clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Storing Data

Updates

1. MP 2 - Linked-List in C due Today

1. MP 3 - PNG out today

1. HW 2 - Due Wednesday at midnight

1. Exam - Next week Thursday

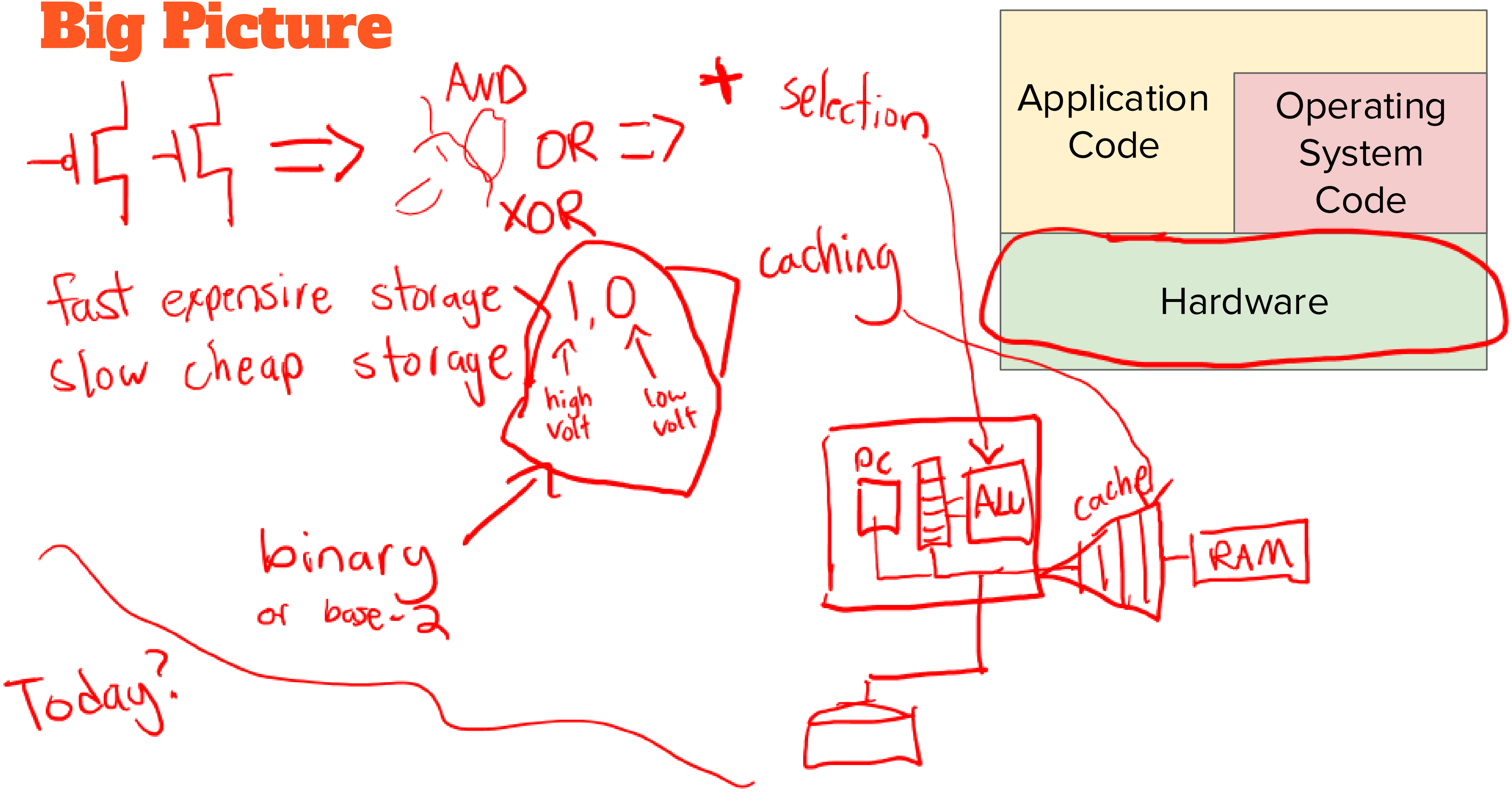
Exam 1

1. Study guide - Released now under announcements on the website

1. Practice exam - Released by the weekend

1. Exam Review - Next Tuesday during class

Big Picture



Application Code

Operating System Code

Hardware

fast expensive storage

slow cheap storage

Storing Data

Today's LGs - Building up your intuition when working with bytes.

- Be able to describe large numbers in bytes.
- Be able to describe small numbers in bytes.
  - Be able to go between hexadecimal, decimal, and binary.
  - Be able to go between little and big endianness.
- Understand the implications of how bytes are stored when coding in C.

