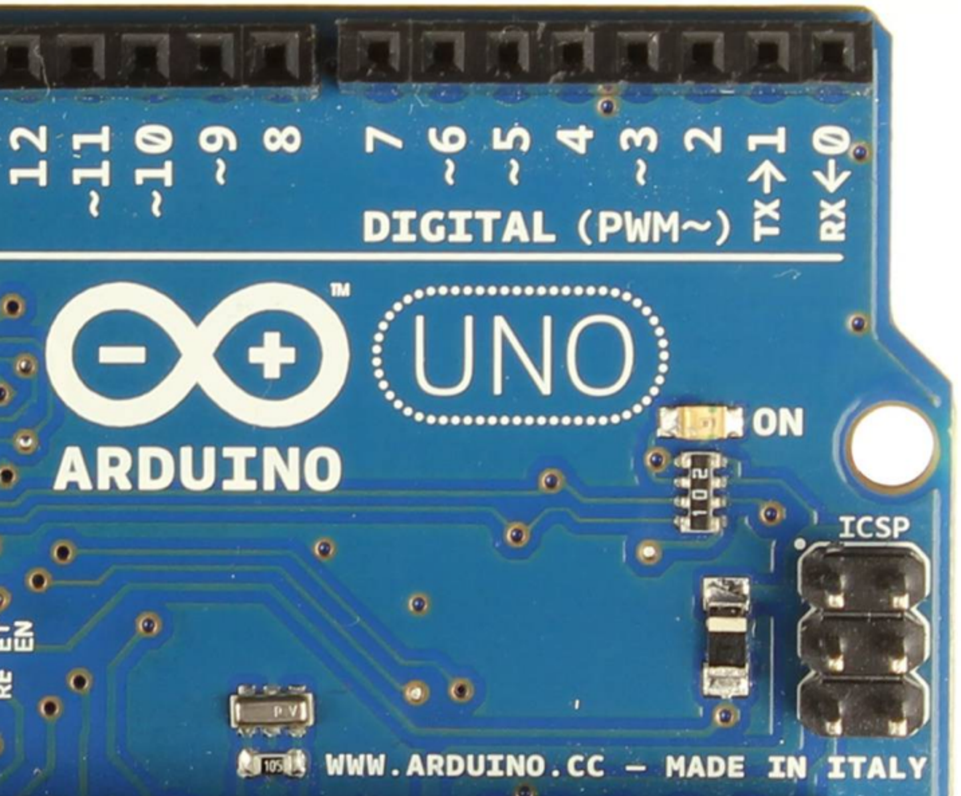
Programming for Arduino

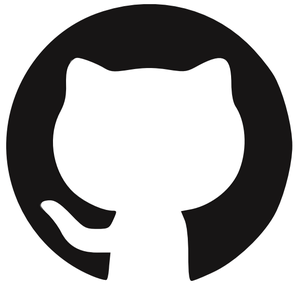


Brown IEEE
Alexander Hadik

Getting Started

www.sublimetext.com

www.arduino.cc



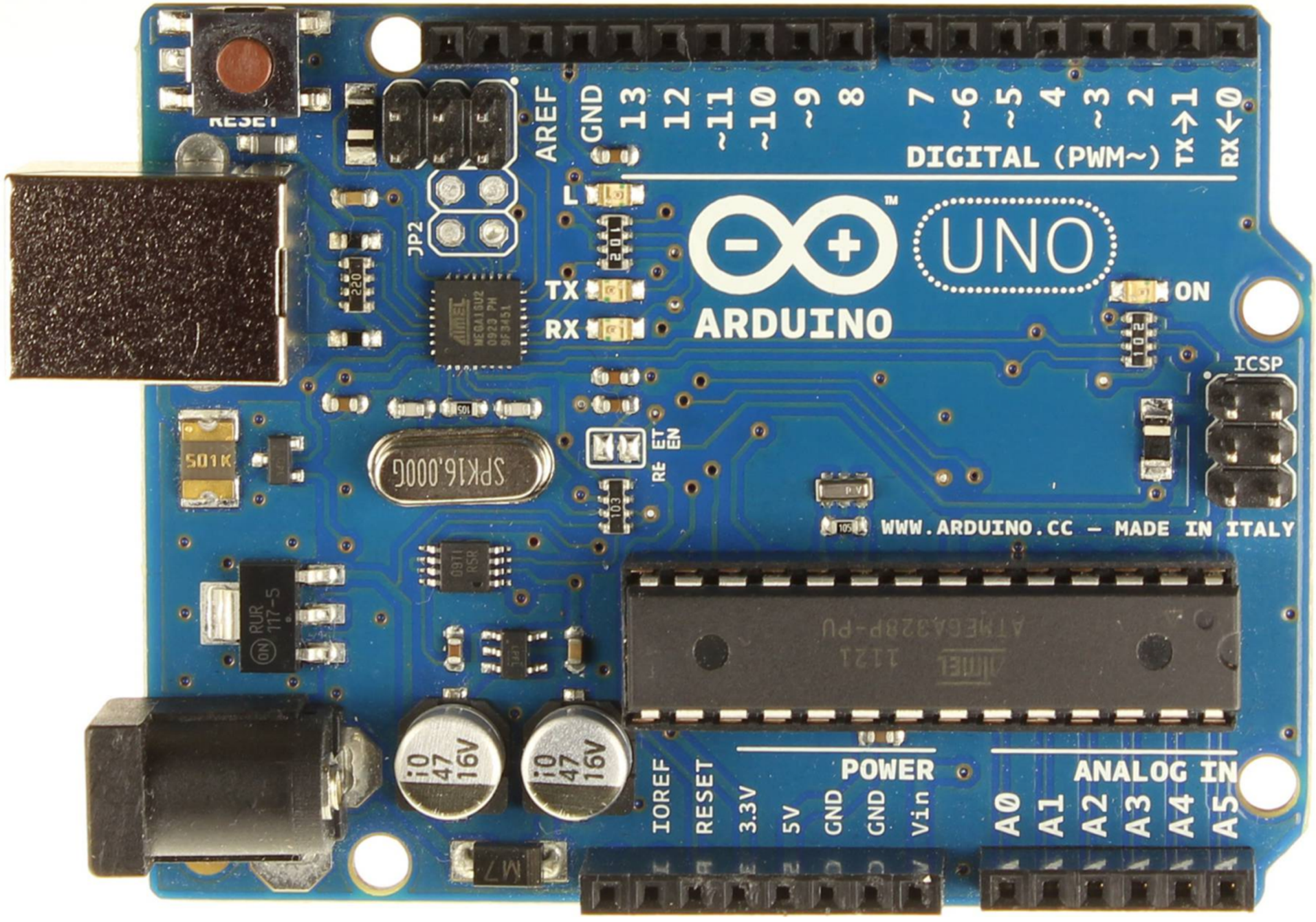
github.com/ahadik/arduino_workshop

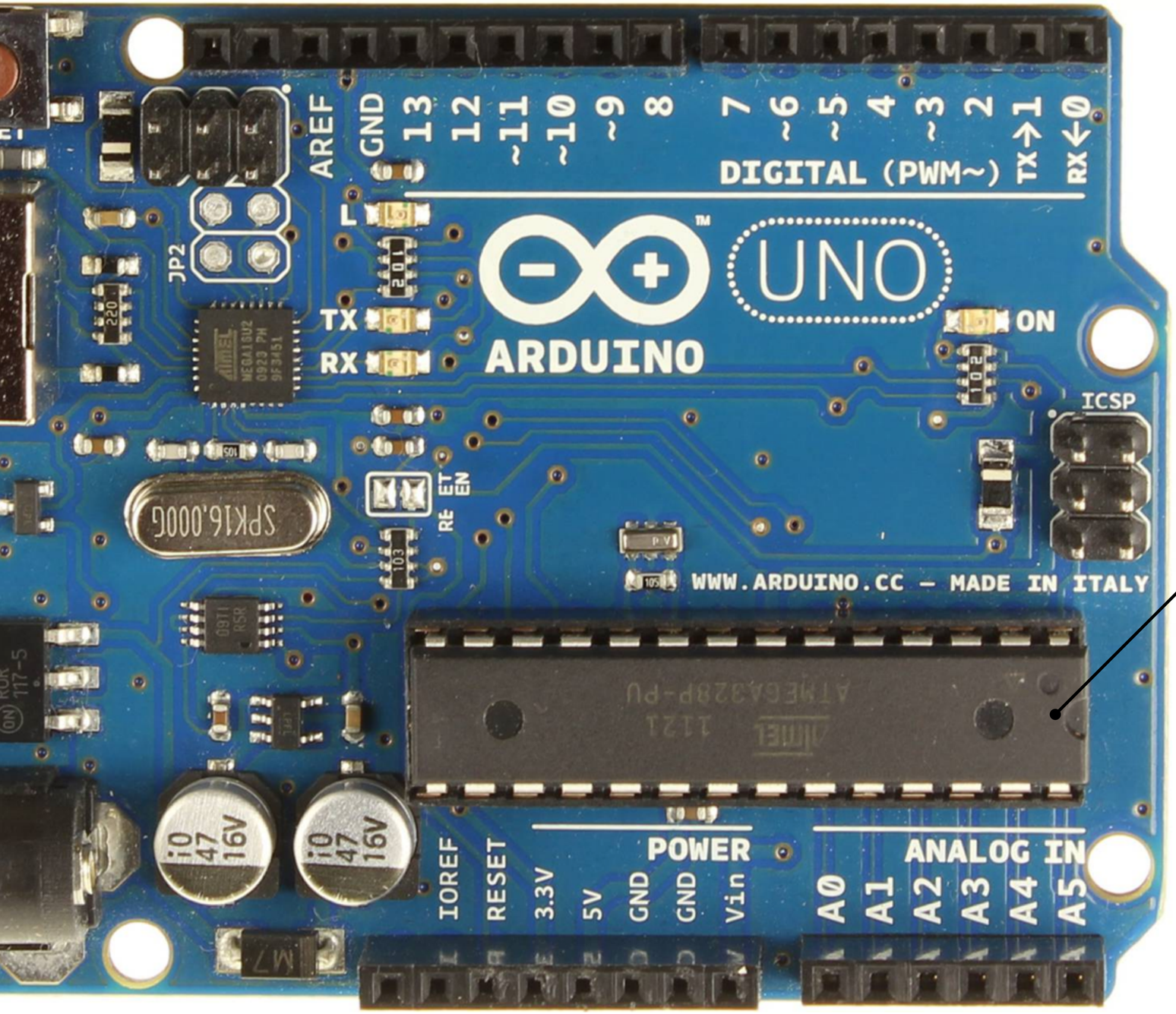
Architecture

Data Types

Programming

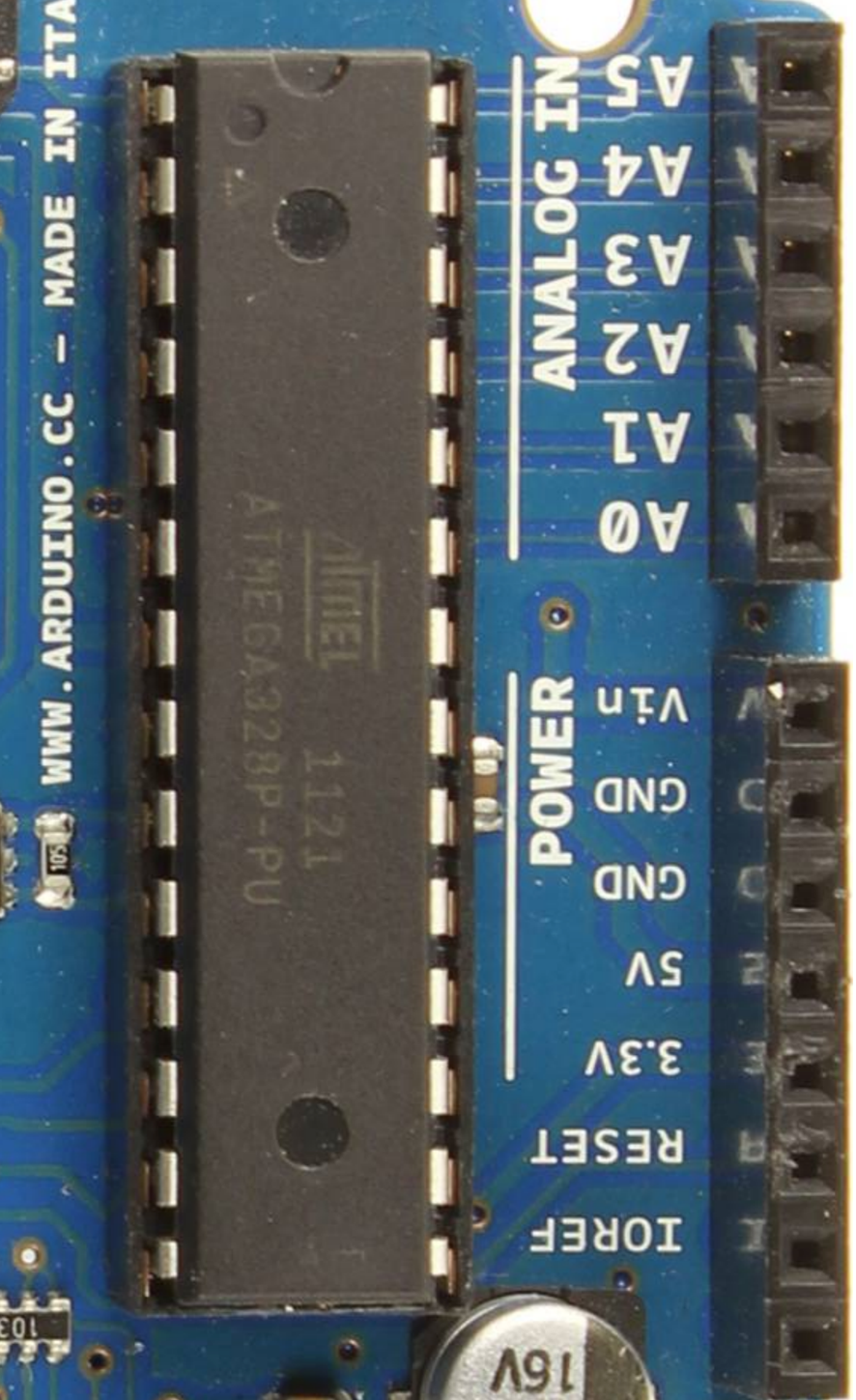
Architecture

