

Pointers

Embedded Software Essentials

C2M1V3

Pointer Types [S2a]

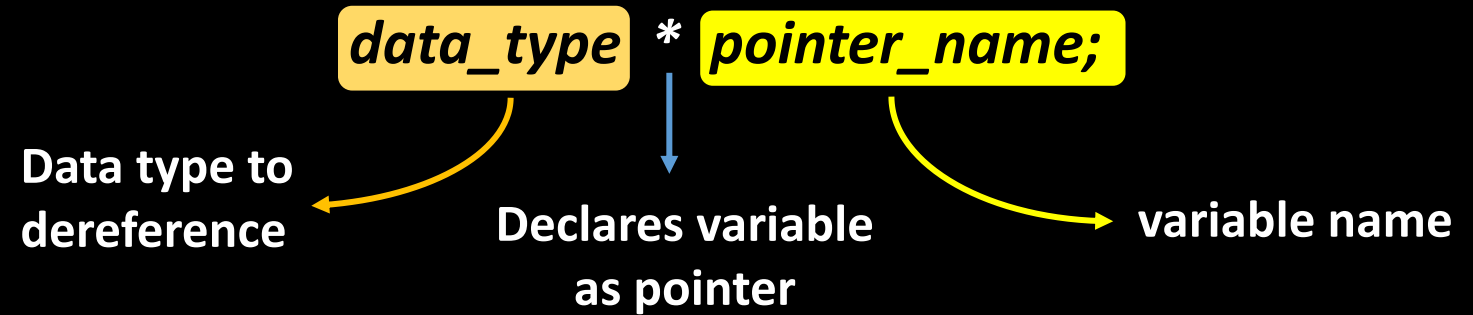
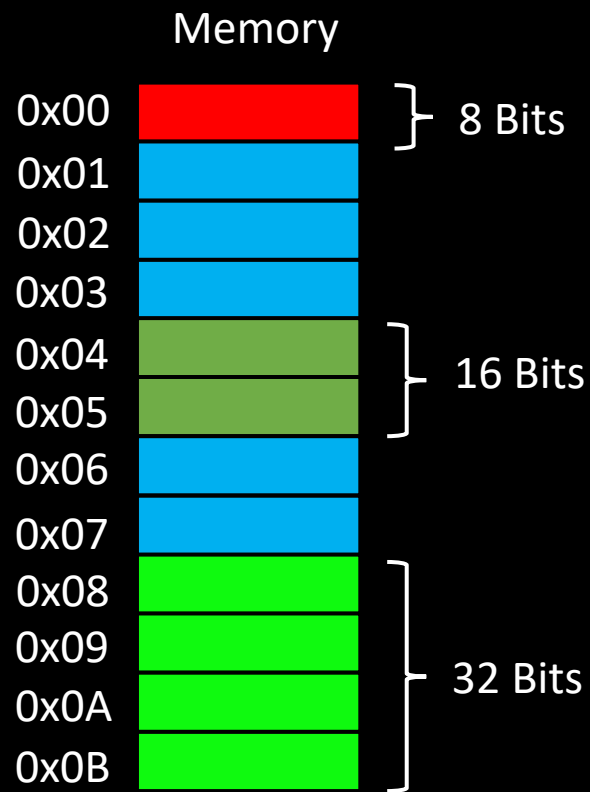
Pointer type denotes the **data type** that a pointer will *dereference*

