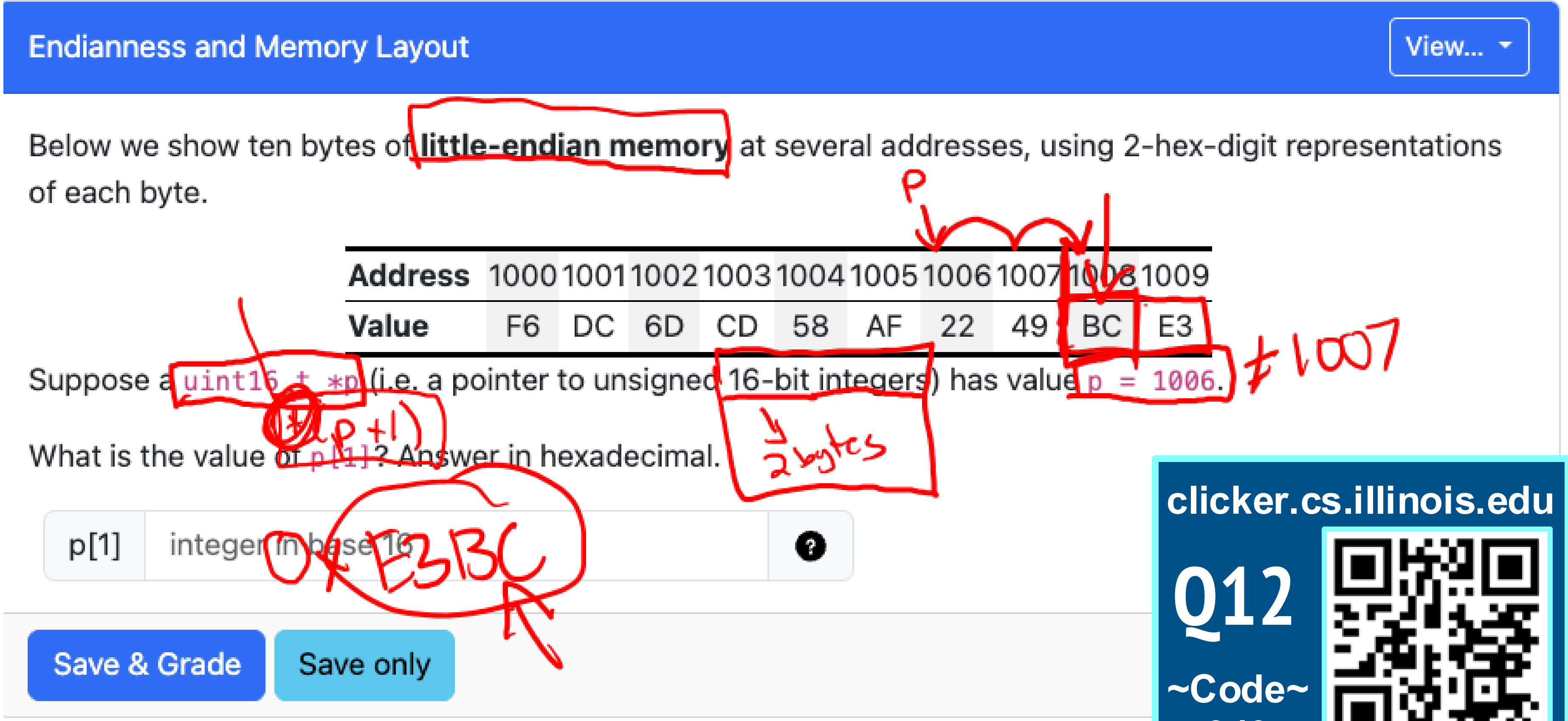
Address	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009
Value	A5	D3	AC	D6	77	2A	37	3B	30	46

Suppose a `uint16_t *p` (i.e. a pointer to unsigned 16-bit integers) has value `p = 1005`

2 bytes

What is the value of `*(p - 1)`? Answer in hexadecimal.

