MEMORY MANAGEMENT & POINTER BASIC

Agenda

Memory management

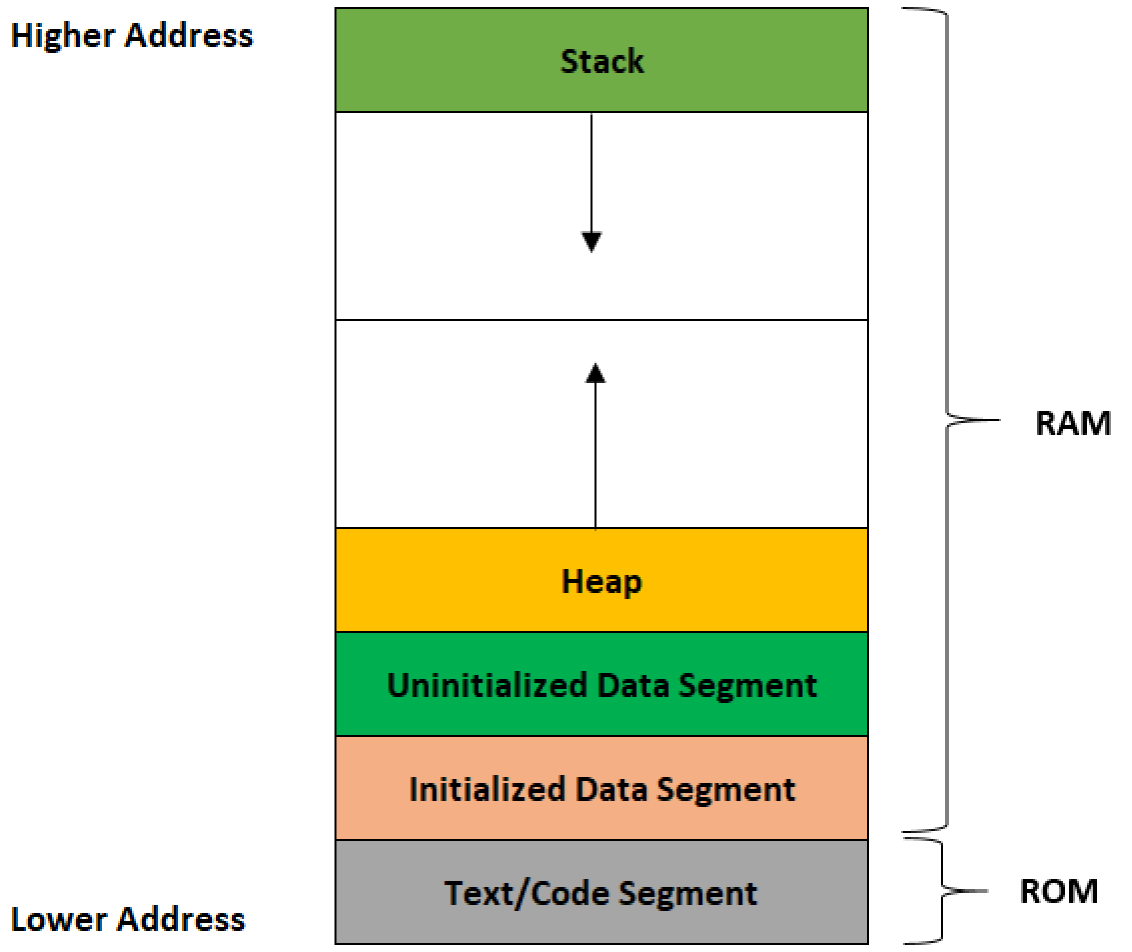
Pointer basic

C MEMORY MANAGEMENT

Content

- Memory layout of C program
- Memory layout of Embedded C program

Memory layout of C program



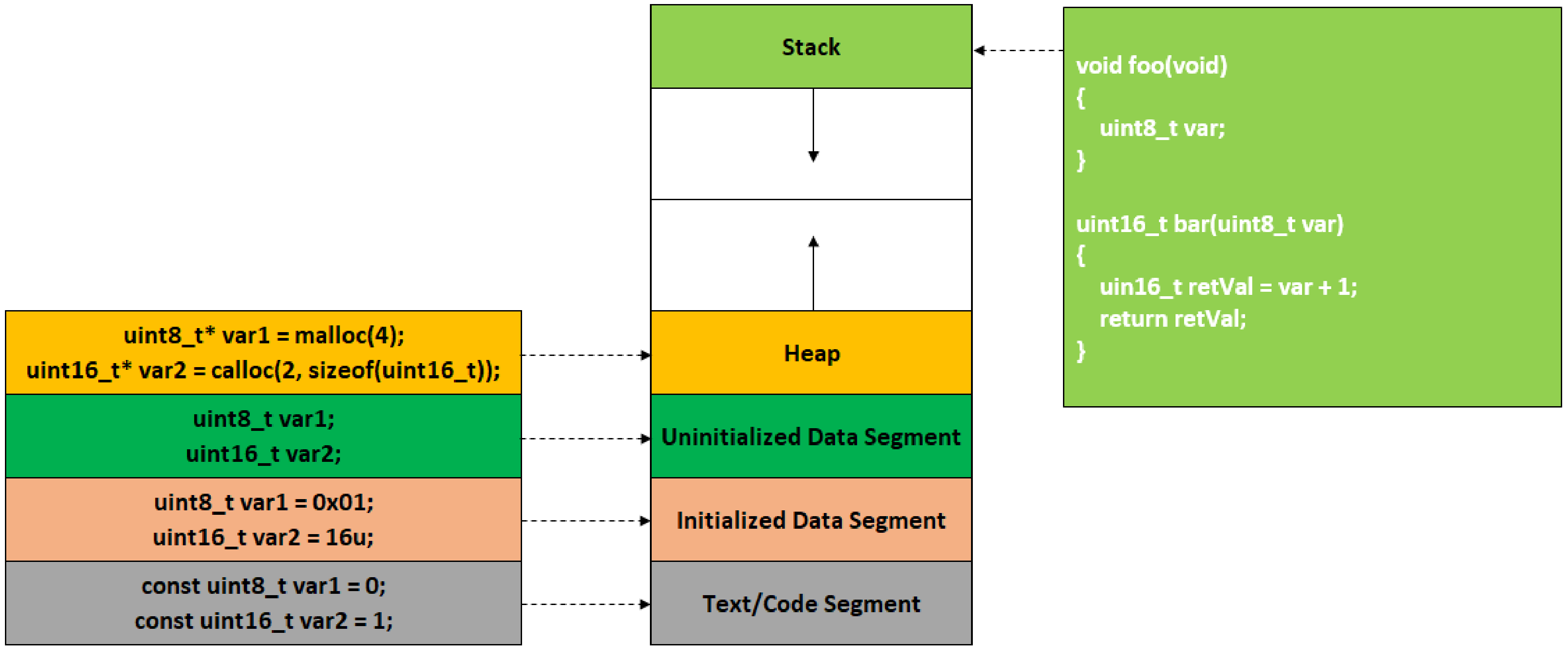
Memory layout of C program

- Text/Code Segment: Store program equivalent machines instructions
- Initialized Data Segment: Store global variables, static variables that are initialized by programmer
- Unitialized Data Segment (bss): Store global variables, static varibales that are initialized to zero or do not have explicit initialization in source code.