	Memory
0x00	
0x01	
0x02	
0x03	
0x04	
0x05	
0x06	
0x07	
0x08	
0x09	
0x0A	
0x0B	

*data_type * pointer_name;*

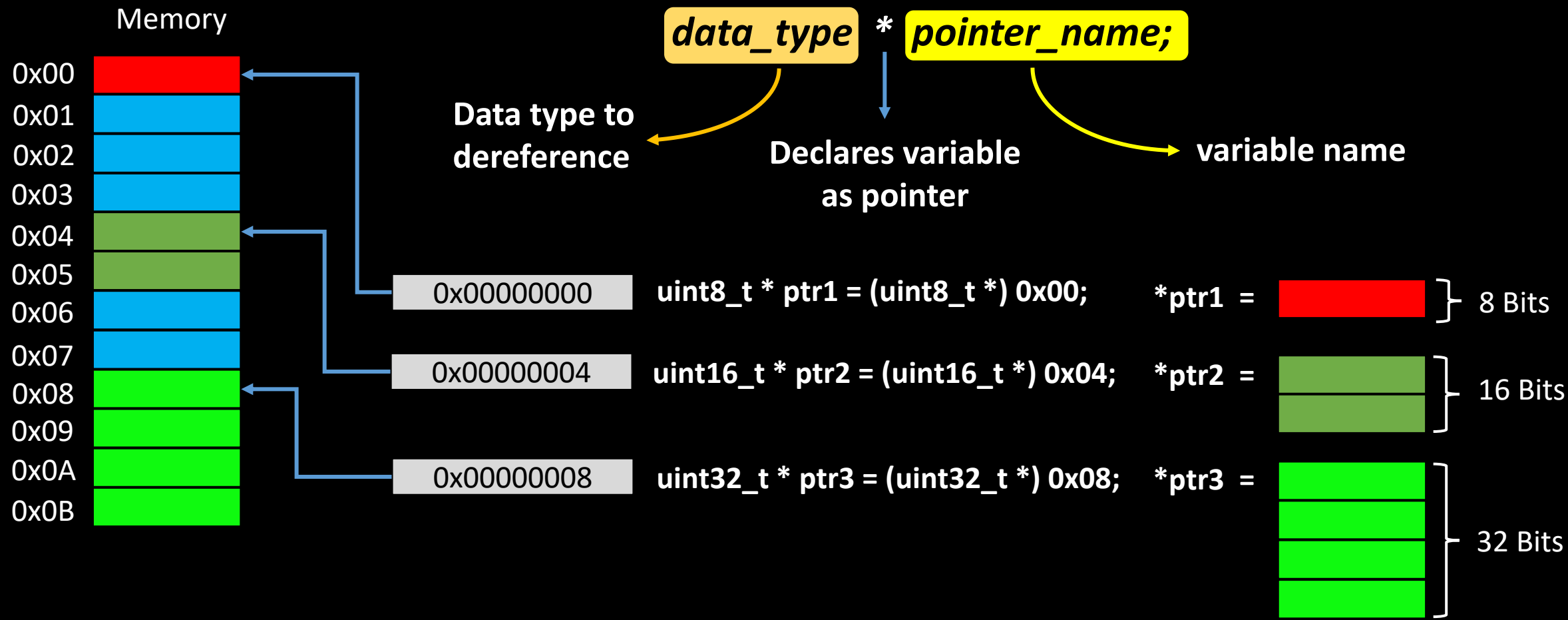
Pointer Types [S2b]

Pointer type denotes the **data type** that a pointer will *dereference*



Pointer Types [S2c]

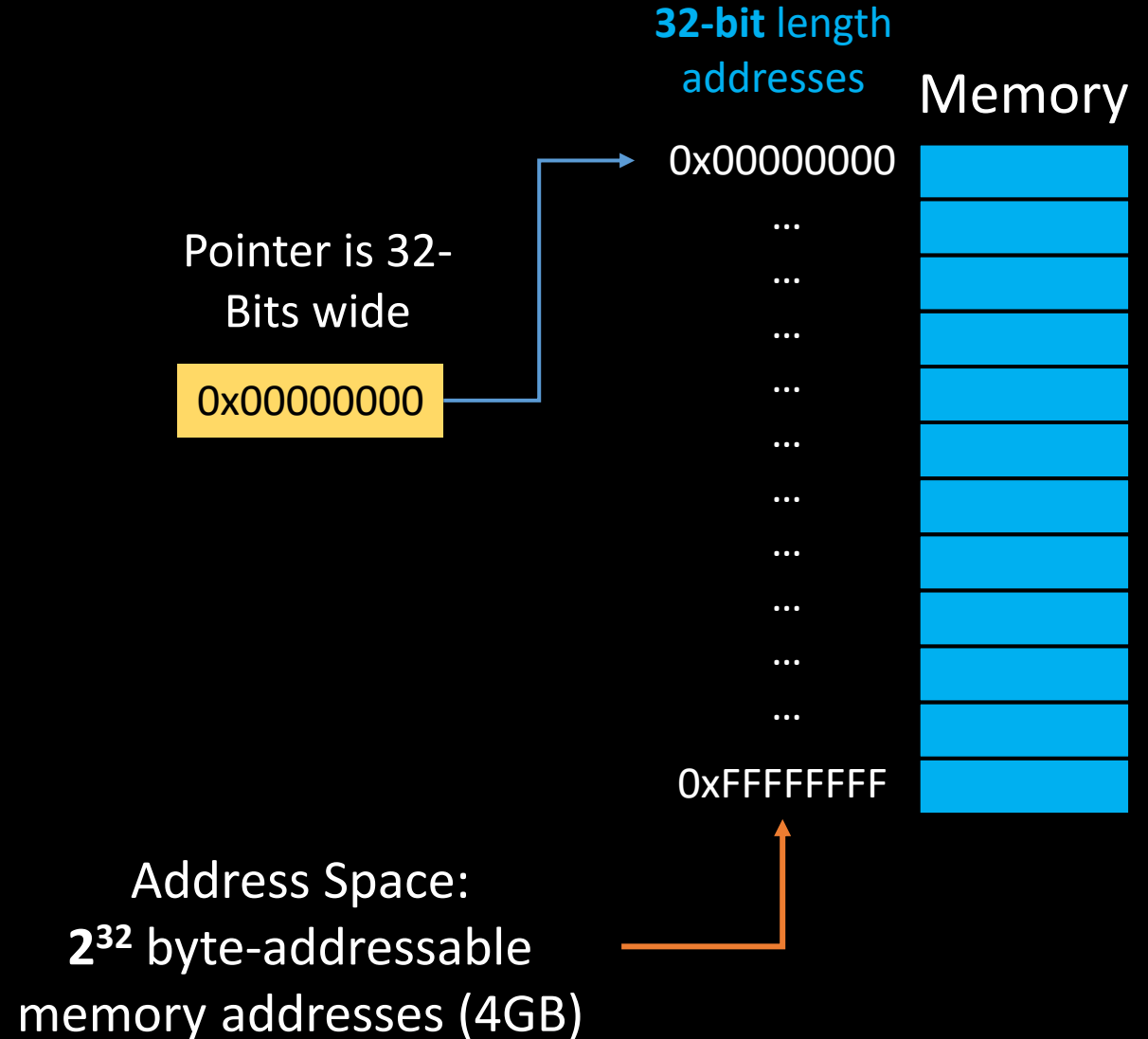
Pointer type denotes the **data type** that a pointer will *dereference*



