

# Pointers

**address** = index of a byte in memory

**pointer** = a piece of data storing an address

				0x20	Address
				0x1C	
80	00	00	00	0x18	
				0x14	
00	00	00	FF	0x10	
				0x0C	
				0x08	
				0x04	
00	00	00	13	0x00	
0x03	0x02	0x01	0x00	byte offset (little endian order)	

# Variables as Addresses in C

```
int x; // x at 0x1C
int y; // y at 0x08
```

```
x = 1;
y = 0x30FA2B93;
```

				0x20	Address	
00	00	00	01	0x1C		X
				0x18		
				0x14		
				0x10		
				0x0C		
30	FA	2B	93	0x08		y
				0x04		
				0x00		
0x03	0x02	0x01	0x00	byte offset (little endian order)		

# Pointer Operations

**address** = index of a byte in memory

**pointer** = a piece of data storing an address

**T\* x;** declare a pointer x that will point to something of type T

**&x** address of x (get a pointer to x)

**\*x** contents at address given by pointer x (aka “dereference x” – follow the pointer)

# Pointer Example

```
int* p; // p: 0x10
```

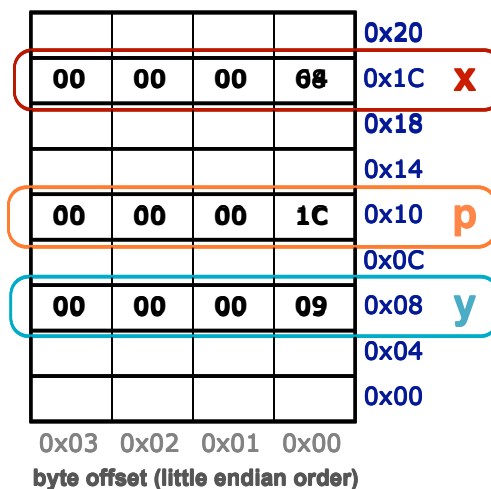
```
int x = 8; // x: 0x1C
```

```
int y = 3; // y: 0x08
```

```
p = &x;
```

```
y = 1 + *p;
```

```
*p = 100;
```



# Pointer Example

```
void do_something(int* p1, int* p2) {
    int temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}

void main() {
    int x = 5;
    int y = 3;

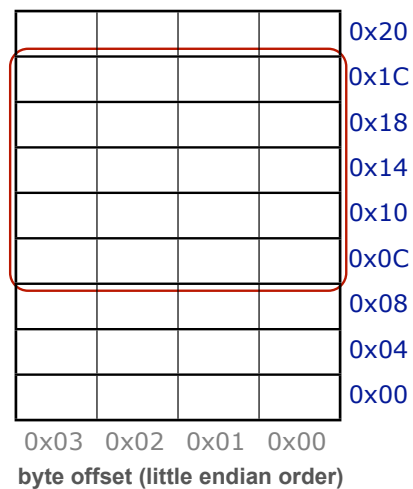
    printf(“%d %d\n”, x, y);

    do_something(&x, &y); // swap!

    printf(“%d %d\n”, x, y);
}
```