Memory layout of C program

- Heap Segment: where dynamic memory allocation usually takes place and is managed by malloc, realloc and free.
- Stack Segment: Store function frames (Function parameters, Return address, Saved previous frame pointer, Local variables)

Memory layout of C program



```
uint8_t* var1 = malloc(4);
uint16_t* var2 = calloc(2, sizeof(uint16_t));
```

```
uint8_t var1;
uint16_t var2;
```

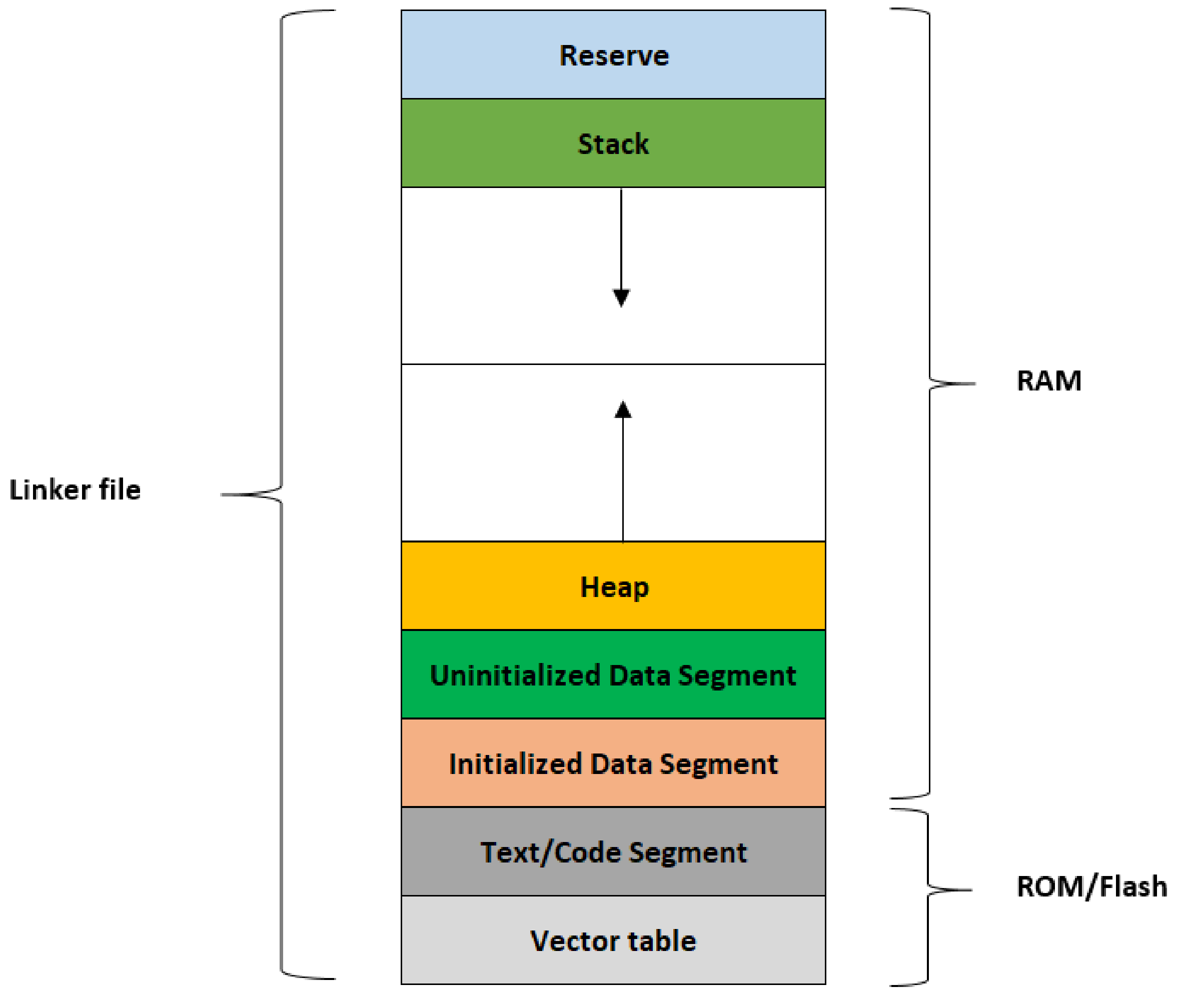
```
uint8_t var1 = 0x01;
uint16_t var2 = 16u;
```

```
const uint8_t var1 = 0;
const uint16_t var2 = 1;
```

```
void foo(void)
{
   uint8_t var;
}

uint16_t bar(uint8_t var)
{
   uin16_t retVal = var + 1;
   return retVal;
}
```