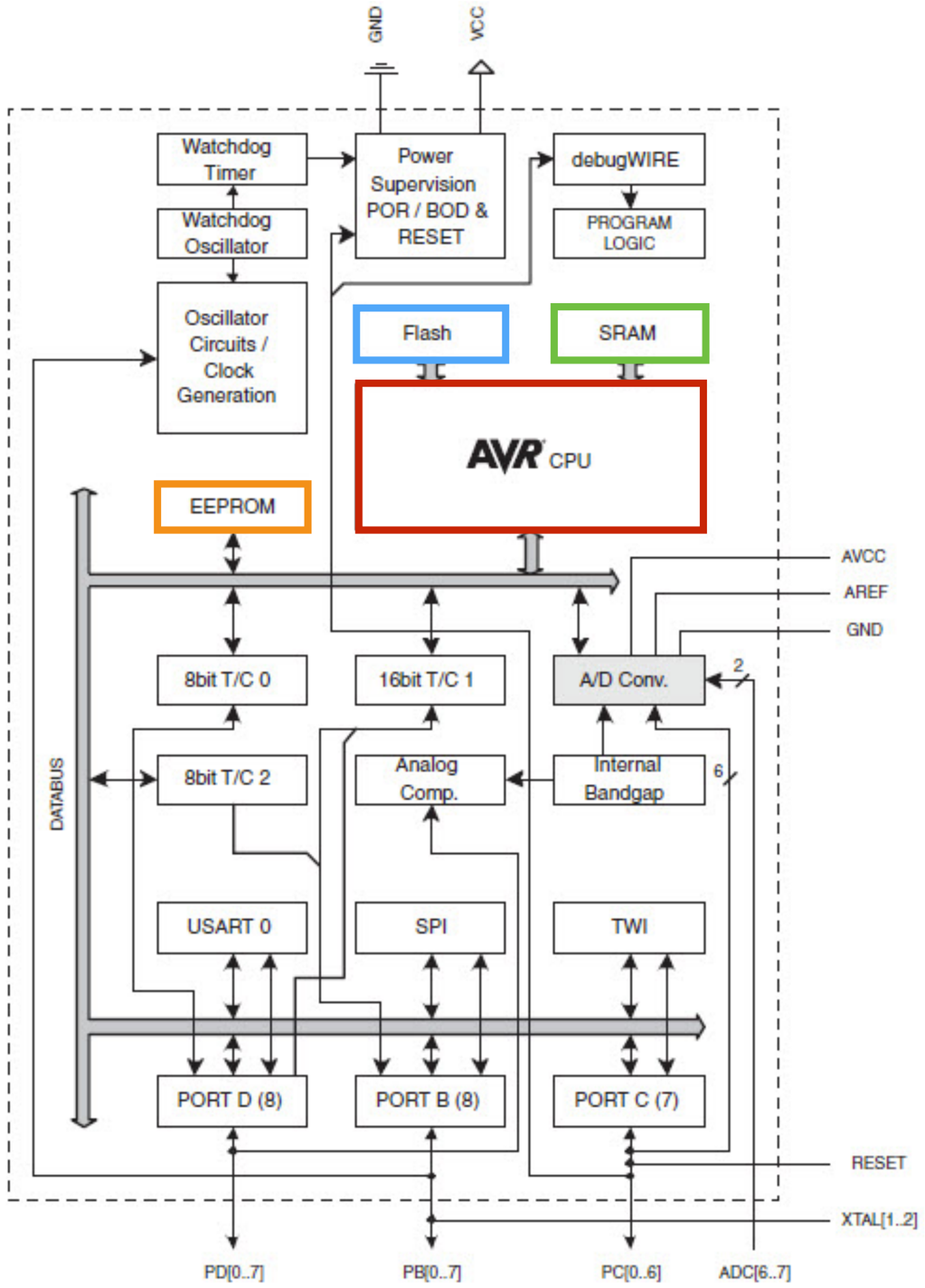

RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
UNO
ARDUINO
ON
TX
RX
ICSP
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5



Processor

Atmega328-P



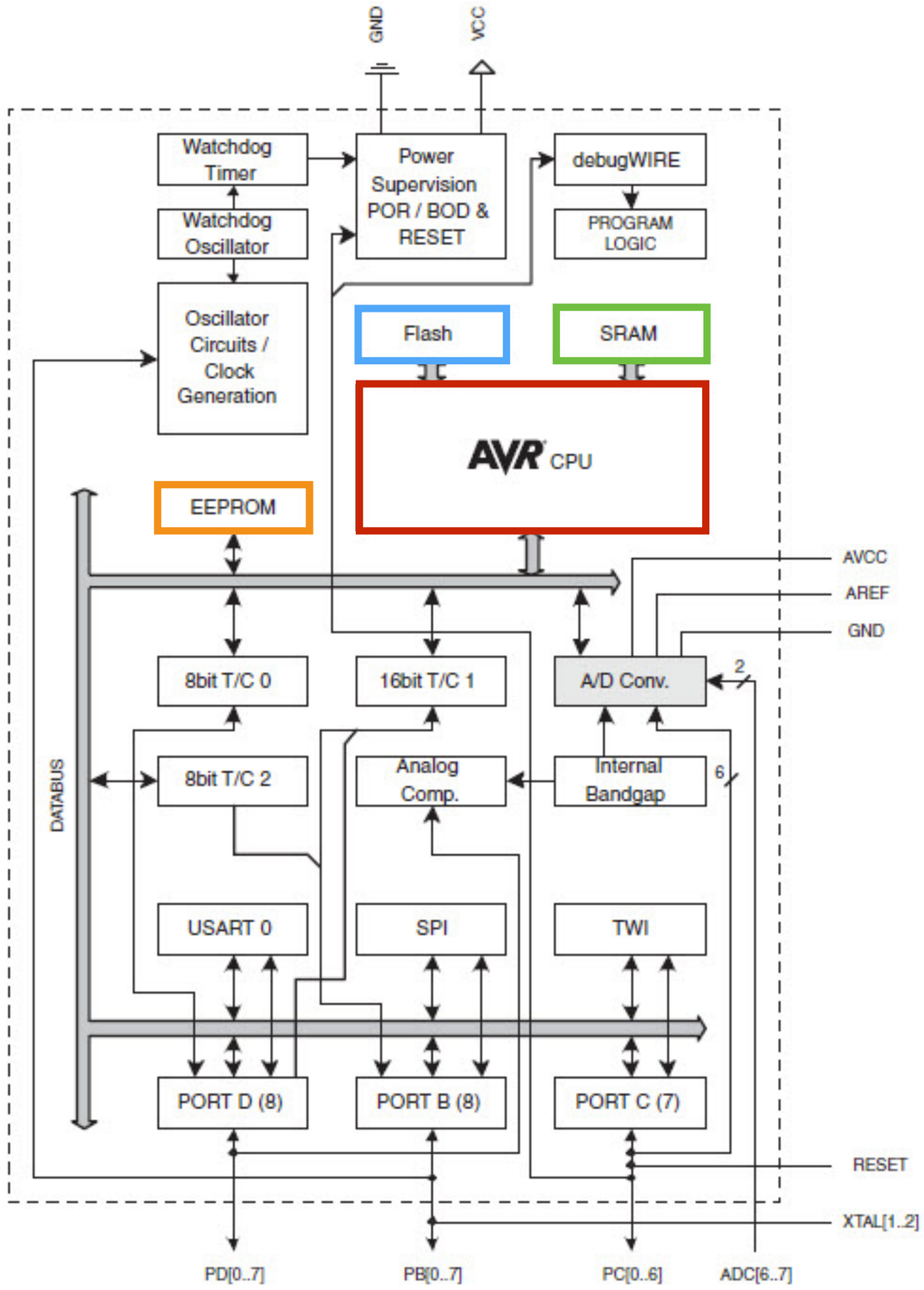


Flash

SRAM

EEPROM

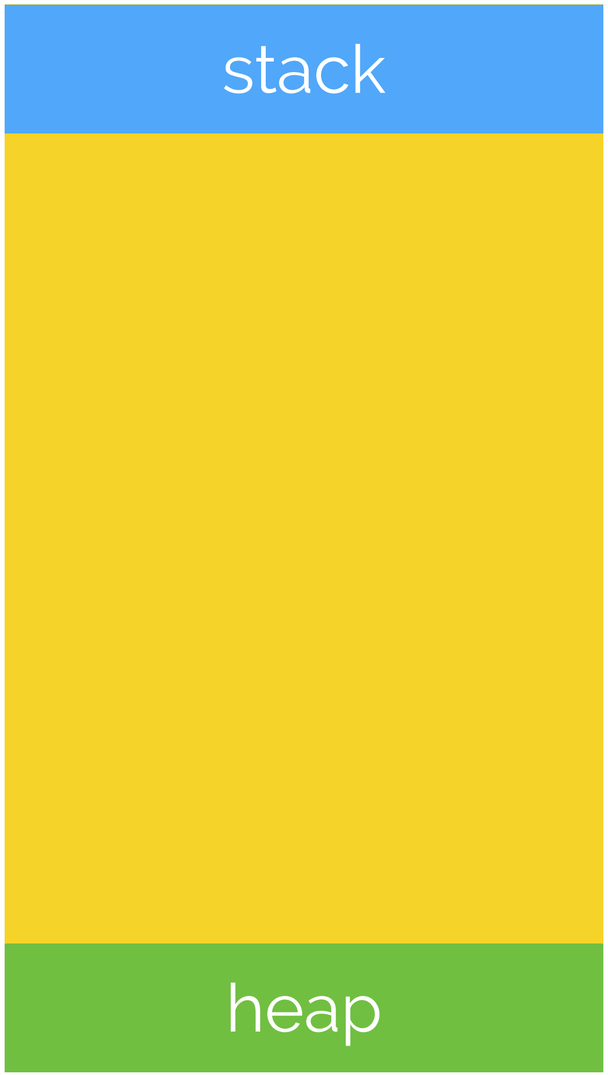
CPU



Flash	SRAM	EEPROM
32 kb	2 kb	1 kb
code	local variables	persistent data
initial values	volatile	
non-volatile		