Agenda

1. Bytes
   a. Big numbers
   b. Hexadecimal

why

1. Data Types
   a. Storing a char
   b. Storing an int
   c. Little and big endian

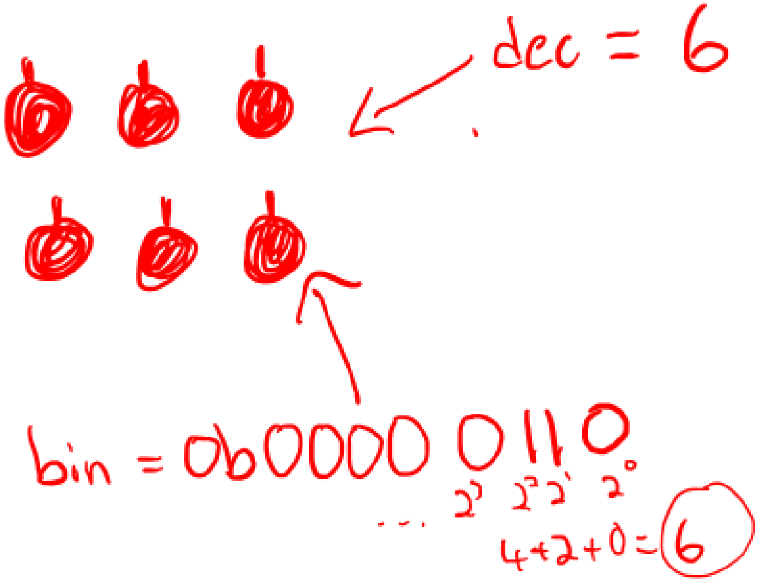
store bytes

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 amounts



dec = 6

bin = 0b0000 0110

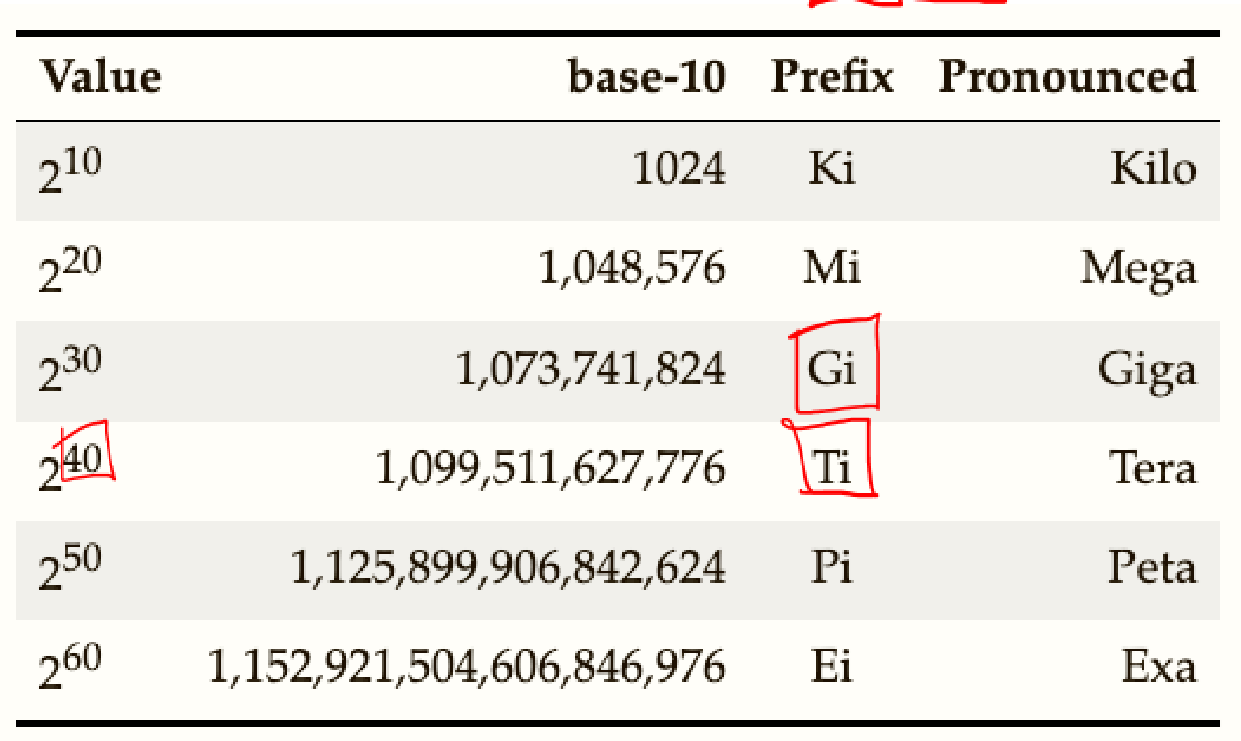
... 2^2 2^1 2^0

$4+2+0=\boxed{6}$

How do we talk about big numbers?

My computer's memory holds... $\boxed{64}$ $\boxed{\text{GB}}$?

$$\underset{\substack{\parallel\\ G}}{2^{30}} \cdot \underset{\substack{\parallel\\ 64}}{2^{6}} = \boxed{2^{36}\text{ bytes}}$$