Memory layout of Embedded C program



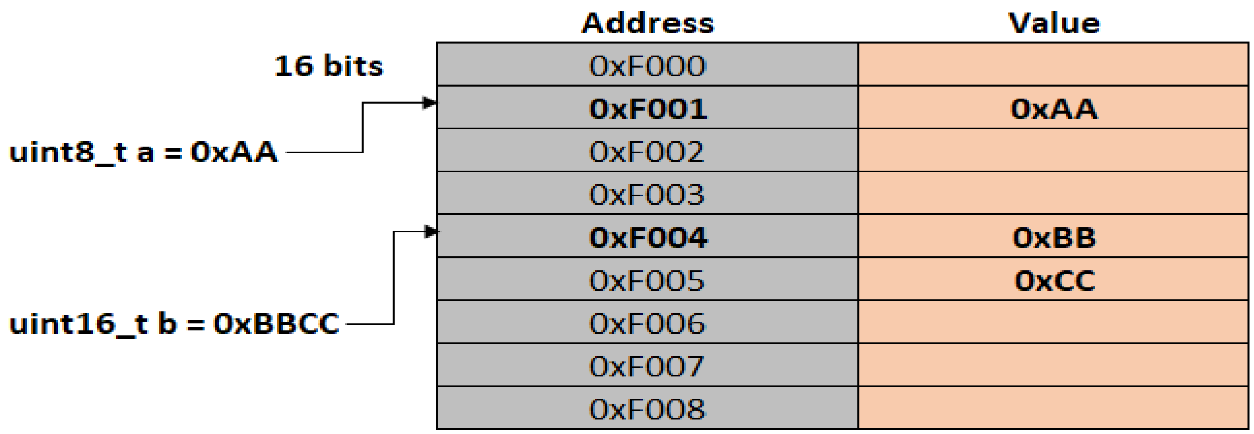
POINTER BASIC

Content

- Variable address
- Pointer variable
- Assigning address to a pointer
- Dynamic memory allocation
- Pointer arithmetic

Variable address

- The variable address is a number that indicates where the data is stored in memory



Address	Value
0xF000	
0xF001	**0xAA**
0xF002	
0xF003	
0xF004	**0xBB**
0xF005	**0xCC**
0xF006	
0xF007	
0xF008	

E.g.