SRAM

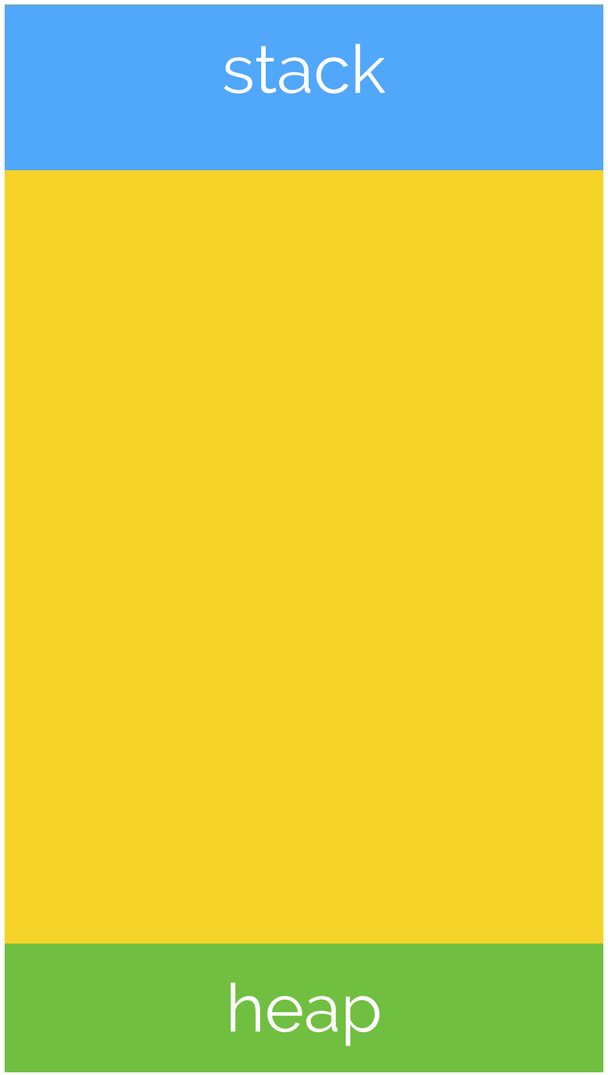
```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

SRAM

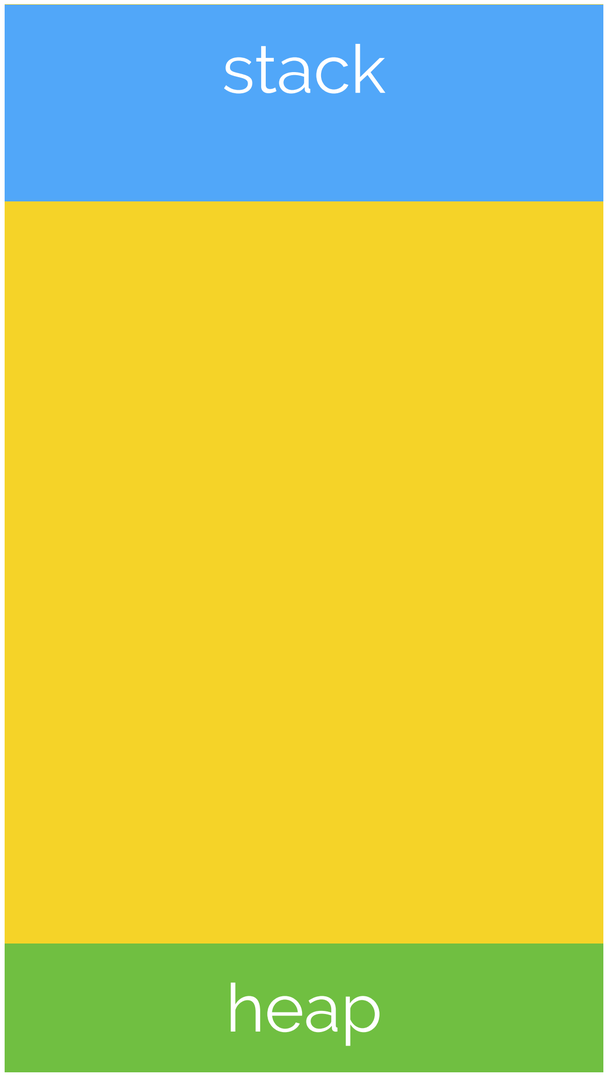
```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

SRAM

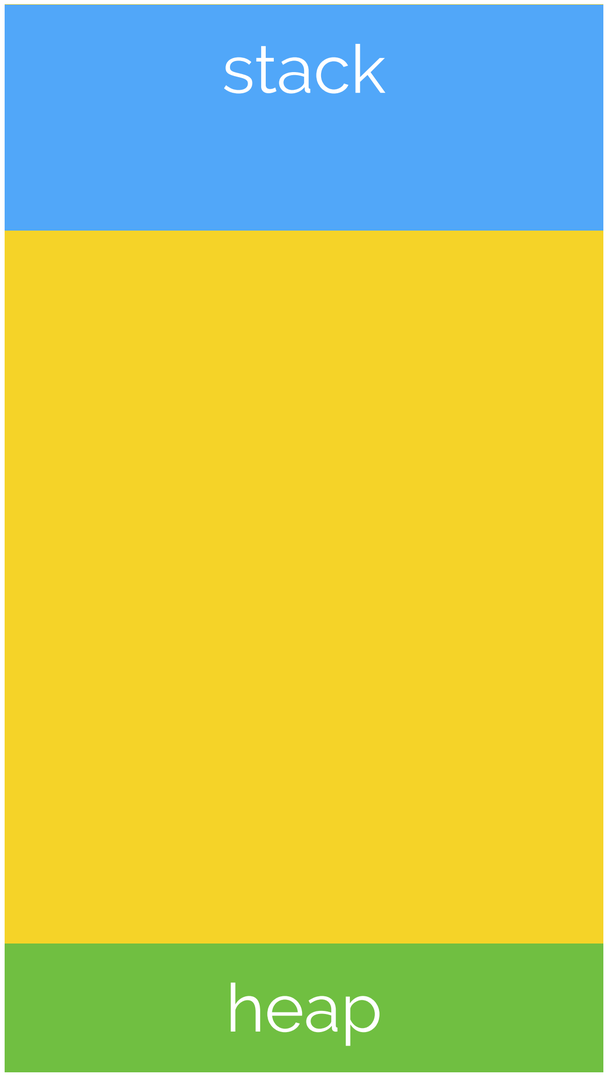
```
personA = “alice”
personB = “bob”
personC = “cindy”
personD = “dave”
print(personC)
```



SRAM (2 kb)

SRAM

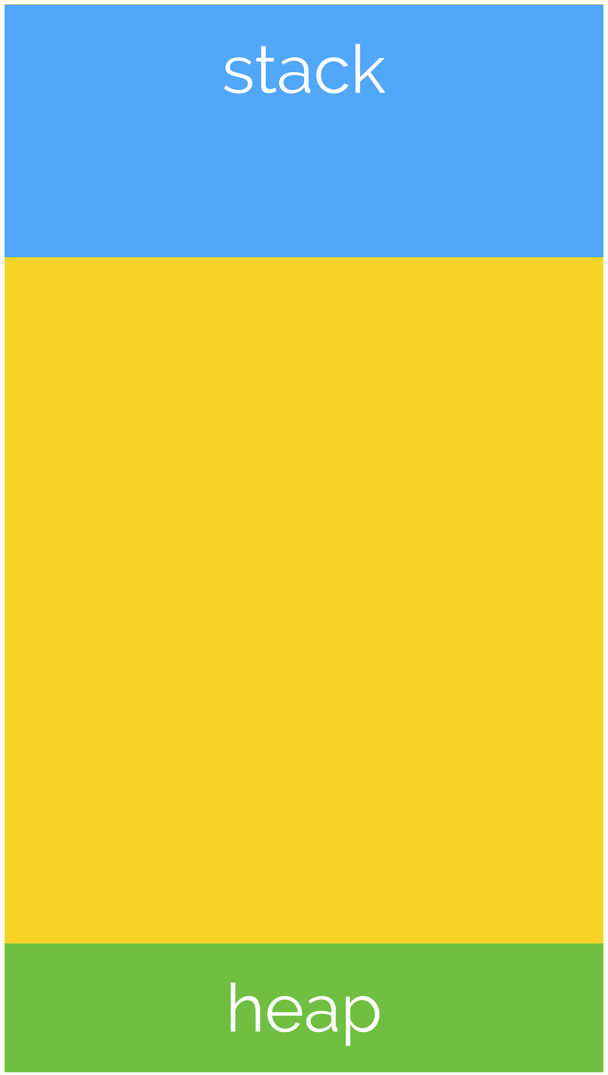
```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

SRAM

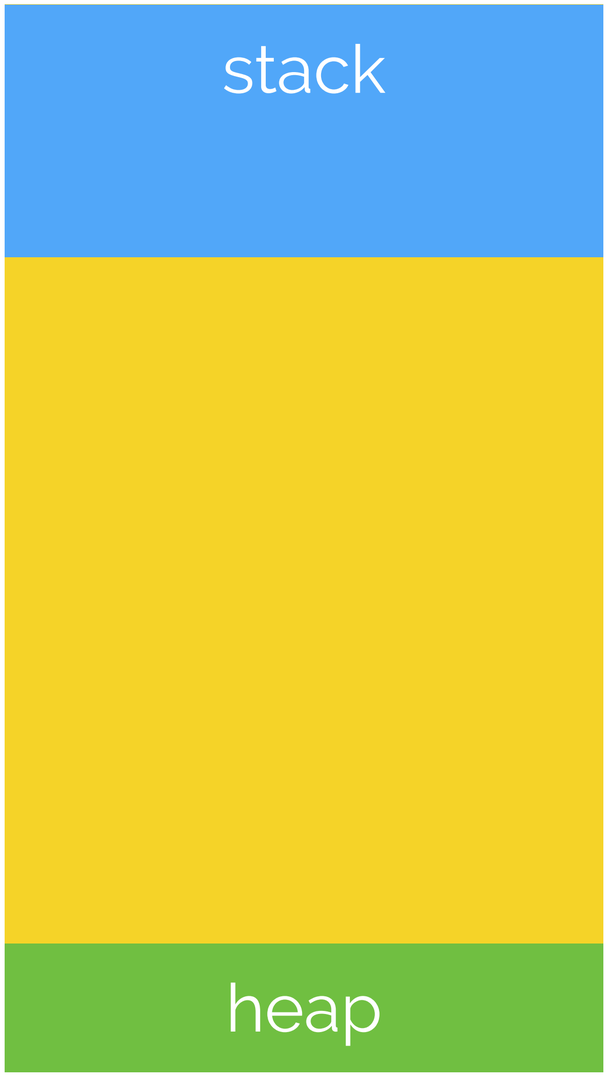
```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