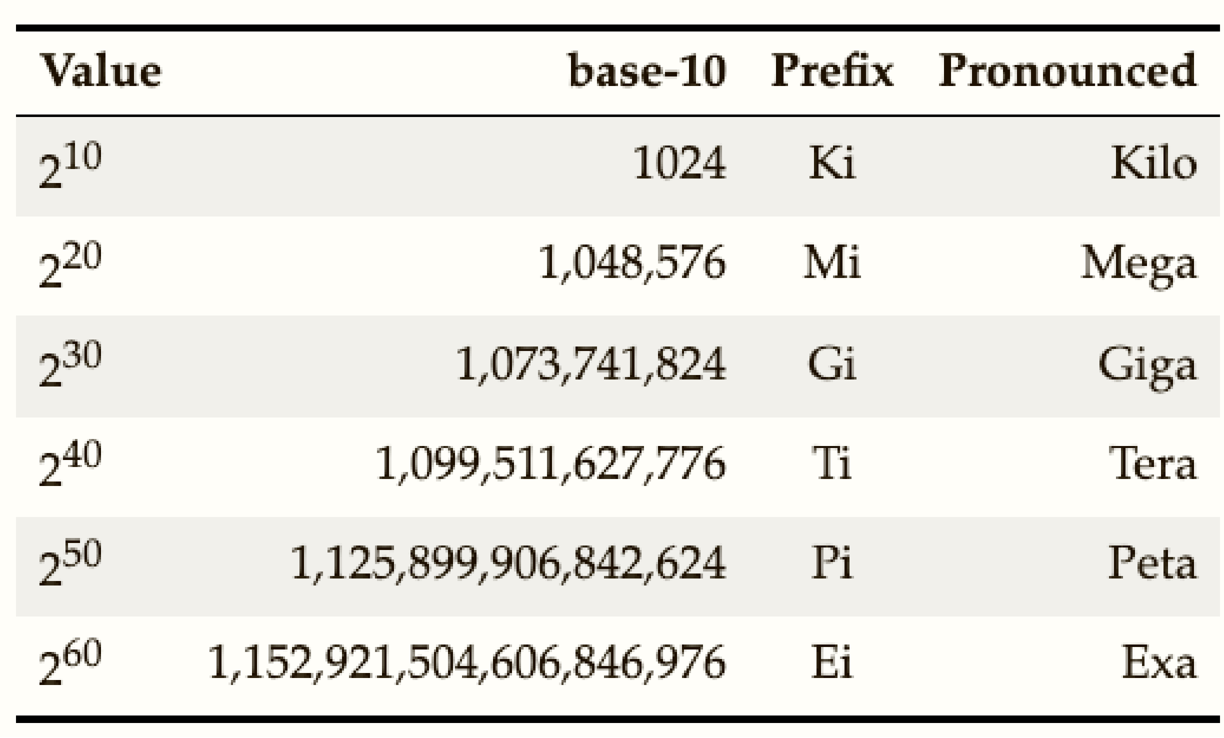
Value	base-10	Prefix	Pronounced
2^{10}	1024	Ki	Kilo
2^{20}	1,048,576	Mi	Mega
2^{30}	1,073,741,824	$\boxed{\text{Gi}}$	Giga
$2^{\boxed{40}}$	1,099,511,627,776	$\boxed{\text{Ti}}$	Tera
2^{50}	1,125,899,906,842,624	Pi	Peta
2^{60}	1,152,921,504,606,846,976	Ei	Exa

$$2^{2} \cdot 2^{40} = \boxed{2^{42}\text{ bytes}}$$

4T

What is the following translated to?

2^(45) Bytes



Value	base-10	Prefix	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

A) 16 GiB

B) 32 GiB

C) 16 TiB

[✓ D) 32 TiB]

E) 16 PiB

How do we talk about small numbers?

1 byte = 1010 0010

32 bits →

4 bytes = 10100010010000111 00

issue = binary is hard to read for humans

idea#1 = what if we write it out in decimal?

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

clicker.cs.illinois.edu

Q3

~Code~
340

RGB = 225 100 25

↑ red ↑ Green ↑ blue

problem - hard to switch to bits
(bit shifting...) UTF-8

737,560 → 10000? ← what is the highest order bit?

← How many bytes does this number need?

A) 127

B) 128

[✓ C) 255]

D) 256

How do we talk about small numbers?

issue - binary is hard to read

256 digits

1111 1111

0000 0000

idea #1 - decimal ✗ too difficult to go back to bytes/bits

idea #2 - 1 digit per byte - base 256 ☹ ✗ can't come up with that many

idea #3 - 2 digits per byte - base-16 ☺

15 - 0 = 16 digits

1111 - 0000

1 byte = 0000 0000

☺

Hexadecimal

Base-16

16 symbols/digits   0-9   A-F

10   6

0x ← header

dec   binary

0x0F == 15 == 0b0000 1111

dec

0x10 == 16 == 0b0001 0000

16   16^0

0x13 == 19 == 0b0001 0011

16^1 16^0

16+3=19

We have

- * less digits per byte than 8 1's and 0's
- * Easy conversion to binary/bits
- * new base number system to learn

How many bits can 1 digit in hexadecimal represent?

1, 0

0xF = how many bits?

clicker.cs.illinois.edu

Q4

~Code~
340

A) 1

B) 2

[✓ C) 4]

D) 8

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

How many bits

clicker.cs.illinois.edu

Q5

~Code~
340

A) 1

[✓ B) 2]

C) 3

D) 4

Big Picture Review

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.

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

What is 0x4F in binary?

clicker.cs.illinois.edu

Q6

~Code~
340

A) 0b0100 1111

B) 0b1111 0100

C) 0b0010 1111

D) 0b1111 0010

What is 0b1011 1111 in hexadecimal?