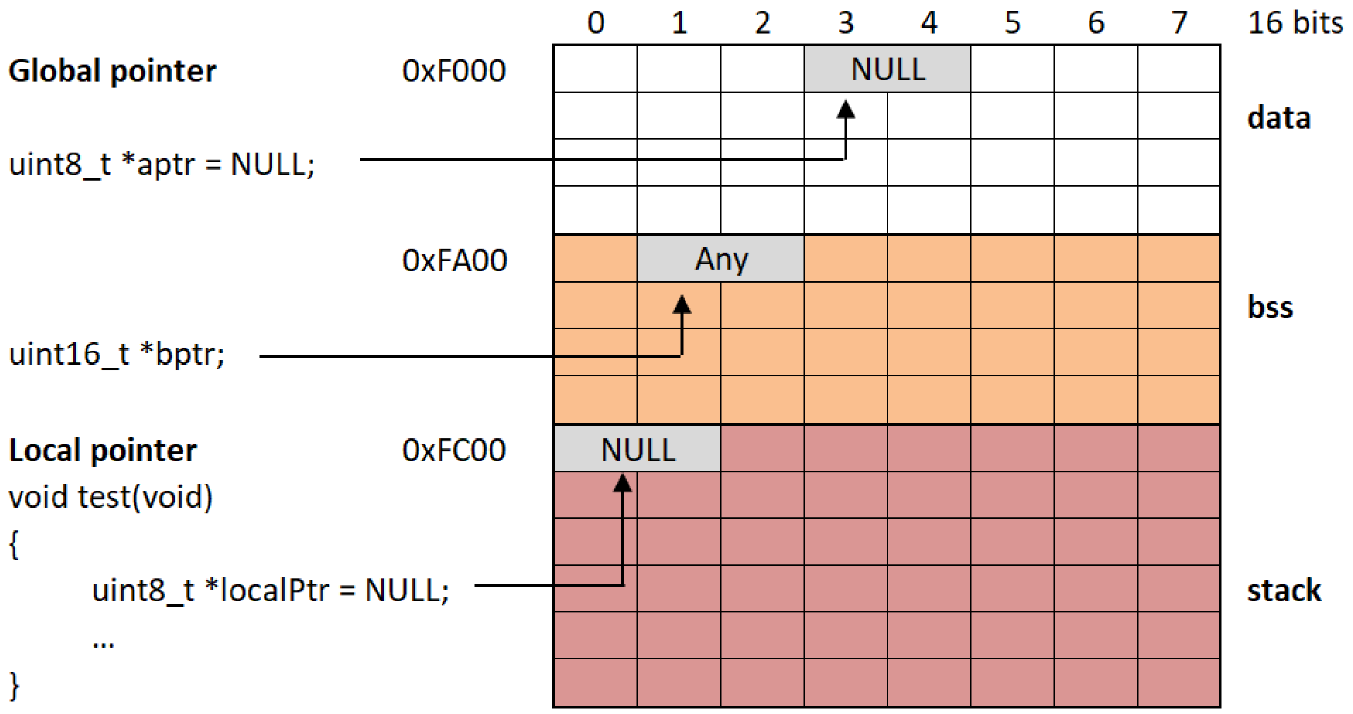
```
printf("Address of a: %x \n", &a);
printf("Address of b: %x \n", &b);