*(p - 1) 0x77D6



Endianness and Memory Layout

Below we show ten bytes of **little-endian memory** at several addresses, using 2-hex-digit representations of each byte.

p

Address	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009
Value	F6	DC	6D	CD	58	AF	22	49	BC	E3

≠ 1007

Suppose a `uint16_t *p` (i.e. a pointer to unsigned 16-bit integers) has value `p = 1006`.

*(p+1)

2 bytes

What is the value of `p[1]`? Answer in hexadecimal.

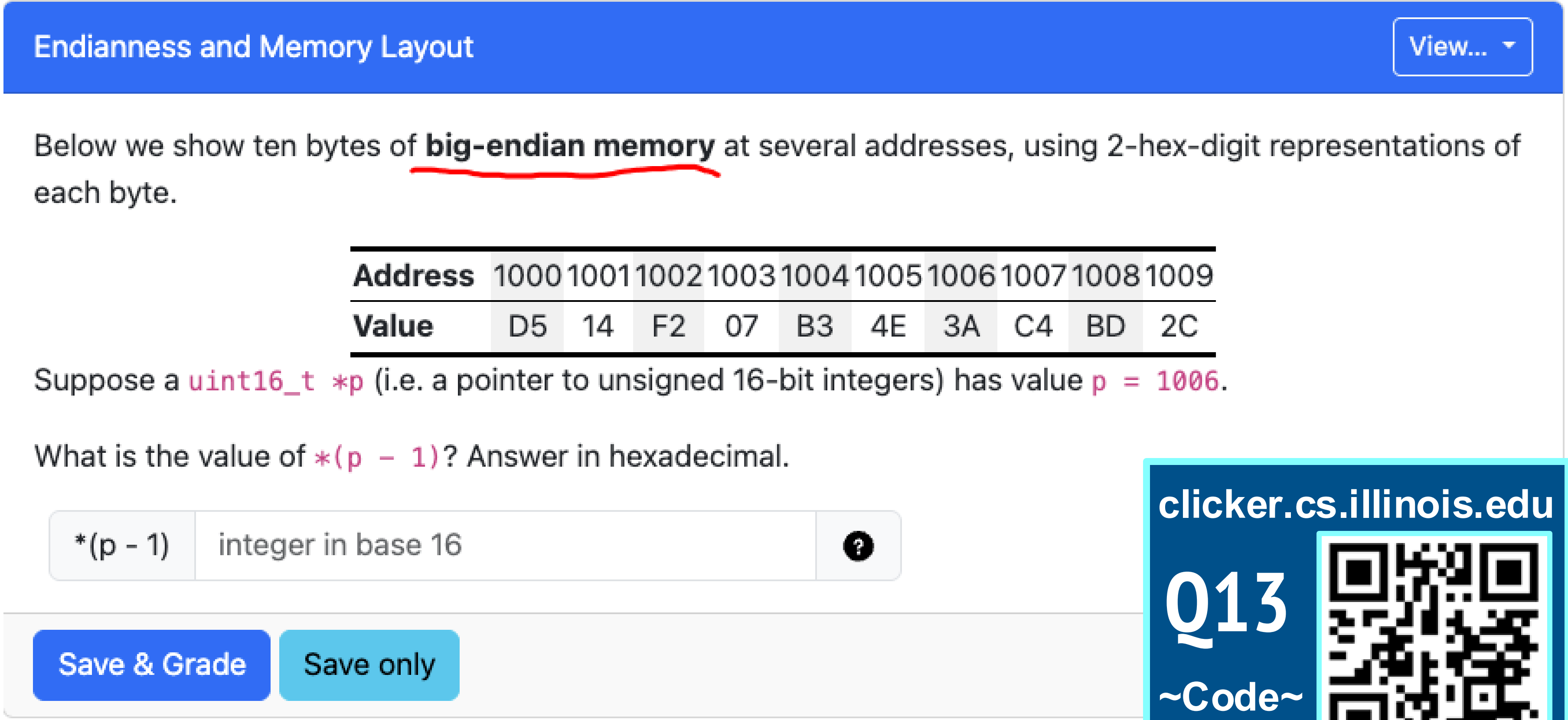
p[1] integer in base 16 ?