F

8+0+2+1 = 11

A	[✓ B]	C	D	E	F
10	11	12	13	14	15

0xBF

clicker.cs.illinois.edu

Q7

~Code~
340

A) 0x10F

B) 26

C) 0x11F

[✓ D) 0xBF]

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

clicker.cs.illinois.edu

Q8

~Code~
340

A) 1
B) 2
[✓ C) 3]
D) 4
E) 5
F) 6

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

clicker.cs.illinois.edu

Q9

~Code~
340

A) No idea

B) I think I know

C) I’m confident I know

Does 0x23 equal 23?

↑

16^1

$32 + 3 = 35$

clicker.cs.illinois.edu

Q10

~Code~
340

A) Yes

[✓ B) No]

C) Unsure

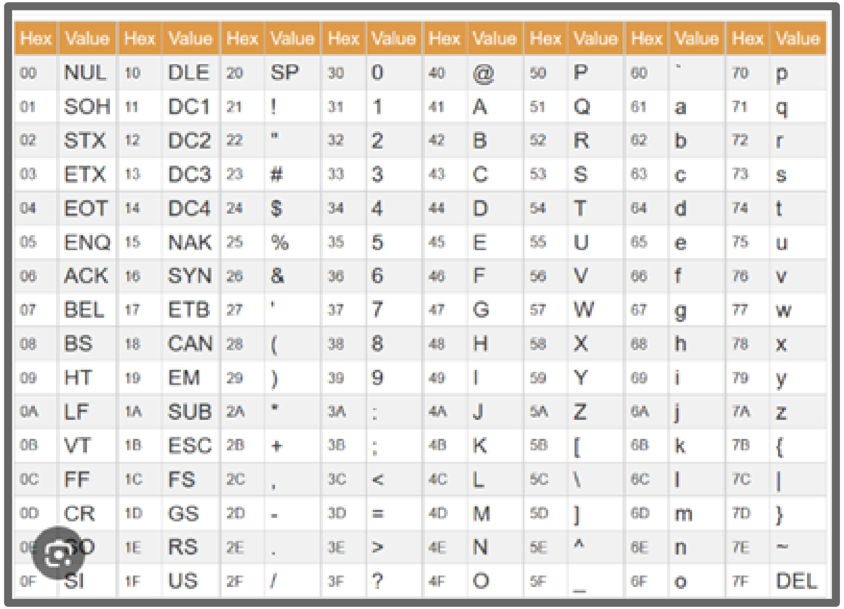
Data Types

char - 1 byte

–number

- 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

interpretations

ascii - Character
binary
hex
dec

– of the bits

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

clicker.cs.illinois.edu

Q11

~Code~
340

A) 0xF

[✓ B) 0xFF]

C) 0b1111 1111

D) 0xFFFF

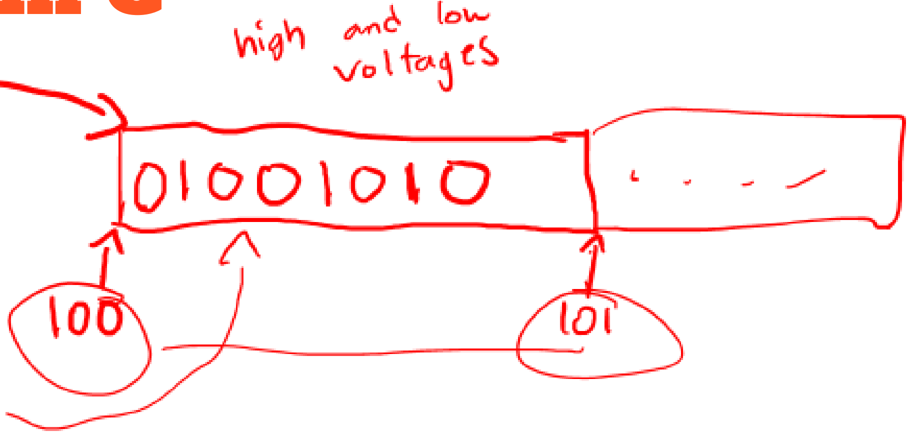
Tying it all together in C

```
char var = 'J';  ← 1 byte
```

→ 74 ascii

0x4A

```
printf("%d", var); → 74
printf("%x", var); → 0x4A
printf("%c", var); → J
```