Pointer Size [S3]

- Pointers hold **Addresses** to a location in memory
- All Pointers are the same length
 - ARM → 32-bit Pointer Length
- Address Space = $2^{\text{PointerLength}}$

32-bit ARM Architecture



Pointer Size [S4]

- All Pointers are the same length

```
uint8_t * ptr1 = (uint8_t *) 0x00;  
uint16_t * ptr2 = (uint16_t *) 0x04;  
uint32_t * ptr3 = (uint32_t *) 0x08;  
float * ptr4 = (float *) 0x0C;
```

```
sizeof(uint8_t*) = sizeof(int16_t*)  
                 = sizeof(uint32_t*)  
                 = sizeof(float*)  
                 = 32-Bits!
```

```
sizeof(ptr1) = sizeof(ptr2)  
             = sizeof(ptr3)  
             = sizeof(ptr4)  
             = 32-Bits!
```

- Pointers Dereference different sized data

```
sizeof(*ptr1)  ≠ sizeof(*ptr2)  ≠ sizeof(*ptr3)  ≠ sizeof(*ptr4)  
    1 Byte      2 Byte          4 Byte          4 Byte
```

Pointer Operators [S5]

- Dereference Operator = `*`
 - Accesses data at address
- Address-of Operator = `&`
 - Provides address of variable
- Integers are not addresses
- Cast to explicit address for Peripheral Memory

```
uint32_t var;
```

```
uint32_t * ptr = &var;
```

```
*ptr = 0xABCD1234;
```

```
uint16_t * ptr = (uint16_t *) 0x480C0000;
```



Casts Integer to address

Null Pointers [S6]

- At time of pointer declaration, you might not know the address
 - Use a NULL Pointer for Initialization
- Null Pointers point to nothing
 - Used to check for valid pointer
- Dereferencing a NULL Pointer can cause an exception

This pointer will have garbage data

```
uint32_t * ptr;
```


Using un-initialized pointer can potentially corrupt your memory

```
#define NULL ((void*)0)  
uint32_t * ptr = NULL;
```

```
if (ptr == NULL) {  
    /* error! */  
}  
*ptr = 0xABCD1234;
```


Pointer Example [S7a]


```
typedef struct foo {  
    uint8_t  varA;  
    uint8_t  varB;  
    uint16_t varC;  
} foo_t;
```



At least
32 Bits

```
foo_t varS;  
uint8_t Num;
```

```
foo_t * varS_ptr = &varS;  
uint8_t * ptr_Num = &num;
```



Each Pointer is 32-bits

Pointer Example [S7b]

```
typedef struct foo {  
    uint8_t  varA;  
    uint8_t  varB;  
    uint16_t varC;  
} foo_t;
```

```
foo_t varS;  
uint8_t Num;
```

```
foo_t * varS_ptr = &varS;  
uint8_t * ptr_Num = &num;
```

varS_ptr->varA → Dereferences 8-bits
varS_ptr->varB → Dereferences 8-bits
varS_ptr->varC → Dereferences 16-bits