SRAM

```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

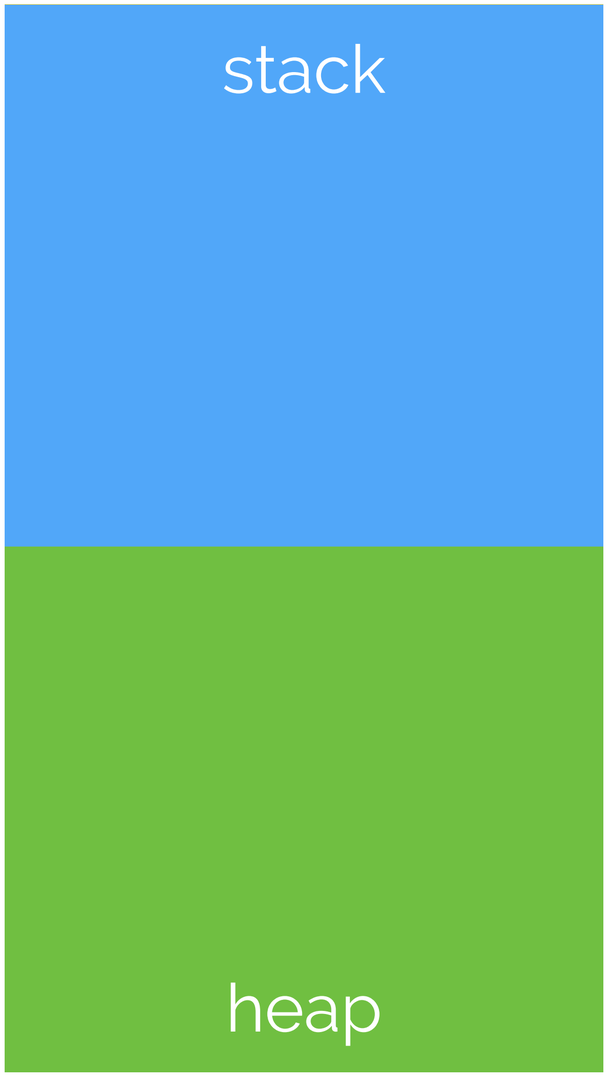
SRAM

```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```

SRAM (2 kb)

SRAM

```
personA = “alice”

personB = “bob”

personC = “cindy”

personD = “dave”

print(personC)
```



SRAM (2 kb)

Data Types

everything is a number

	boolean	byte	character	word	integer	floating point	long	unsigned long
size (bits)	8	8	8	16	16	32	32	32
min	0	0	-128	0	-32768	-3.4028235E38	-2,147,483,648	0
max	1	255	127	65535	32767	-3.4028235E38	2,147,483,647	4,294,967,295
name	`boolean`	`byte`	`char`	`word`	`int`	`float`	`long`	`unsigned long`

```
num = 1
for 1 to 5:
  num = num*10
  print(num)
```

What will this print?

```
num = 1
for 1 to 5:
  num = num*10
  print(num)
```

What will this print?

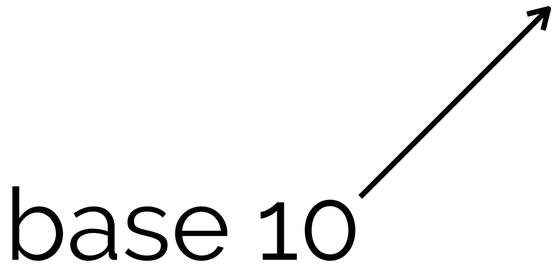
```
num = 1
for 1 to 5:
  num = num*10
  print(num)
```

```
10
100
1000
10000
100000
```

What will this print?

```
num = 1
for 1 to 5:
  num = num*10
  print(num)
```

```
10
100
1000
10000
100000
```



```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

What will this print?

```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

What will this print?

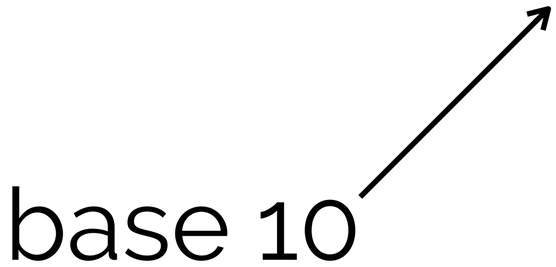
```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

```
2
4
8
16
32
```

What will this print?

```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

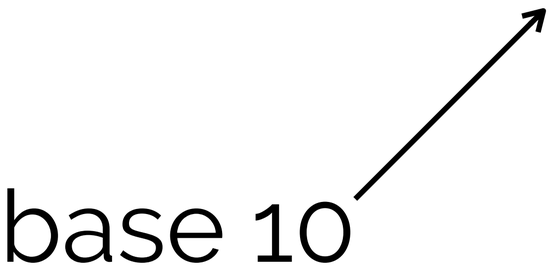
```
2
4
8
16
32
```



What will this print?

```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

```
2      10
4      100
8      1000
16     10000
32     100000
```



What will this print?

```
num = 1
for 1 to 5:
  num = num*2
  print(num)
```

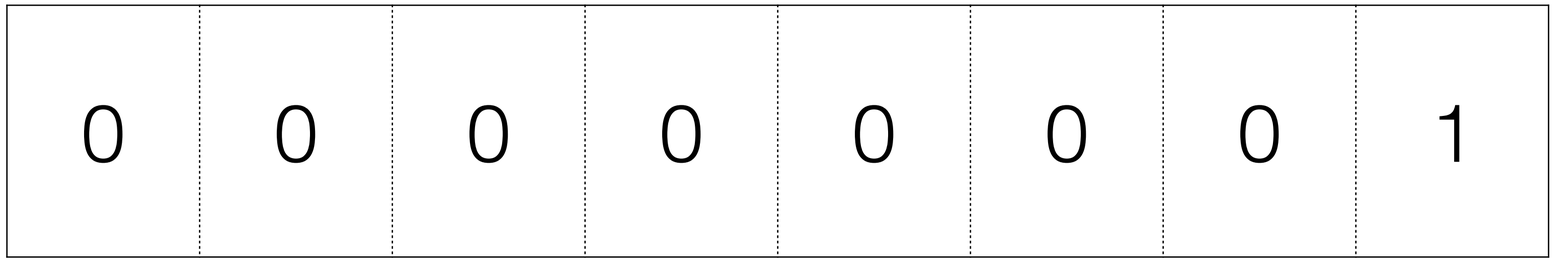
```
2      10
4      100
8      1000
16     10000
32     100000
```

base 10

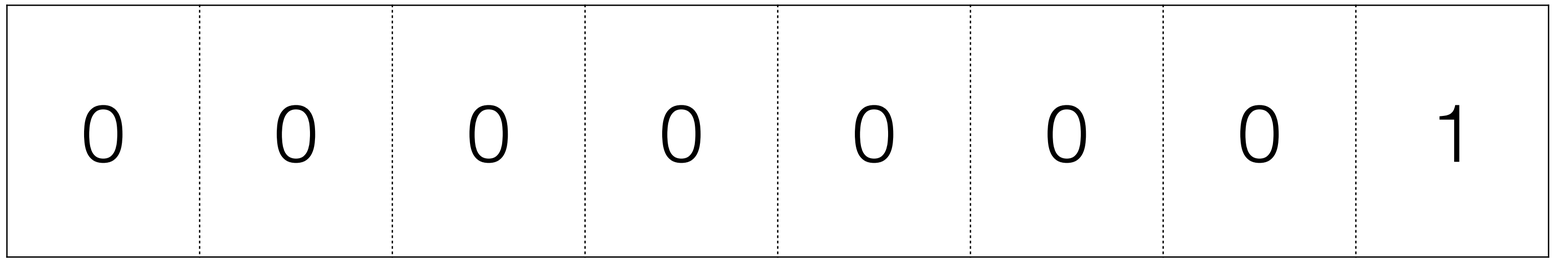
base 2

```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```

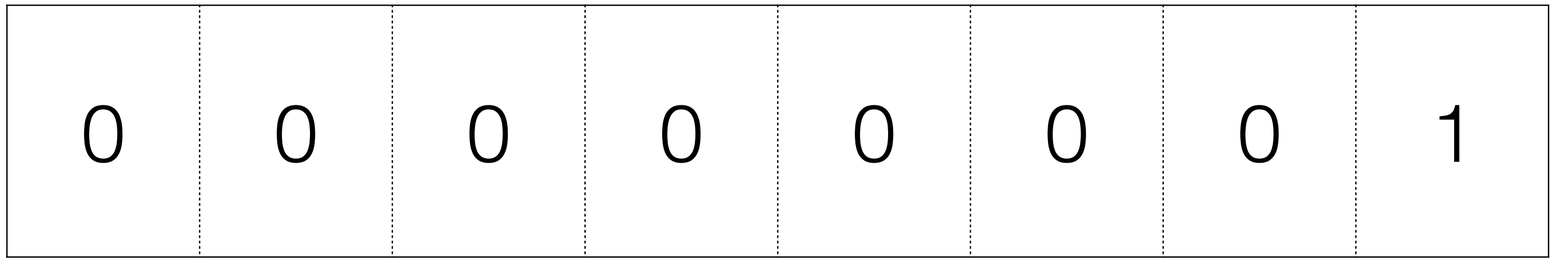
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



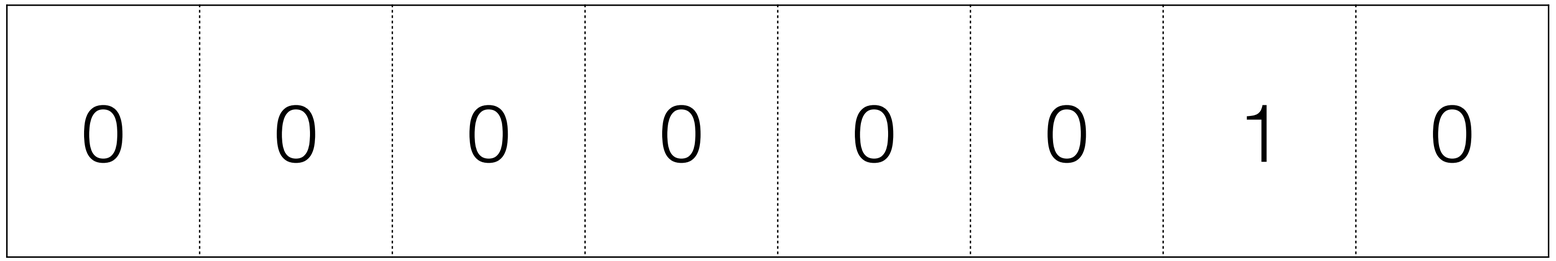
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



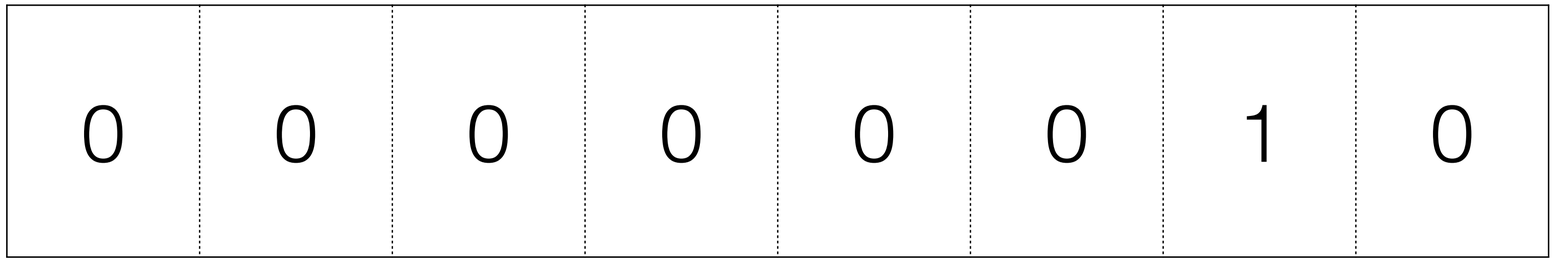
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