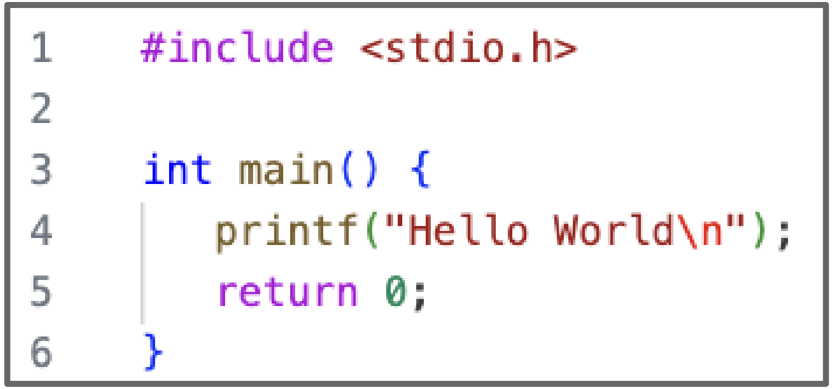


Data is stored as bytes

same file
same bytes
different views

PNG files....
also just bytes

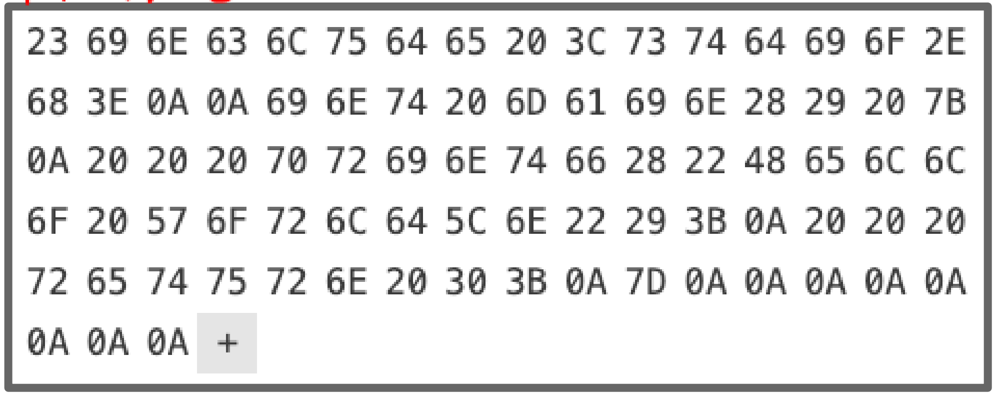
main.c



```c
#include <stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
```

main.c



```
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 20 7B
0A 20 20 20 70 72 69 6E 74 66 28 22 48 65 6C 6C
6F 20 57 6F 72 6C 64 5C 6E 22 29 3B 0A 20 20 20
72 65 74 75 72 6E 20 30 3B 0A 7D 0A 0A 0A 0A 0A
0A 0A 0A
```

What is 0x23 in ASCII?

A) #

B) i

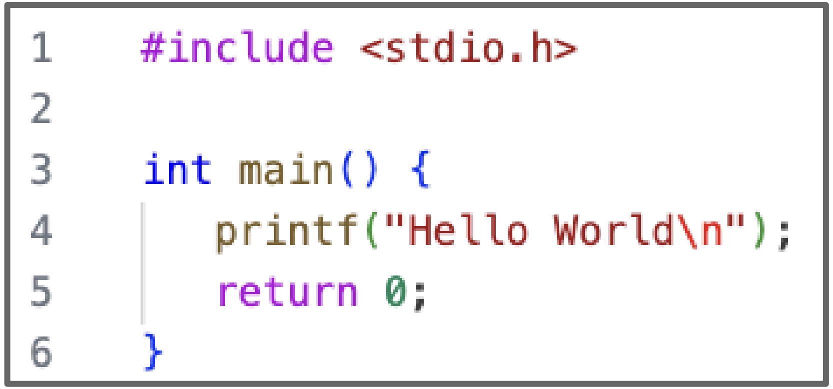
C) 23

D) I need an ascii table please

clicker.cs.illinois.edu

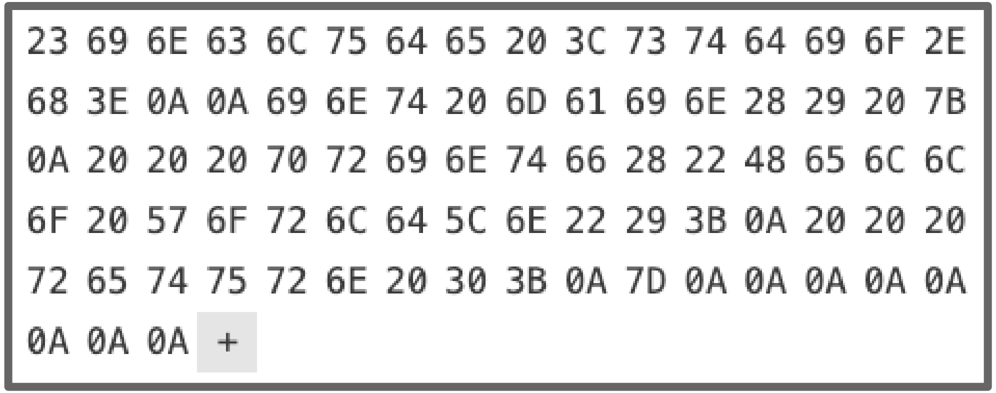
Q12

~Code~
340



```c
#include <stdio.h>

int main() {
    printf("Hello World\n");
    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 20 7B
0A 20 20 20 70 72 69 6E 74 66 28 22 48 65 6C 6C
6F 20 57 6F 72 6C 64 5C 6E 22 29 3B 0A 20 20 20
72 65 74 75 72 6E 20 30 3B 0A 7D 0A 0A 0A 0A 0A
0A 0A 0A +
```

Int - 4 bytes - 32 bits

2 weeks

- Can print out as negative or positive number (more on this later)
- Can hold any value represented by 32 bits (1’s and 0’s)

How do we store the bytes?