*ptr_Num → Dereferences 8-bits

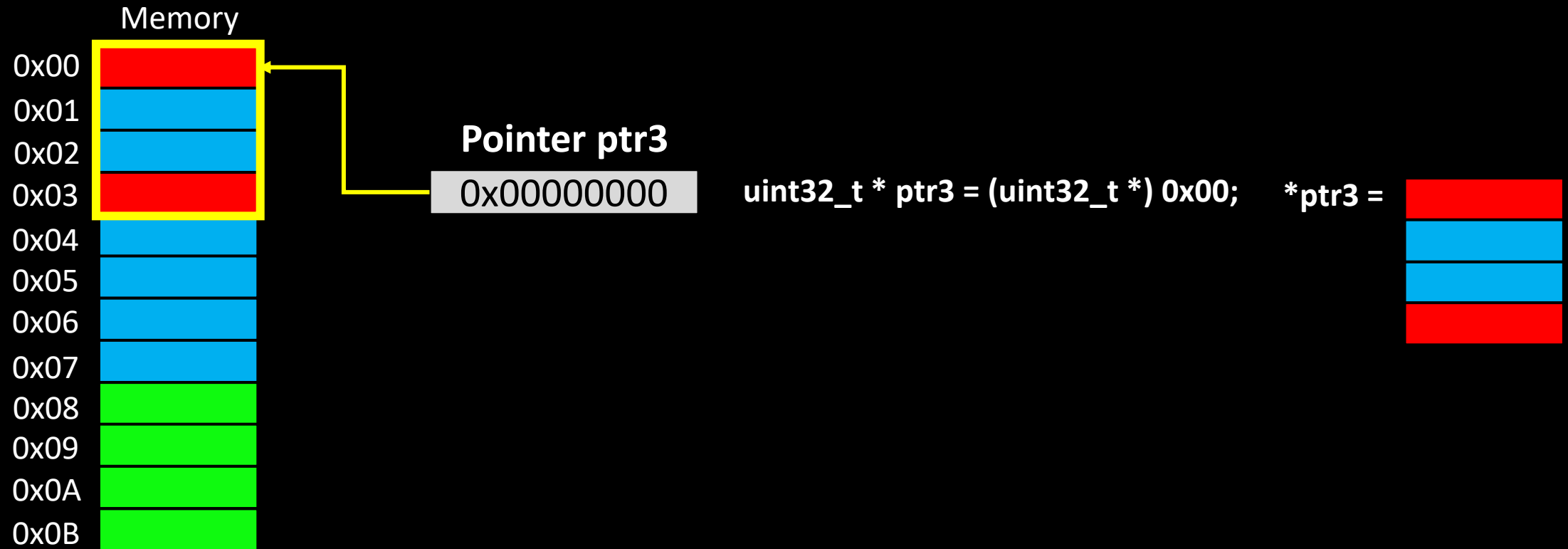
varS_ptr->varC



Structure Pointer Dereference Operator

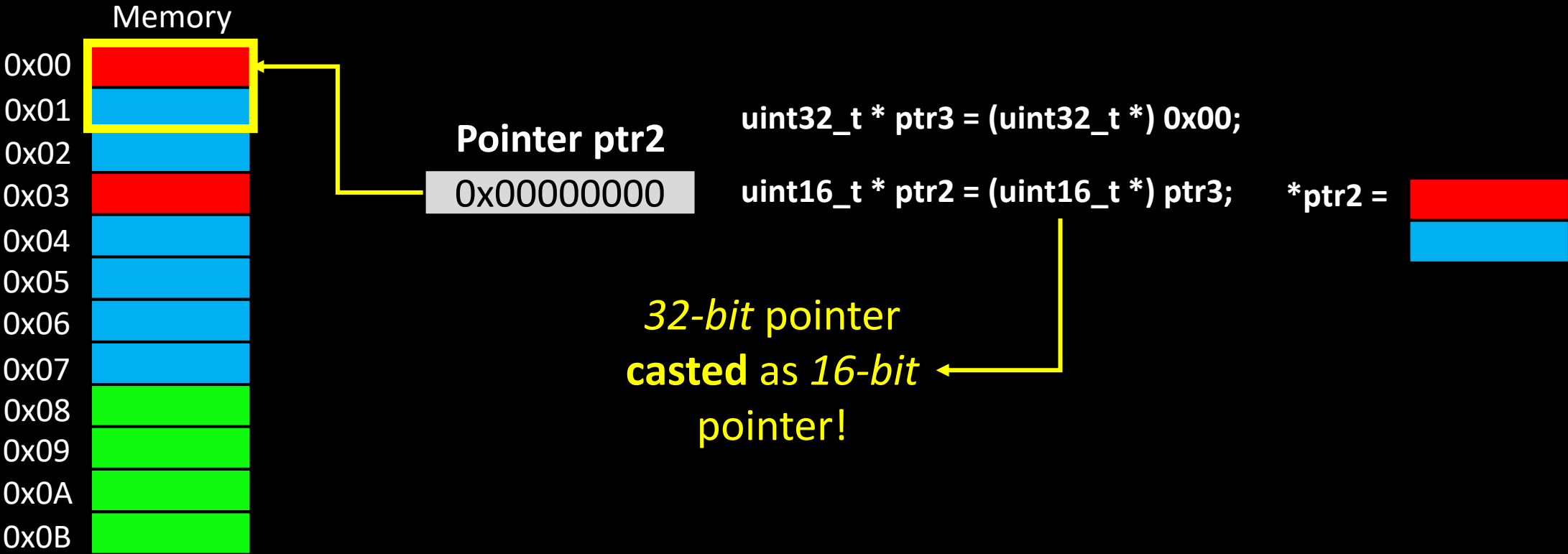
Pointer Casting [S8a]