```

Output

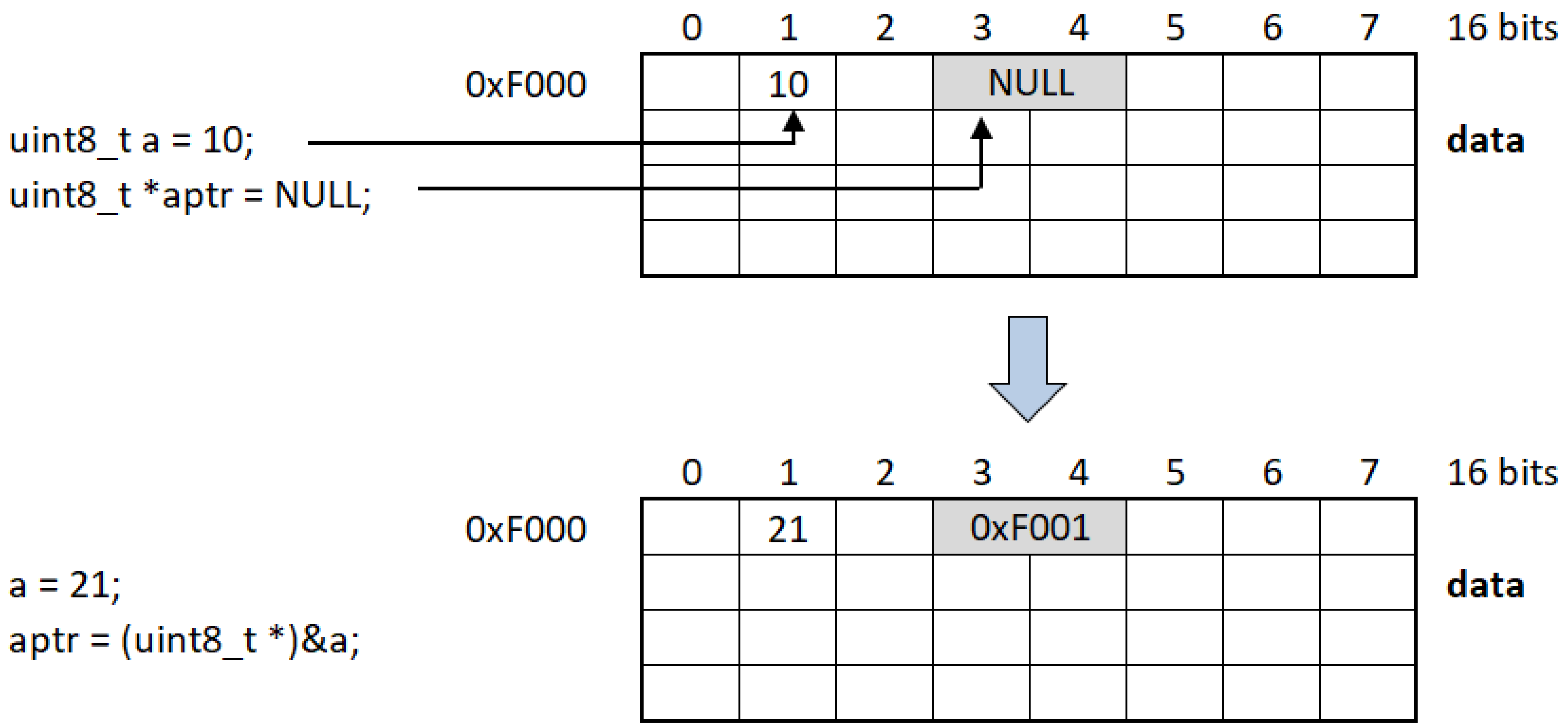
```
Address of a: 0xF001
Address of b: 0xF004
```