Little and Big Endian - the order the bytes are stored

≠ bits

little = little end first

big = big end first

dec = 400 ← int - 4 bytes

⇓

little

hex 0x00 00 01 90

big

computer local

little bytes = [90] [01] [00] [00]

more efficient at some things

big = [00] [00] [01] [90]

human readable

VS.

people use both!

network stuff → big

PNG

Little and Big Endian - the order the bytes are stored

Char - 0xFA

bytes = [FA]

Int - 0xAB000110

[AB][00][01][10]

With which of the following types do we need to consider endianness?

clicker.cs.illinois.edu

Q13

~Code~
340

A) char

[✓ B) char*]

[✓ C) int]

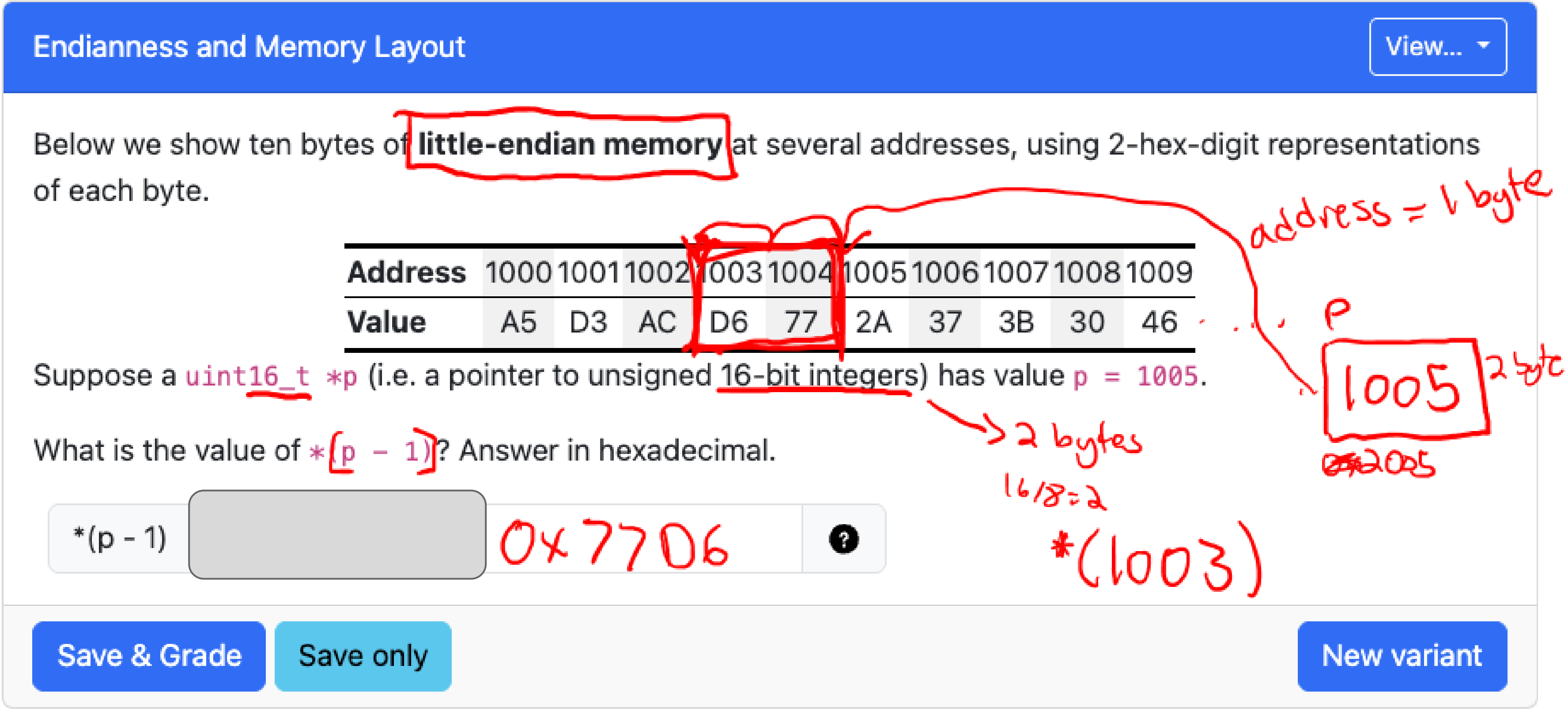
D) array of chars → [0, 1, 3] → [00] [01] [03]

1 char, 2nd char, 3rd char

0x100, 101, 102

E) None of the above

endianness for multi byte types



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	A5	D3	AC	D6	77	2A	37	3B	30	46