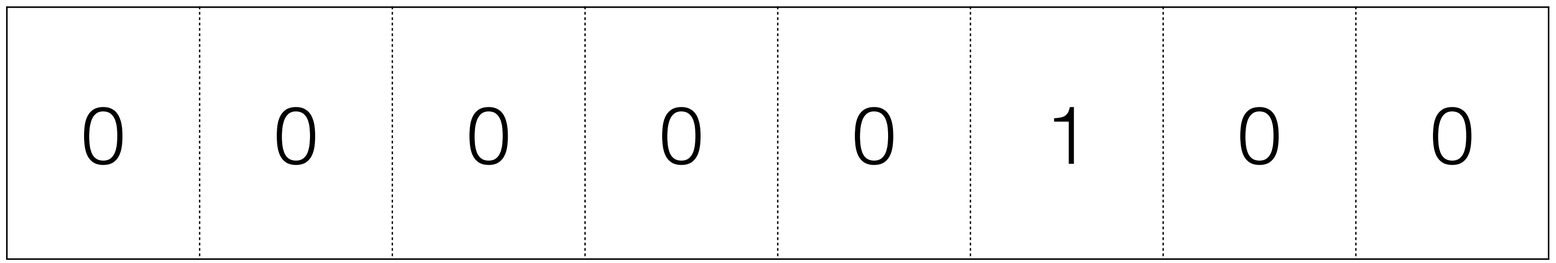
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



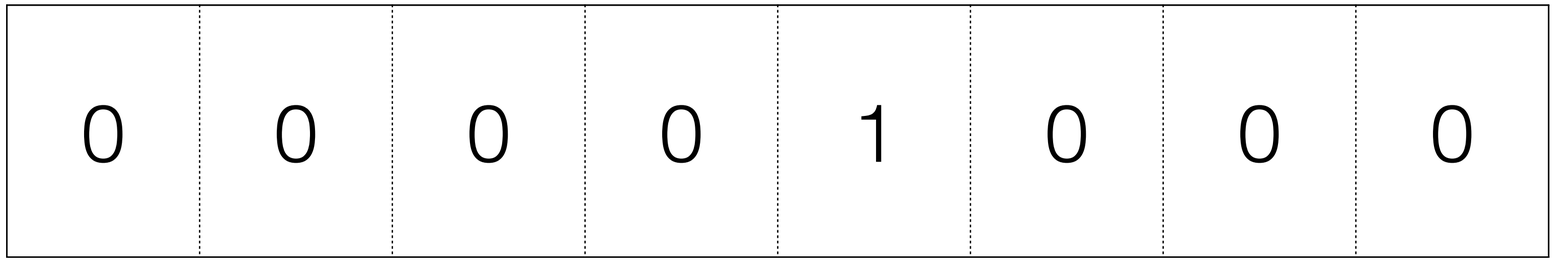
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



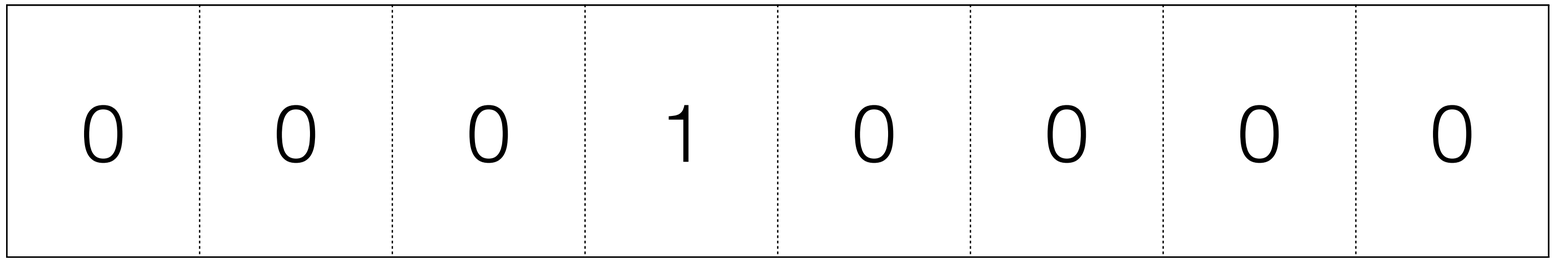
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



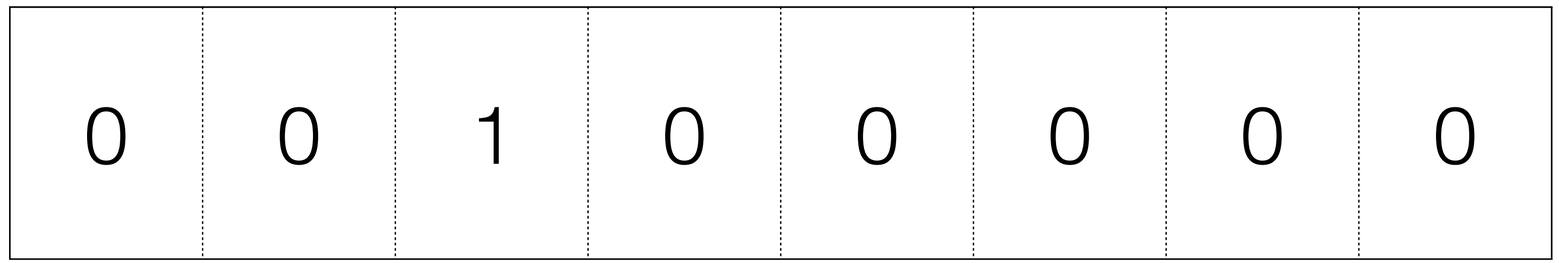
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



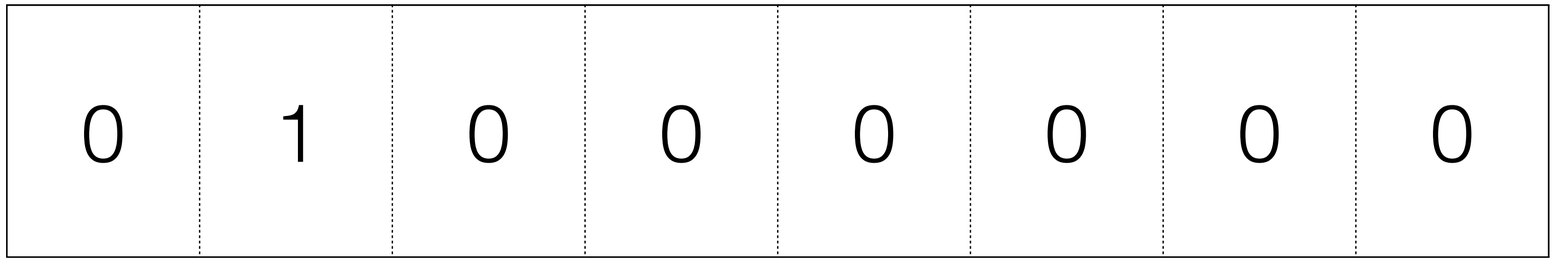
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



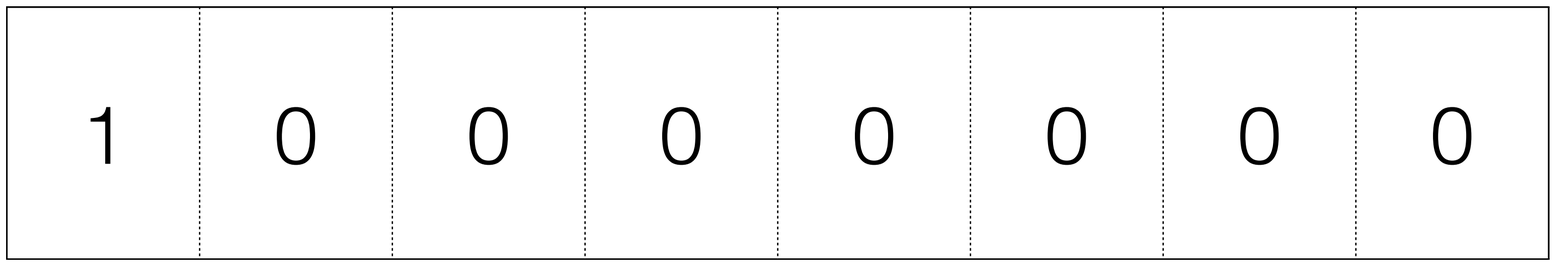
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



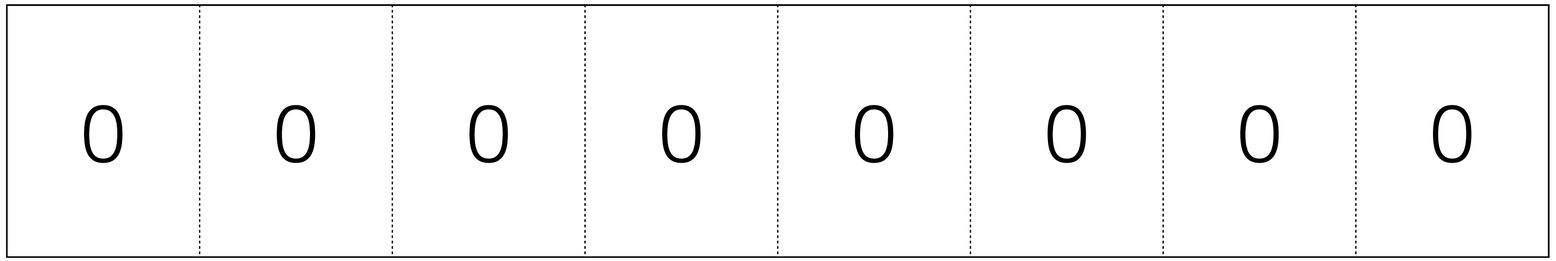
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



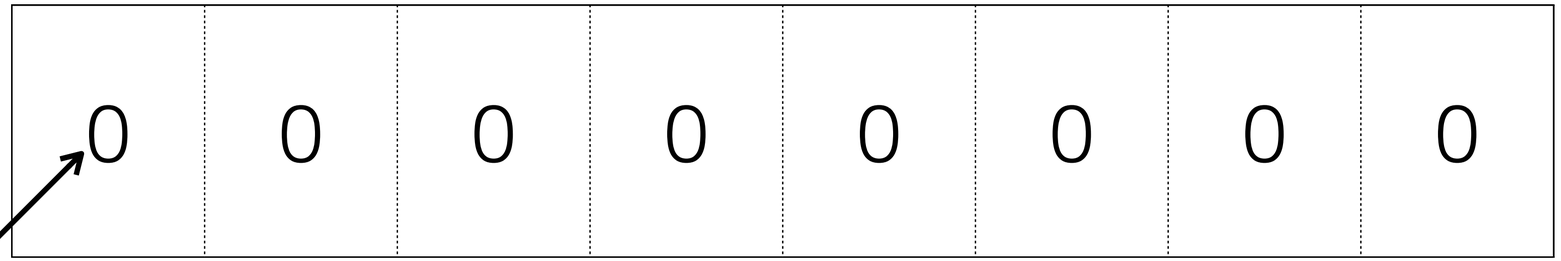
```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```