# Arrays in C

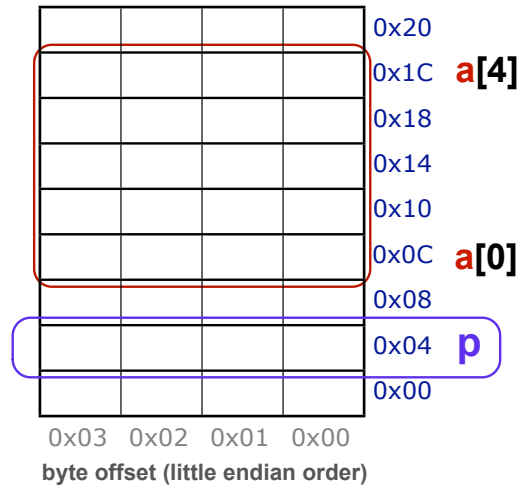
```
int a[5]; // declaration
```



# Arrays as Pointers

```
int a[5];
```

```
int* p;
```

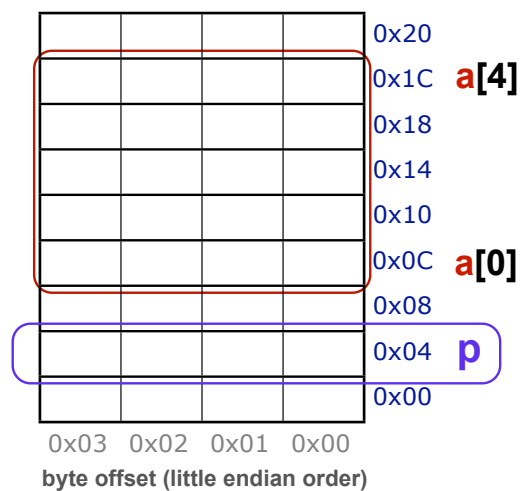


# Arrays as Pointers

```
int a[5];
```

```
int* p;
```

```
p = &a[0];
```

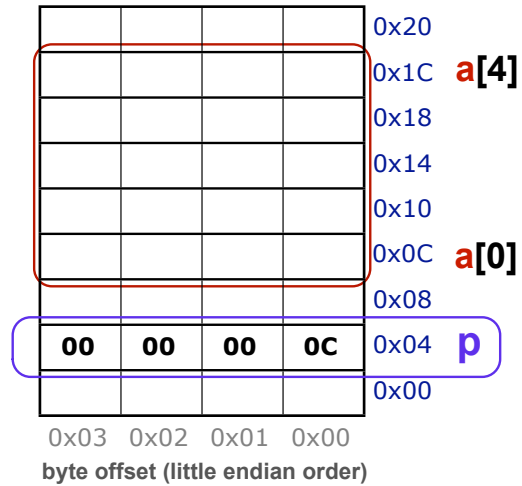


# Arrays as Pointers

```
int a[5];
```

```
int* p;
```

```
p = &a[0];
```

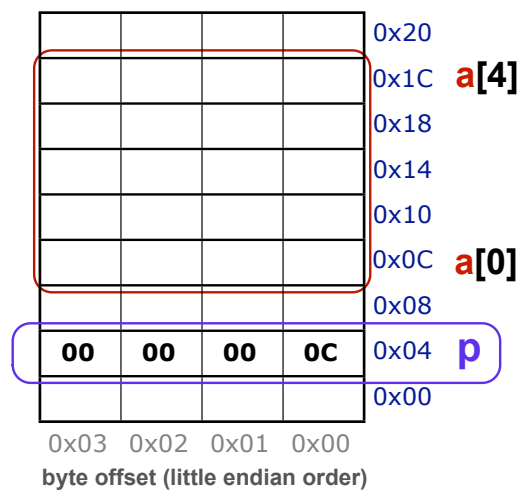


# Arrays as Pointers

```
int a[5];
```

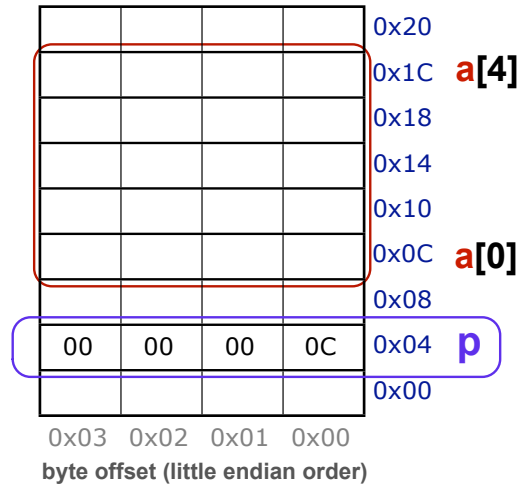
```
int* p;
```

```
p = &a[0]; // or p = a;
```



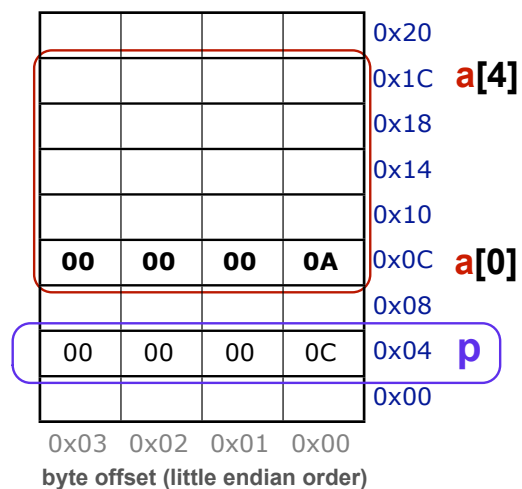
# Arrays as Pointers

```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;
```