address = 1 byte

p

$\boxed{1005}$ 2 byte

2005

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

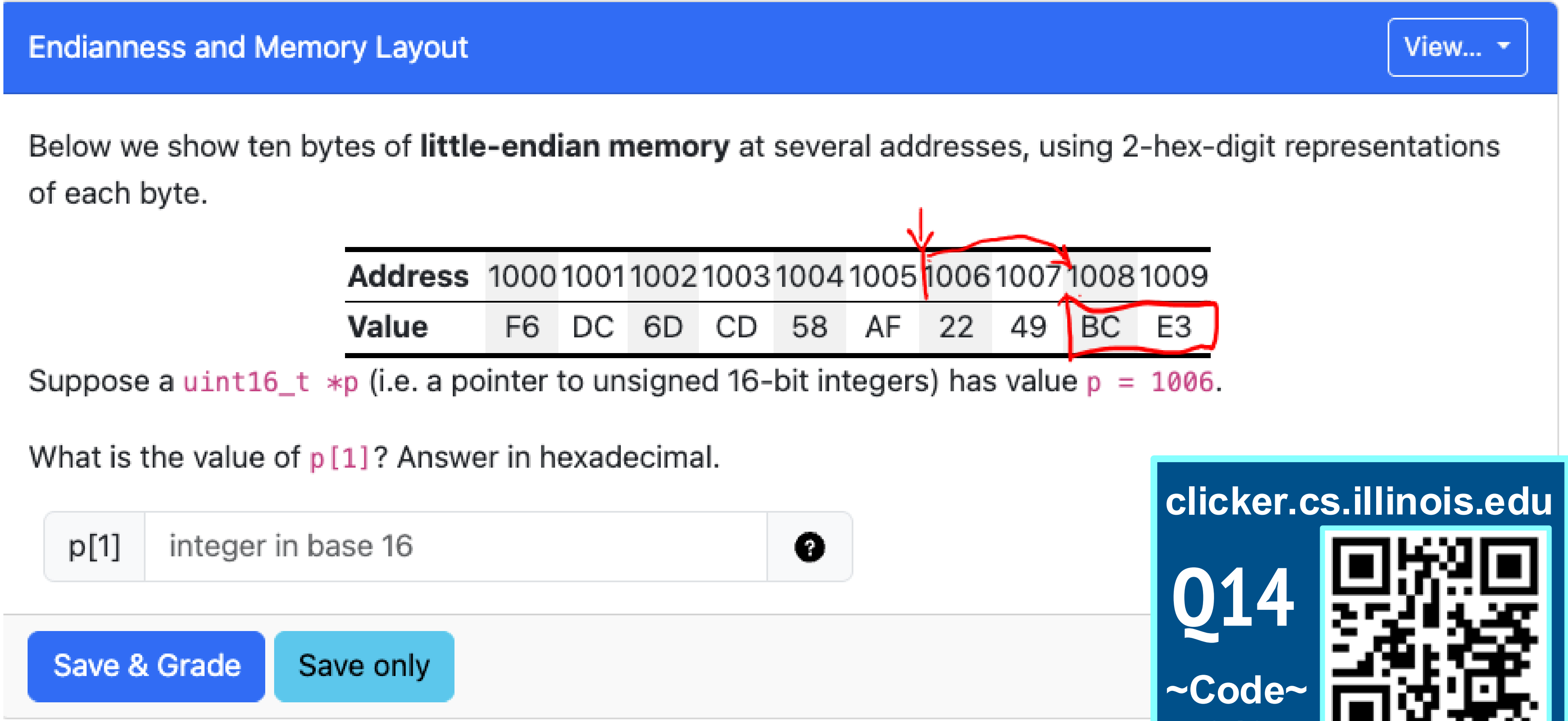
→ 2 bytes

16/8=2

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

*(1003)

*(p - 1) 0x7706



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	F6	DC	6D	CD	58	AF	22	49	BC	E3

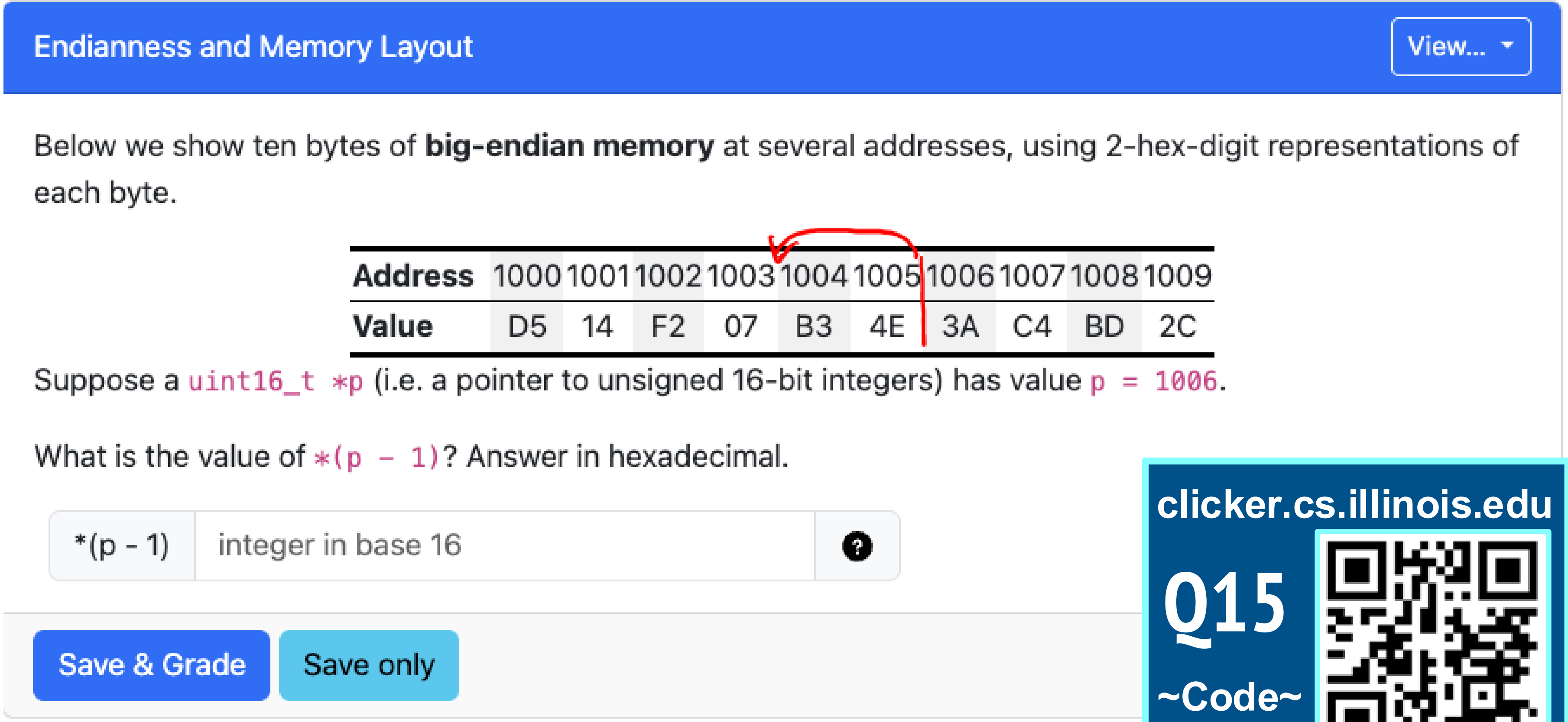
Suppose a `uint16_t *p` (i.e. a pointer to unsigned 16-bit integers) has value `p = 1006`.