0xE3BC

clicker.cs.illinois.edu

Q12

~Code~
340



Endianness and Memory Layout

Below we show ten bytes of **big-endian memory** at several addresses, using 2-hex-digit representations of each byte.

Address	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009
Value	D5	14	F2	07	B3	4E	3A	C4	BD	2C

Suppose a `uint16_t *p` (i.e. a pointer to unsigned 16-bit integers) has value `p = 1006`.

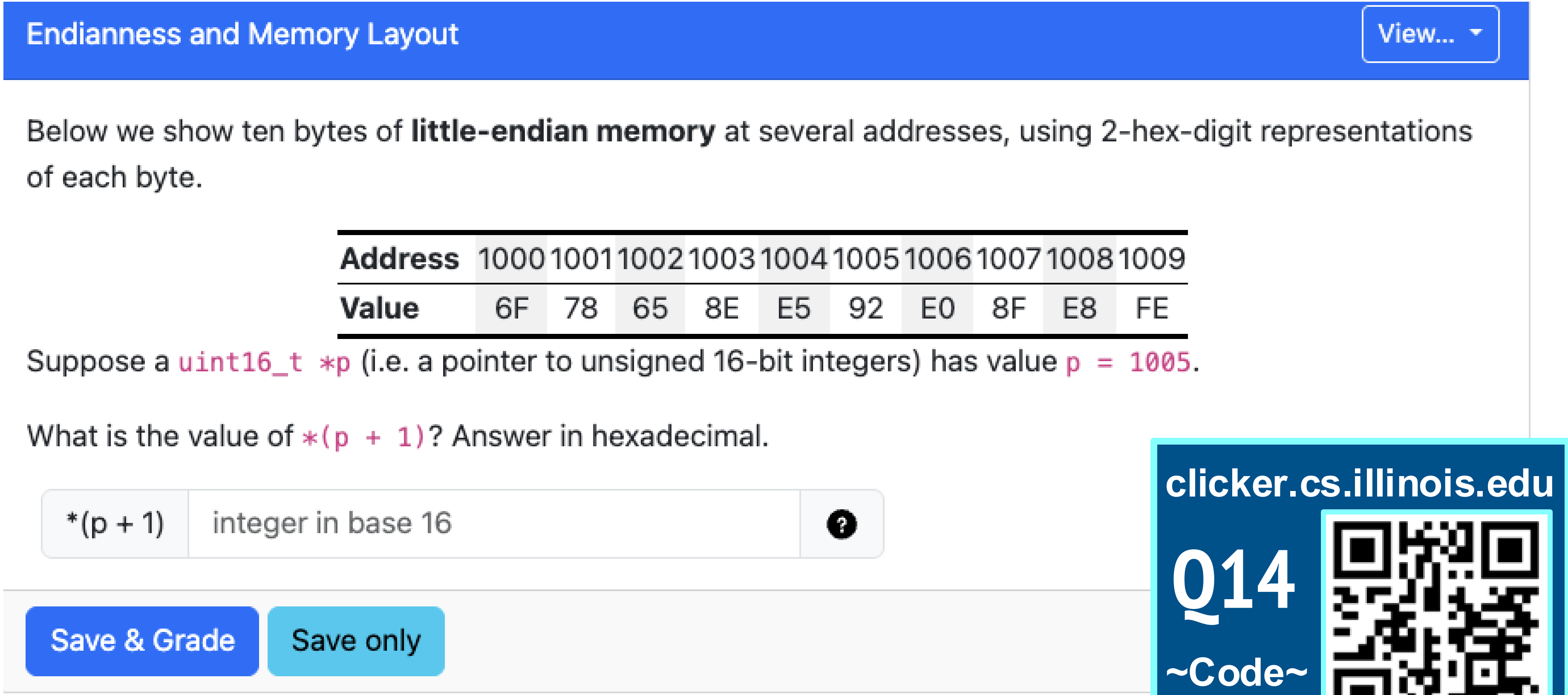
What is the value of `*(p - 1)`? Answer in hexadecimal.