```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```

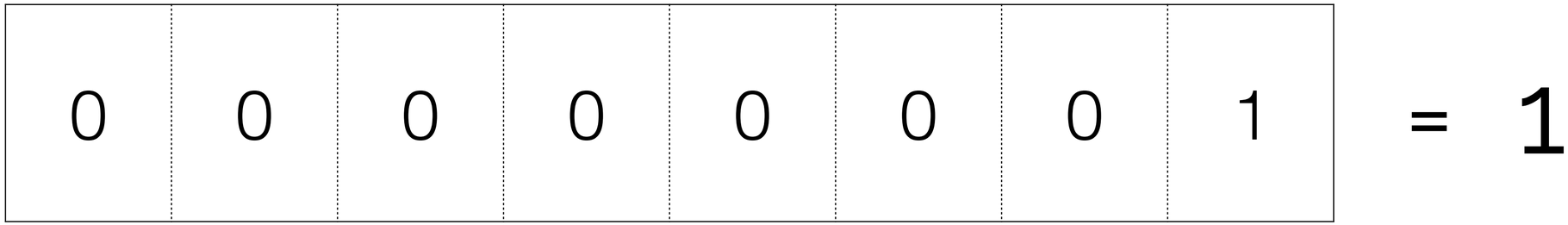


```
byte num = 1;
int i;
for(i=0; i<8; i++){
  num = num*2;
  Serial.println(num);
}
```

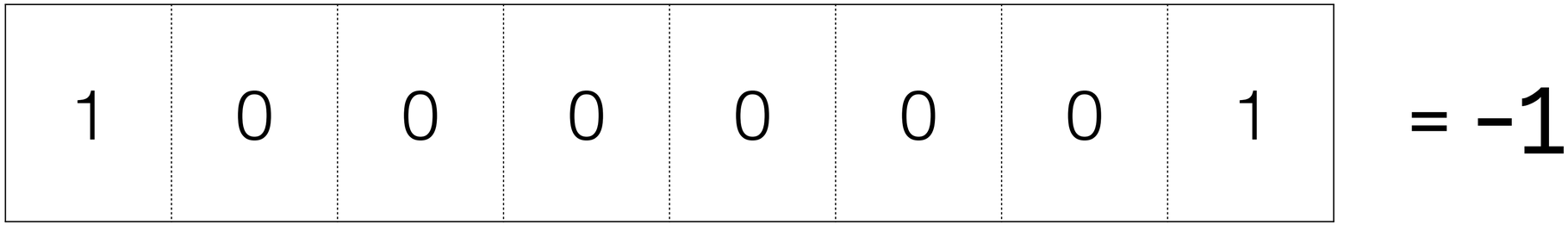


	boolean	byte	character	word	integer	floating point	long	unsigned long
size (bits)	8	8	8	16	16	32	32	32
min	0	0	-128	0	-32768	-3.4028235E38	-2,147,483,648	0
max	1	255	127	65535	32767	-3.4028235E38	2,147,483,647	4,294,967,295
name	`boolean`	`byte`	`char`	`word`	`int`	`float`	`long`	`unsigned long`

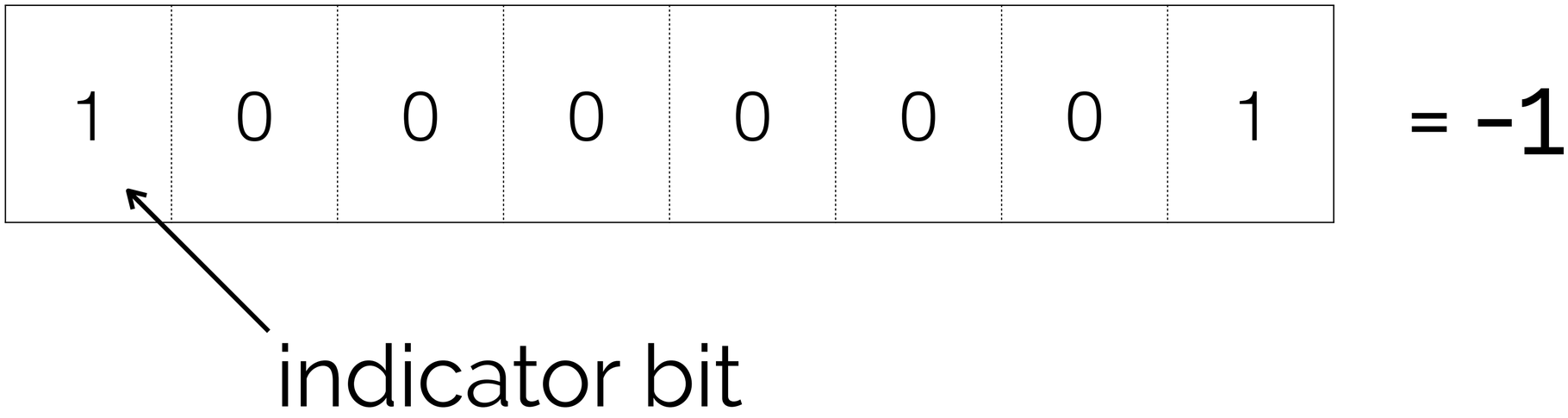
Unsigned vs. Signed Numbers



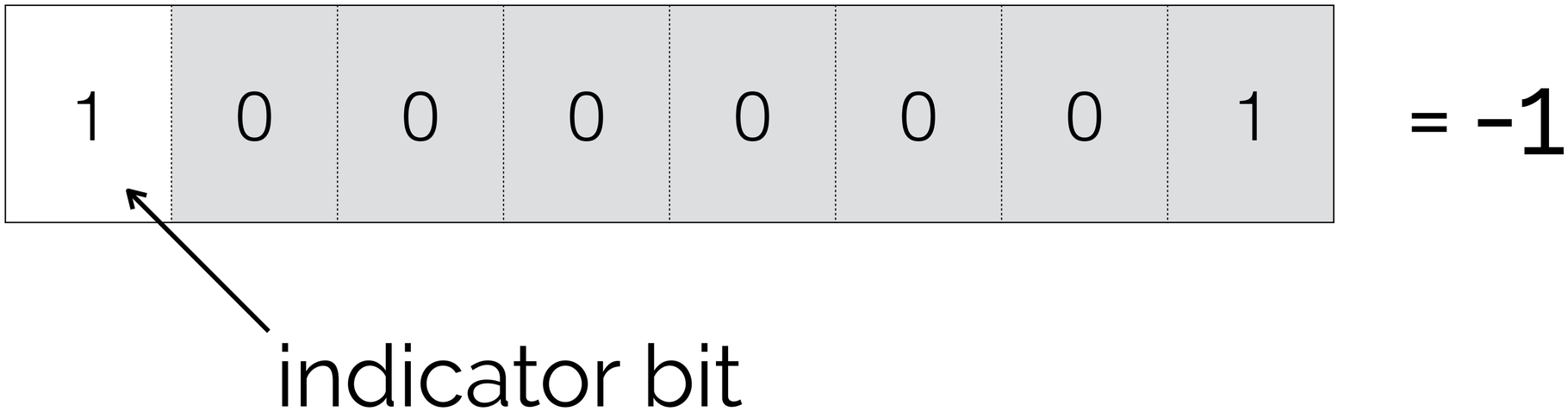
Unsigned vs. Signed Numbers



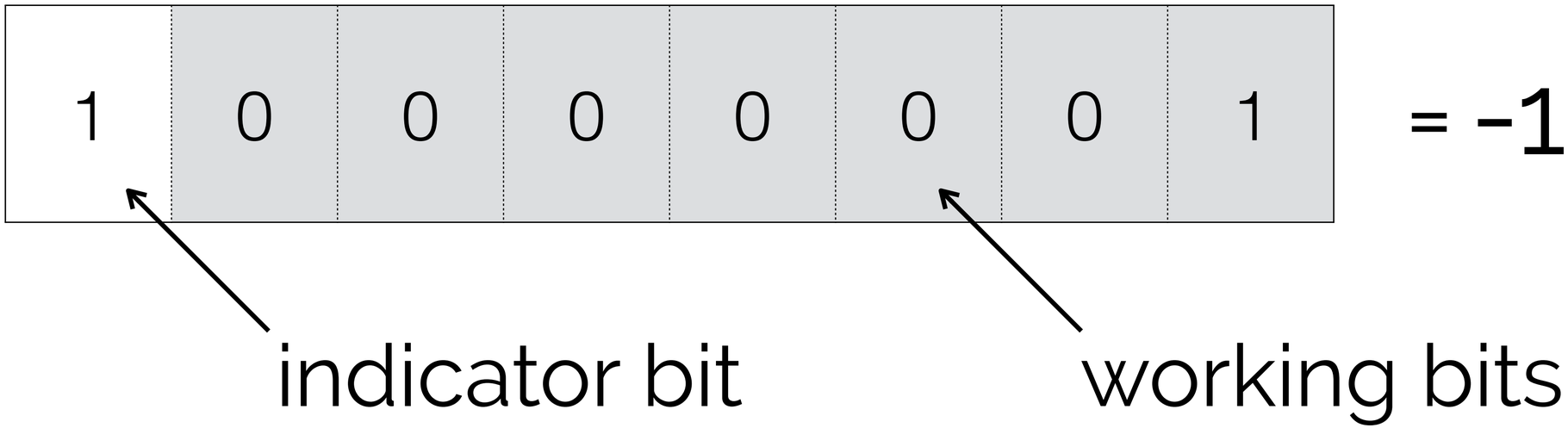
Unsigned vs. Signed Numbers



Unsigned vs. Signed Numbers



Unsigned vs. Signed Numbers



ASCII Characters

Hex	Dec	Char		Hex	Dec	Char		Hex	Dec	Char		Hex	Dec	Char	
0x00	0	NULL	null	0x20	32	Space		0x40	64	@		0x60	96	`	
0x01	1	SOH	Start of heading	0x21	33	!		0x41	65	A		0x61	97	a	
0x02	2	STX	Start of text	0x22	34	"		0x42	66	B		0x62	98	b	
0x03	3	ETX	End of text	0x23	35	#		0x43	67	C		0x63	99	c	
0x04	4	EOT	End of transmission	0x24	36	$		0x44	68	D		0x64	100	d	
0x05	5	ENQ	Enquiry	0x25	37	%		0x45	69	E		0x65	101	e	
0x06	6	ACK	Acknowledge	0x26	38	&		0x46	70	F		0x66	102	f	
0x07	7	BELL	Bell	0x27	39	'		0x47	71	G		0x67	103	g	
0x08	8	BS	Backspace	0x28	40	(		0x48	72	H		0x68	104	h	
0x09	9	TAB	Horizontal tab	0x29	41	)		0x49	73	I		0x69	105	i	
0x0A	10	LF	New line	0x2A	42	*		0x4A	74	J		0x6A	106	j	
0x0B	11	VT	Vertical tab	0x2B	43	+		0x4B	75	K		0x6B	107	k	
0x0C	12	FF	Form Feed	0x2C	44	,		0x4C	76	L		0x6C	108	l	
0x0D	13	CR	Carriage return	0x2D	45	-		0x4D	77	M		0x6D	109	m	
0x0E	14	SO	Shift out	0x2E	46	.		0x4E	78	N		0x6E	110	n	
0x0F	15	SI	Shift in	0x2F	47	/		0x4F	79	O		0x6F	111	o	
0x10	16	DLE	Data link escape	0x30	48	0		0x50	80	P		0x70	112	p	
0x11	17	DC1	Device control 1	0x31	49	1		0x51	81	Q		0x71	113	q	
0x12	18	DC2	Device control 2	0x32	50	2		0x52	82	R		0x72	114	r	
0x13	19	DC3	Device control 3	0x33	51	3		0x53	83	S		0x73	115	s	
0x14	20	DC4	Device control 4	0x34	52	4		0x54	84	T		0x74	116	t	
0x15	21	NAK	Negative ack	0x35	53	5		0x55	85	U		0x75	117	u	
0x16	22	SYN	Synchronous idle	0x36	54	6		0x56	86	V		0x76	118	v	
0x17	23	ETB	End transmission block	0x37	55	7		0x57	87	W		0x77	119	w	
0x18	24	CAN	Cancel	0x38	56	8		0x58	88	X		0x78	120	x	
0x19	25	EM	End of medium	0x39	57	9		0x59	89	Y		0x79	121	y	
0x1A	26	SUB	Substitute	0x3A	58	:		0x5A	90	Z		0x7A	122	z	
0x1B	27	FSC	Escape	0x3B	59	;		0x5B	91	[		0x7B	123	{	
0x1C	28	FS	File separator	0x3C	60	<		0x5C	92	\		0x7C	124	|	
0x1D	29	GS	Group separator	0x3D	61	=		0x5D	93	]		0x7D	125	}	
0x1E	30	RS	Record separator	0x3E	62	>		0x5E	94	^		0x7E	126	~	
0x1F	31	US	Unit separator	0x3F	63	?		0x5F	95	_		0x7F	127	DEL	