Cast pointers to dereference **different sizes** from **same address**



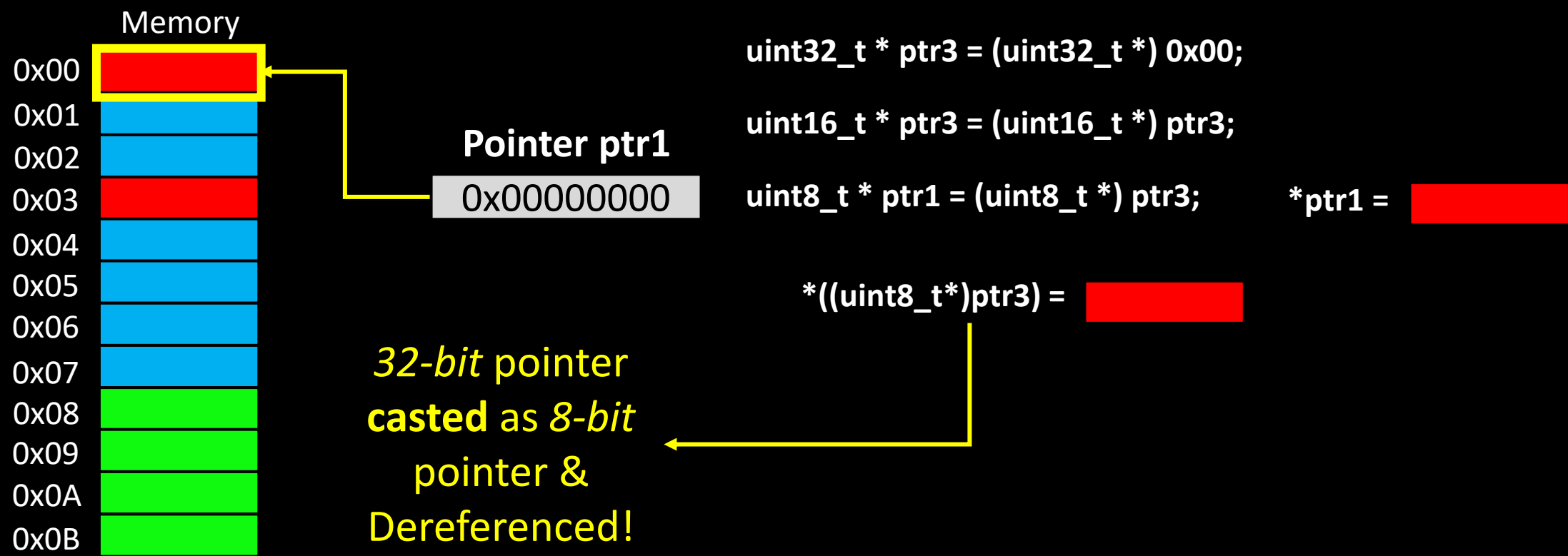
Pointer Casting [S8b]

Cast pointers to dereference **different sizes** from **same address**



Pointer Casting [S8c]

Cast pointers to dereference **different sizes** from **same address**



Pointers in Memory [S9]

- Pointers exist any part of memory
 - Stack, Heap, BSS, Data
- Pointers can reference data in different parts of memory
 - Code, Data, Peripheral