*(p - 1) integer in base 16 ?

clicker.cs.illinois.edu

Q13

~Code~
340



Endianness and Memory Layout

Below we show ten bytes of **little-endian memory** at several addresses, using 2-hex-digit representations of each byte.

Address	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009
Value	6F	78	65	8E	E5	92	E0	8F	E8	FE

Suppose a `uint16_t *p` (i.e. a pointer to unsigned 16-bit integers) has value `p = 1005`.

What is the value of `*(p + 1)`? Answer in hexadecimal.

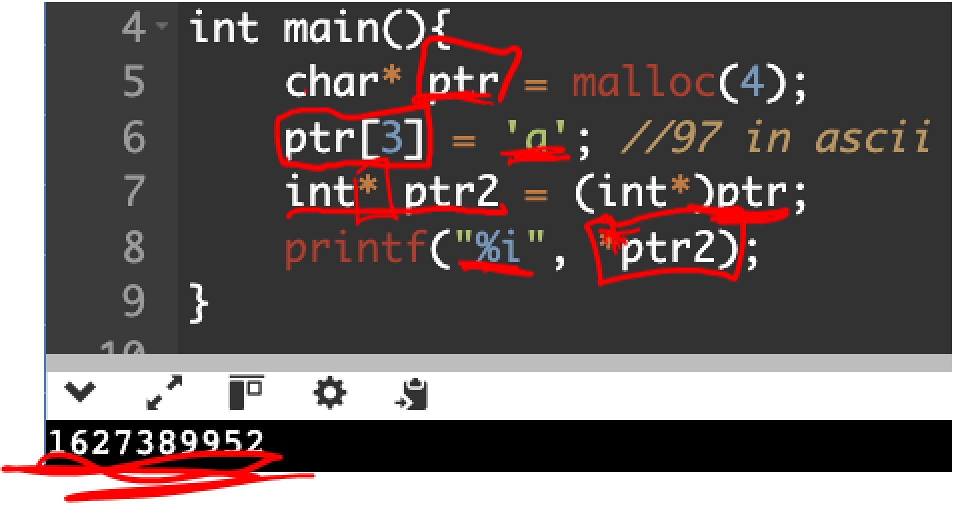
*(p + 1) integer in base 16

clicker.cs.illinois.edu

Q14

~Code~
340

Does my computer use little or big endian?



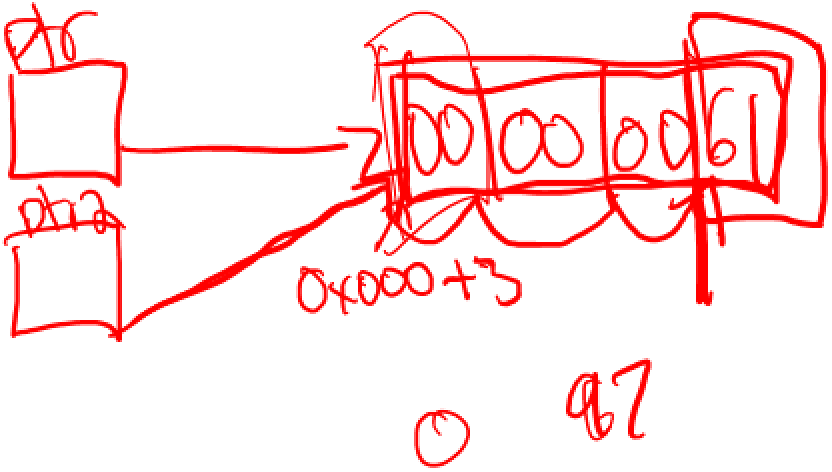
```
int main(){
    char* ptr = malloc(4);
    ptr[3] = 'a'; //97 in ascii
    int* ptr2 = (int*)ptr;
    printf("%i", *ptr2);
}
```

```
1627389952
```