Let's Make Stuff

Numbers

Characters

Let's Make Stuff

Numbers

Characters

```
boolean myBool = false;
```

Let's Make Stuff

Numbers

Characters

```
boolean myBool = false;
int myInt = 3;
```

Let's Make Stuff

Numbers

Characters

```
boolean myBool = false;
int myInt = 3;
unsigned long myBigLong = 4133491;
```

Let's Make Stuff

Numbers

Characters

```
boolean myBool = false;
int myInt = 3;
unsigned long myBigLong = 4133491;
byte myByte = 300;
```

Let's Make Stuff

Numbers

Characters

```
boolean myBool = false;
int myInt = 3;
unsigned long myBigLong = 4133491;
byte myByte = 300;
word myWord = -13;
```

Let's Make Stuff

Numbers

```
boolean myBool = false;
int myInt = 3;
unsigned long myBigLong = 4133491;
byte myByte = 300;
word myWord = -13;
```

Characters

```
char myChar = ‘a’;
```

Let's Make Stuff

Numbers

```
boolean myBool = false;
int myInt = 3;
unsigned long myBigLong = 4133491;
byte myByte = 300;
word myWord = -13;
```

Characters

```
char myChar = ‘a’;
char myChar2 = 97;
```

Values vs. Pointers

```
house myHouse = new_house();
```

myHouse

&myHouse

Values vs. Pointers

```
house myHouse = new_house();
```

myHouse

&myHouse

```
enter(myHouse);
```

Values vs. Pointers

```
house myHouse = new_house();
```

myHouse

```
enter(myHouse);
```

&myHouse

```
enter(&myHouse);
```

Values vs. Pointers

```
house myHouse = new_house();
```

myHouse

&myHouse

```
enter(myHouse);
copy_paste(myHouse);
```

```
enter(&myHouse);
```

Values vs. Pointers

```
house myHouse = new_house();
```

myHouse

&myHouse

```
enter(myHouse);
copy_paste(myHouse);
```

```
enter(&myHouse);
copy_paste(&myHouse);
```

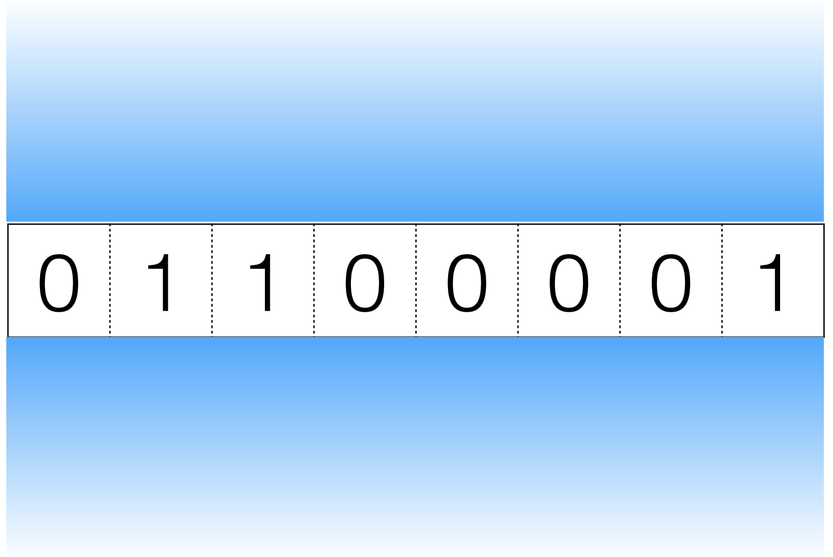
Putting Things Together

Putting Things Together

```
char myChar = ‘a’;
```

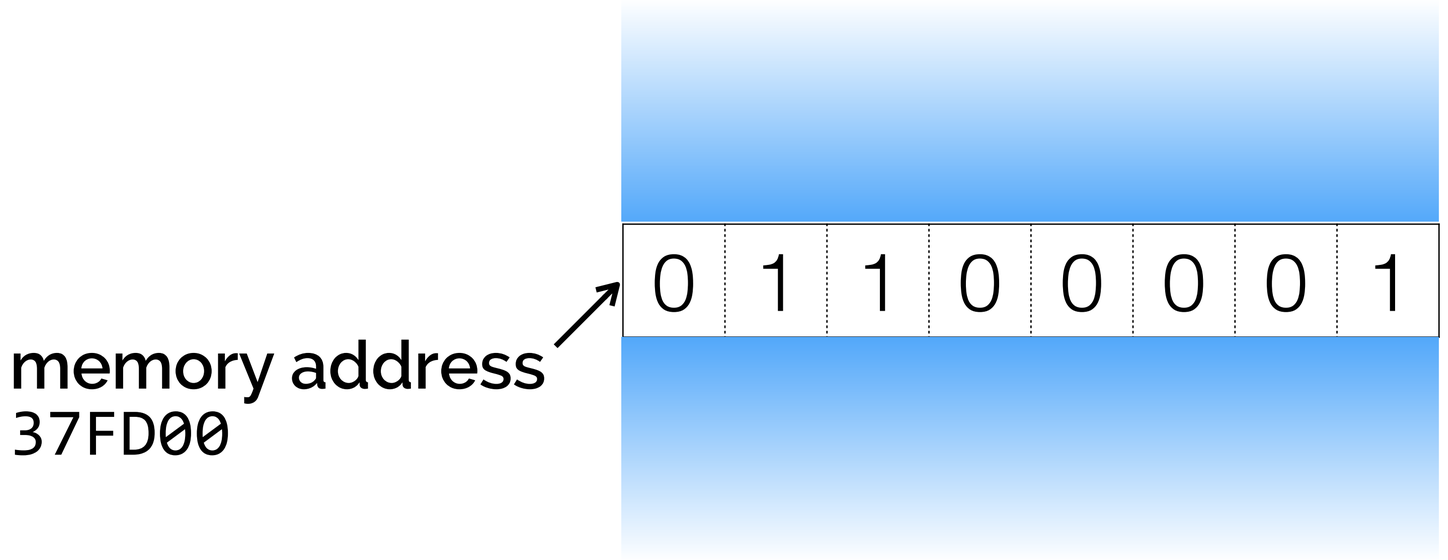
Putting Things Together

```
char myChar = ‘a’;
```



Putting Things Together

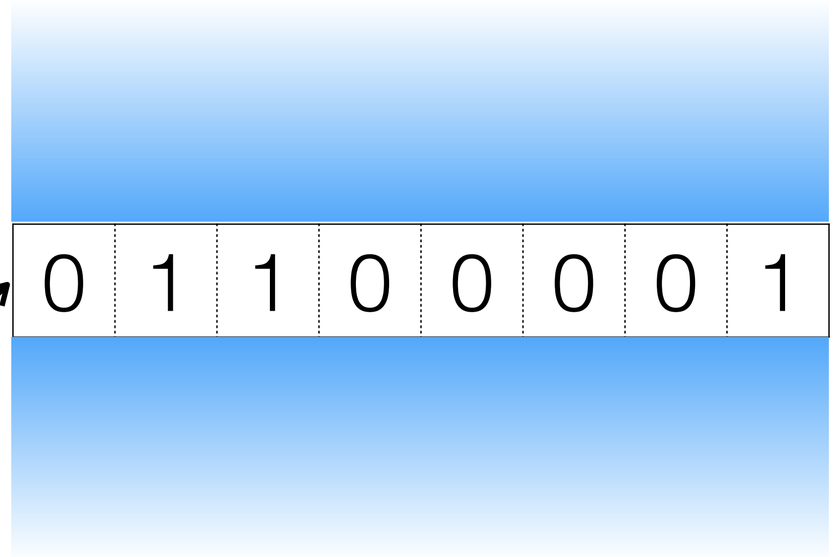
```
char myChar = ‘a’;
```



Putting Things Together

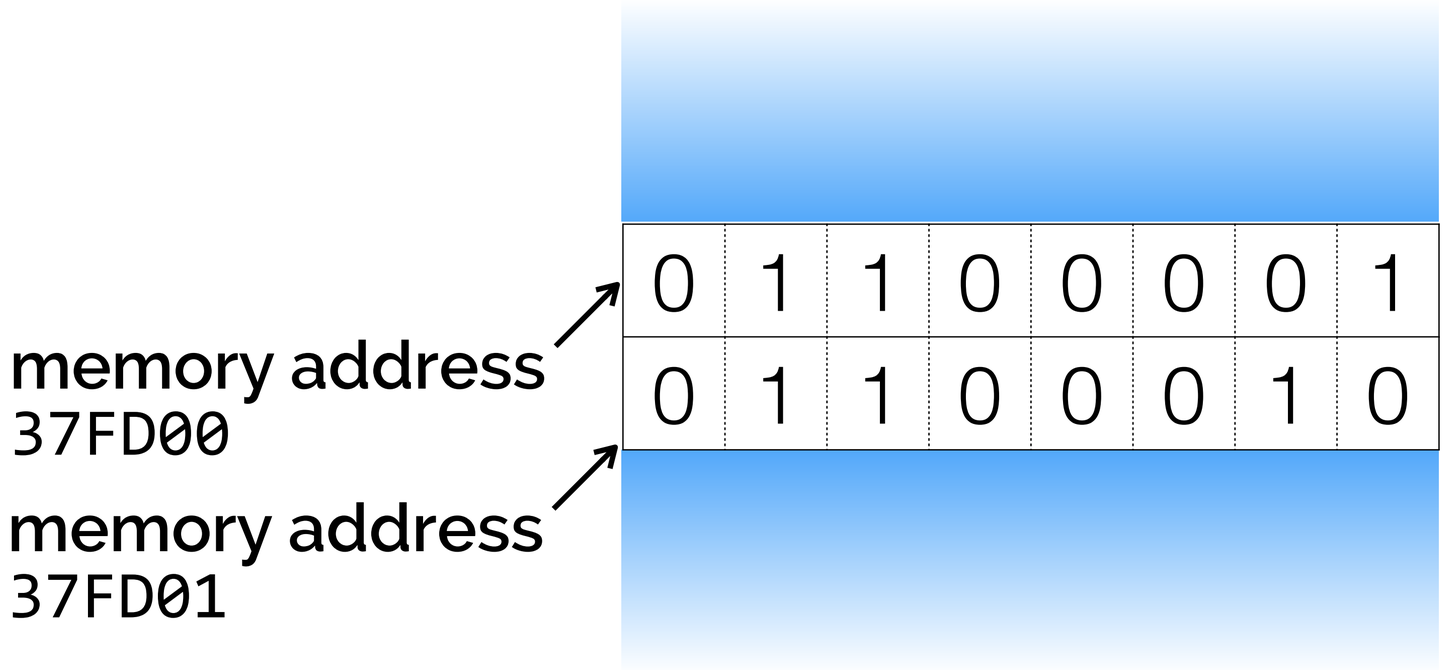
```
char myChar = ‘a’;
char myChar2 = ‘b’;
```