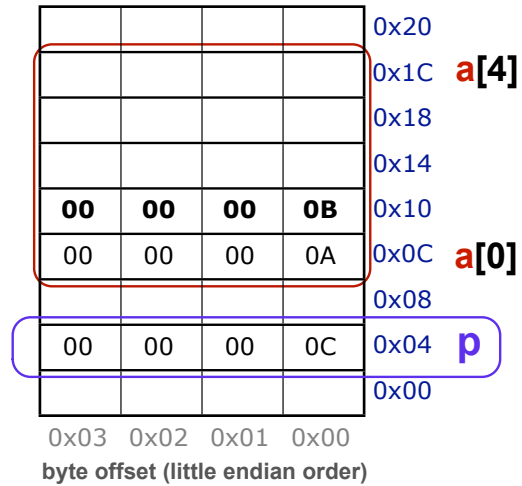
# Arrays as Pointers

```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;
```



# Arrays as Pointers

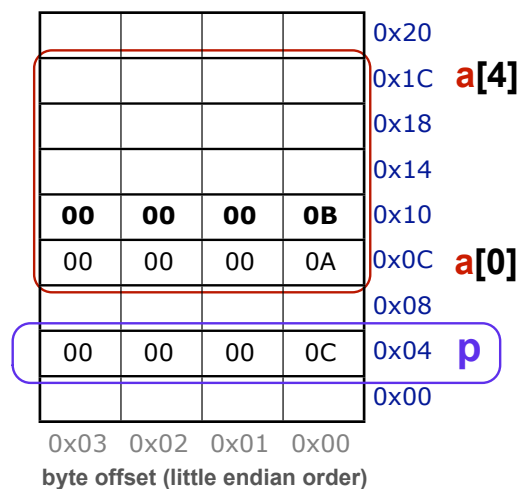
```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;  
p[1] = 0xB;
```



# Pointer Arithmetic

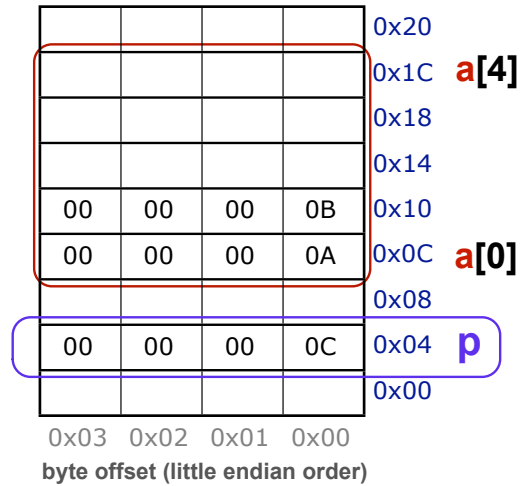
```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;  
p[1] = 0xB;  
*(p + 1) = 0xB;
```

Equivalent!



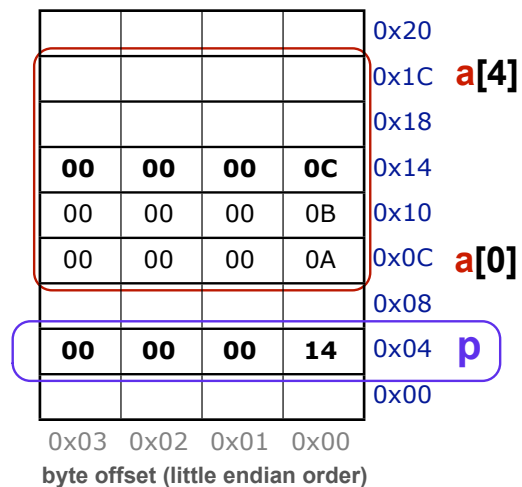
# Pointer Arithmetic

```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;  
*(p + 1) = 0xB;  
  
p = p + 2;  
*p = a[1] + 1;
```



# Pointer Arithmetic

```
int a[5];  
  
int* p;  
  
p = a; // &a[0]  
  
*p = 0xA;  
*(p + 1) = 0xB;  
  
p = p + 2;  
*p = a[1] + 1;
```





# Null-Terminated Strings

*null character*



0x43	0x53	0x43	0x49	0x20	0x32	0x33	0x33	0x30	0x00
'C'	'S'	'C'	'I'	' '	'2'	'3'	'3'	'0'	'\0'