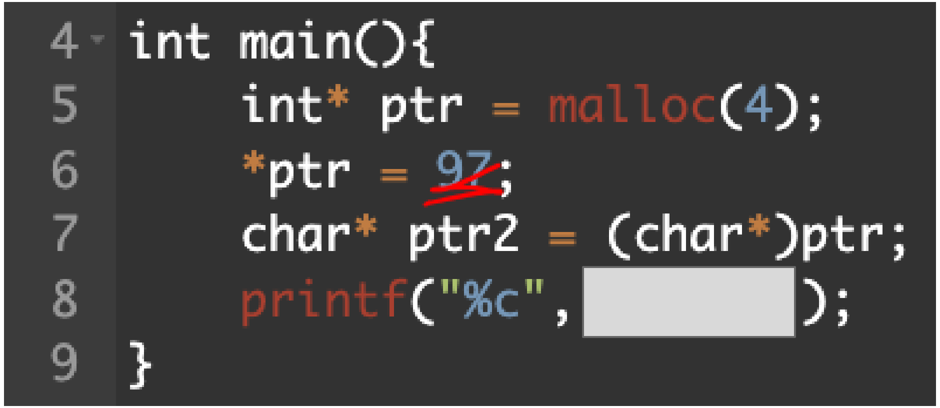
clicker.cs.illinois.edu

Q15

~Code~
340



Given the computer stores bytes in little-endian, how would I print out 'a'?

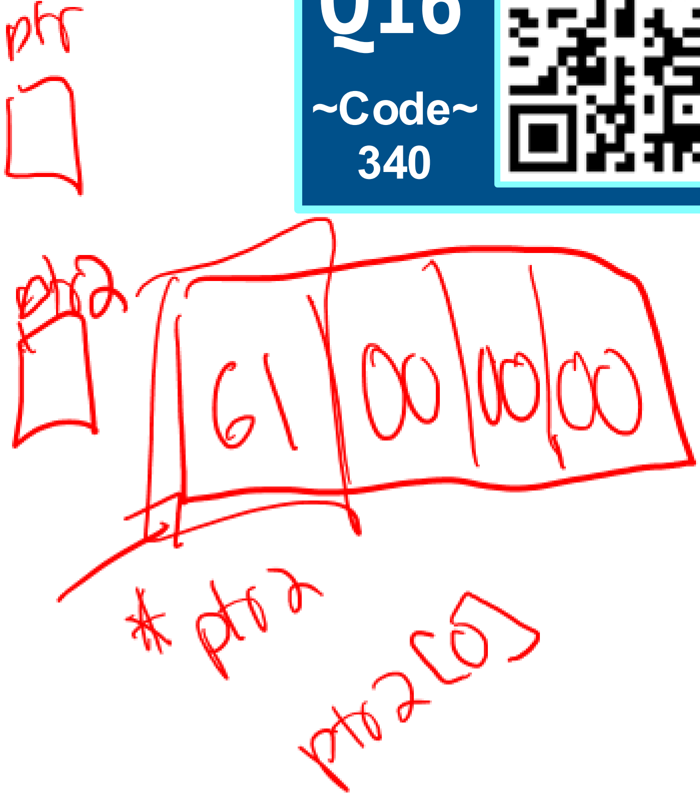


```
int main(){
      int* ptr = malloc(4);
      *ptr = 97;
      char* ptr2 = (char*)ptr;
      printf("%c",         );
}
```

clicker.cs.illinois.edu

Q16

~Code~
340



Any feedback for us?

https://forms.gle/hk8kKjPRfLrowpwX6