memory address
37FD00



Putting Things Together

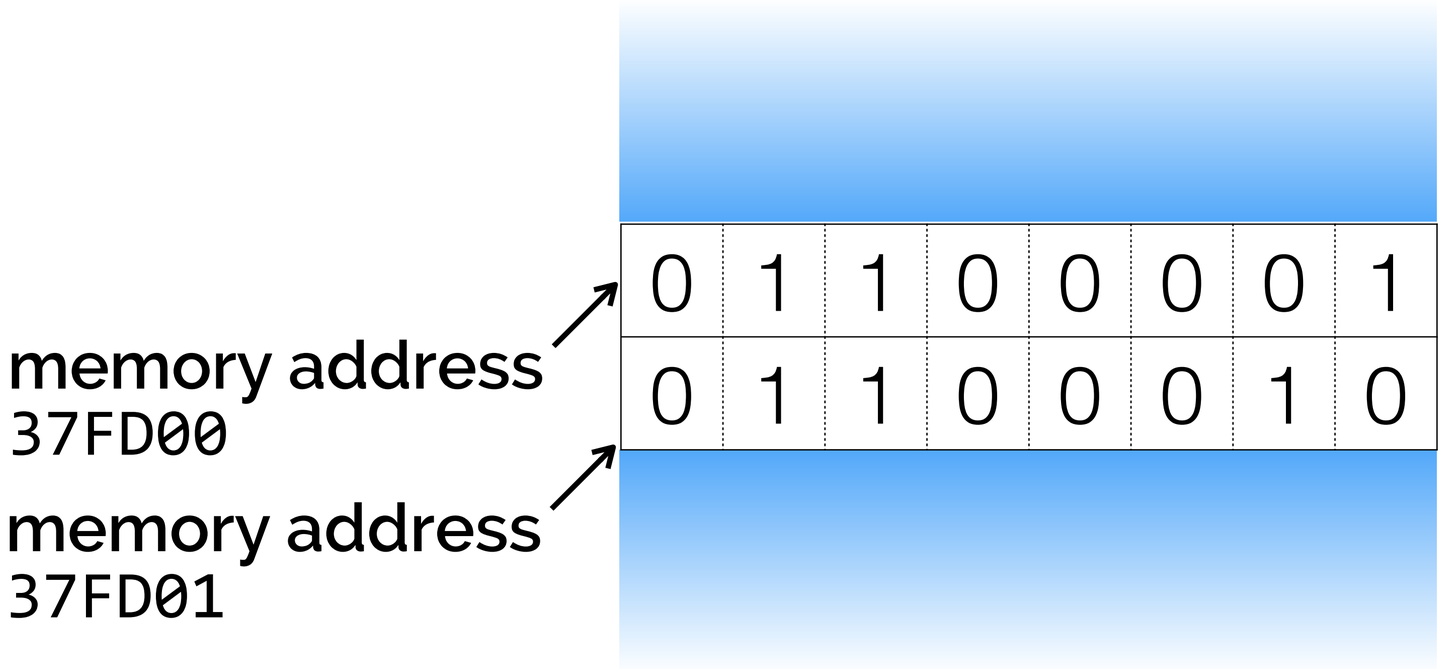
```
char myChar = ‘a’;

char myChar2 = ‘b’;
```



Putting Things Together

```
char myStr[2] = ‘ab’;
```

```
myStr == 37FD00
myStr[0] == a
myStr[1] == b
```



Values vs. Pointers (for real)

```
int  var = 20;   /* actual variable declaration */
int  *ip;        /* pointer variable declaration */
ip = &var;  /* store address of var in pointer variable*/

char buffer[30];
sprintf(buffer, "var Pointer Address = %p\n", ip);
Serial.print(buffer);
sprintf(buffer, "var Value = %d\n", var);
Serial.print(buffer);
```

Arrays

```
char myStr[4] = ‘Alex’;

int myNums[6] = {1, 2, 3, 4, 5, 6};

int myNums2[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

myStr[3] == ‘x’

myNums[3] == 4
```

Structs

```
house myHouse;
```

```
struct house
{
   int address;
   char street[16];
   char city[16];
   char state[16];
};
```

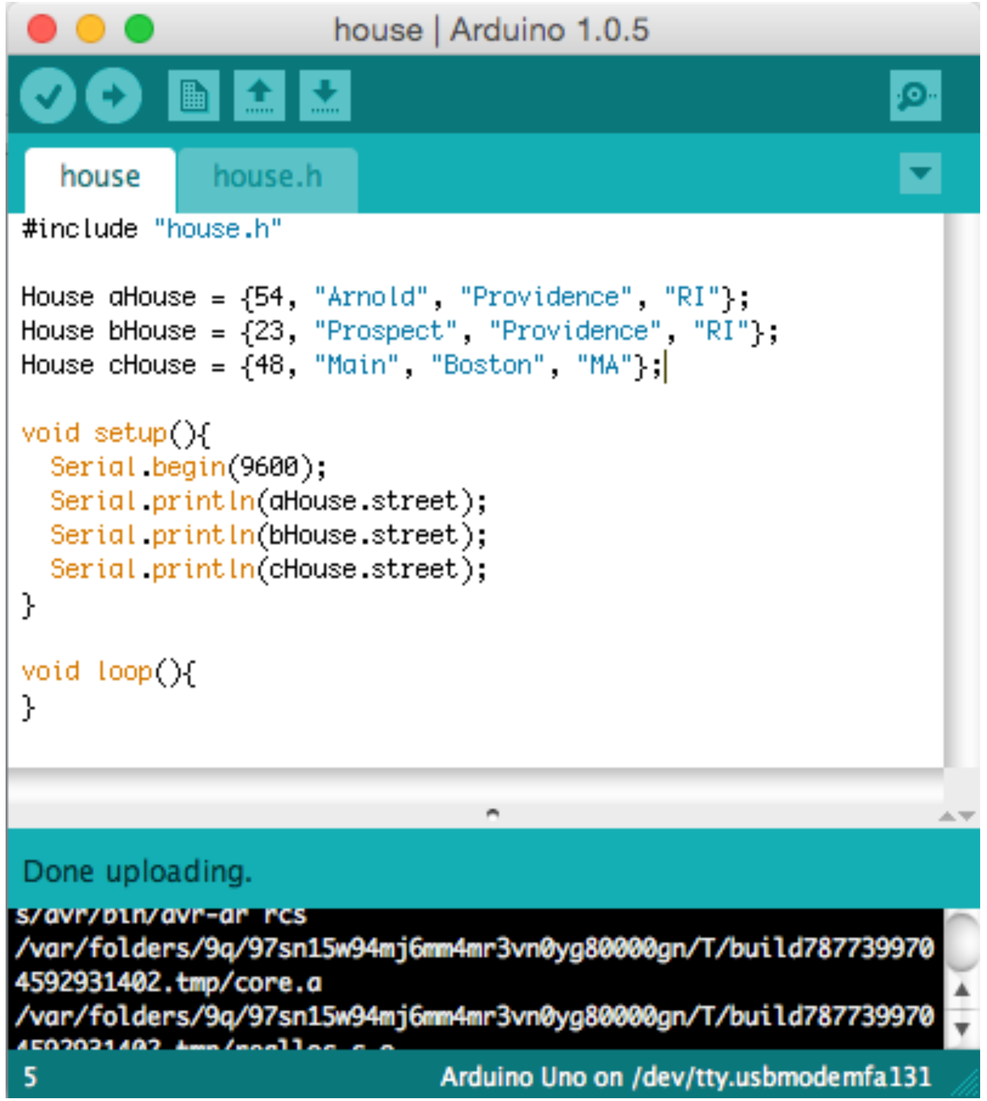
```
House aHouse;
aHouse.address = 54;
aHouse.street = “Arnold St.”
aHouse.city = “Providence”;
aHouse.state = “RI”;
```

Structs and Arduino

1. Create a new Arduino sketch and save it.
2. In a text editor, create a new file. Save it in the same directory as your Arduino sketch named “mystruct.h“.
3. Create the desired attributes of your struct.
4. Close your Arduino sketch and reopen. You should now see your .h file as a tab.
5. At the top of your sketch, include your struct with the line `#include “mystruct.h”`.

Structs and Arduino

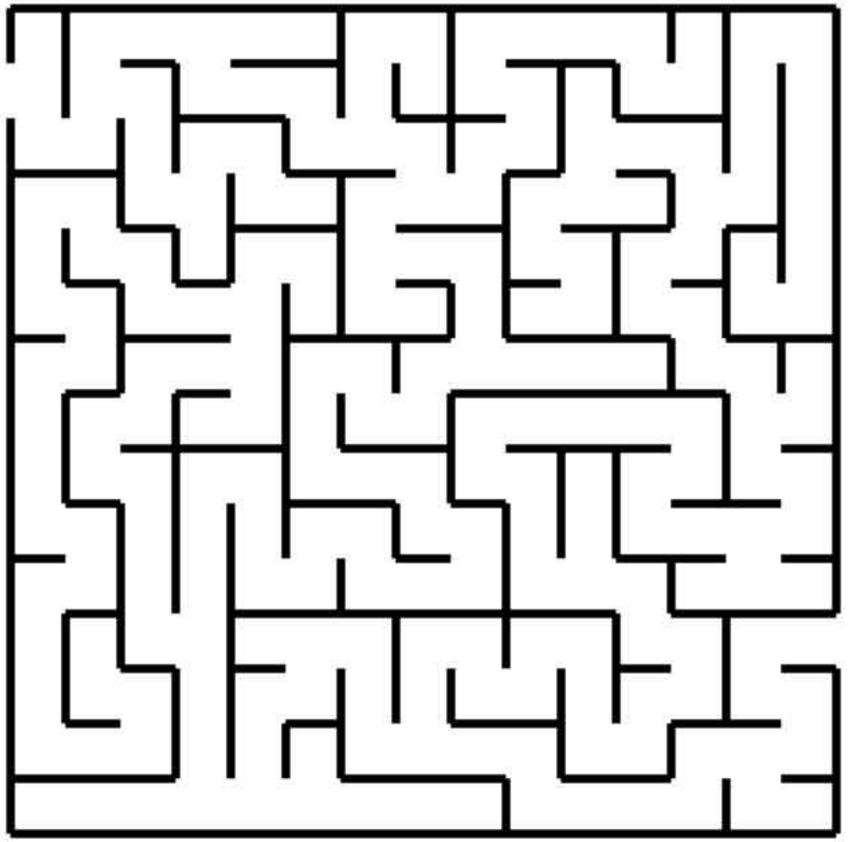
1. You can now use your struct as if it were another data type.
2. **Warning:** You must provide initial values for your struct in order to access and change them later.



Challenge

Create a new Arduino sketch that includes an data structure for a maze using a 2D array and structs.

Hint: Consider what information is needed for one cell of the maze.



Programming

Functions

```
return_type function_name( parameter list )
{
   body of the function
}


int add_numbers( int a, int b )
{
   int c = a + b;
   return c;
}
```

Functions

```
int add_vectors( int *a, int a_length, int *b, int b_length)
{
    int c = /*Add vectors here*/
    return c;
}
```

Functions

```
int *combine_vectors( int *a, int a_length, int *b, int b_length)
{
    int c[a_length+b_length] = /*Combine vectors here*/
    return c;
}
```

For Loops

```
int i;
for (i = 0; i<4; i++){
   Serial.println(i);
}
```

```
int i;
for (i = 0; i<4; i=i+3){
   Serial.println(i);
}
```

```
int i;
for (i = 0; ; i++){
   Serial.println(i);
}
```

```
int i;
for (i = 0; i<4;){
   Serial.println(i);
   i++;
}
```

While Loops

```
int i = 0;
while (i<4){
   Serial.println(i);
   i++;
}
```

```
int i = 0;
while (i != 4){
   Serial.println(i);
   i++;
}
```

Arduino Structure