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

clicker.cs.illinois.edu

Q14

~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`.

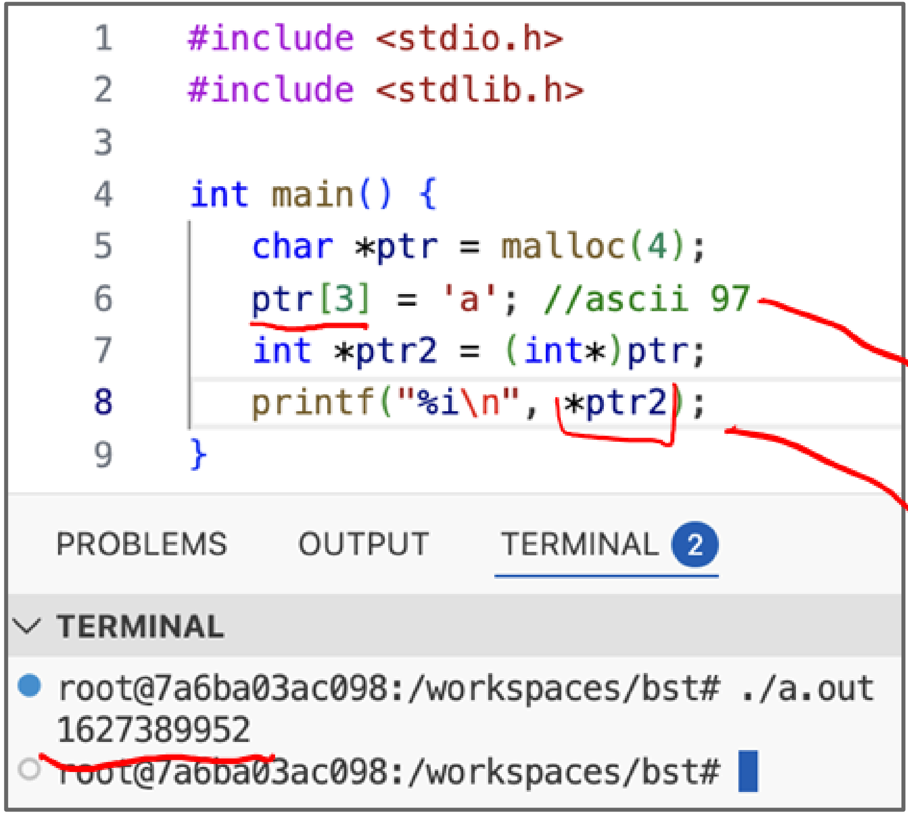
What is the value of `*(p - 1)`? Answer in hexadecimal.

clicker.cs.illinois.edu

Q15

~Code~
340

Challenge: Does this computer use little or big endian?



```c
#include <stdio.h>
#include <stdlib.h>

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

```
root@7a6ba03ac098:/workspaces/bst# ./a.out
1627389952
root@7a6ba03ac098:/workspaces/bst# 
```

[✓ A) Little]

B) Big

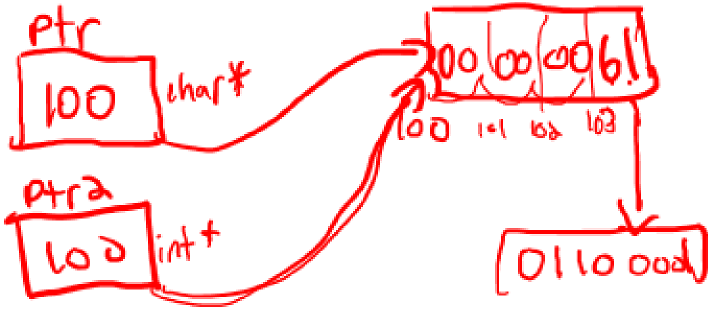
C) Can’t tell

clicker.cs.illinois.edu

Q16

~Code~

340



Interpret 4 bytes at 100 as an int in [little or big] endian

↓ 61