Pointer variable

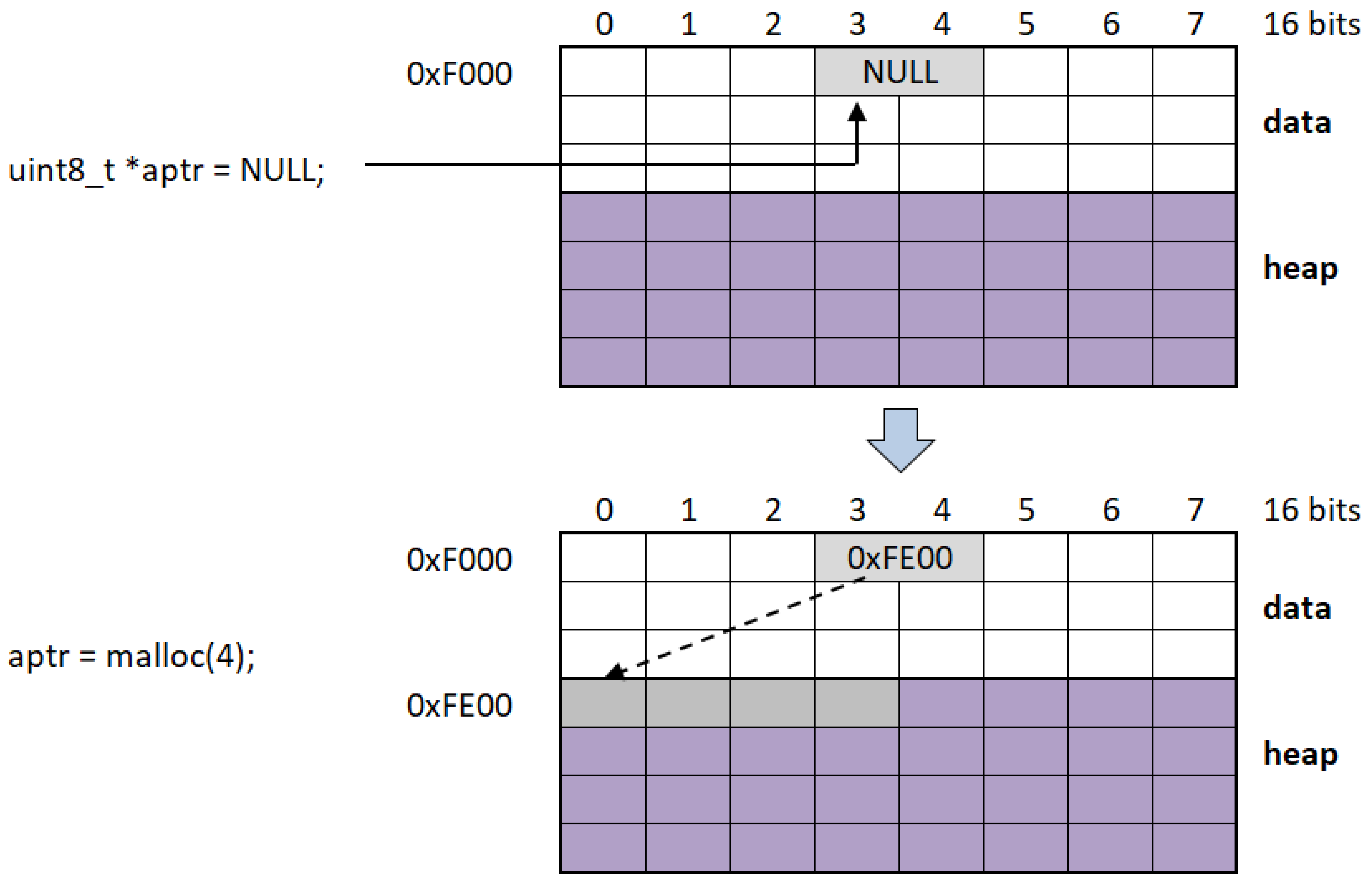
- A **pointer** is a variable whose value is the address of another variable
- **Syntax**: <kiểu dữ liệu> * <tên biến>



Assigning address to a pointer

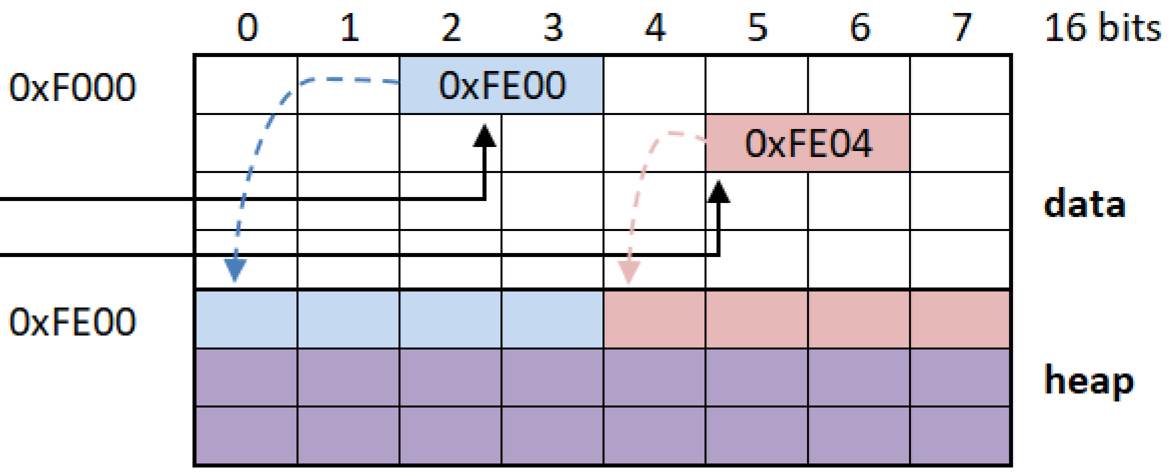


Dynamic memory allocation



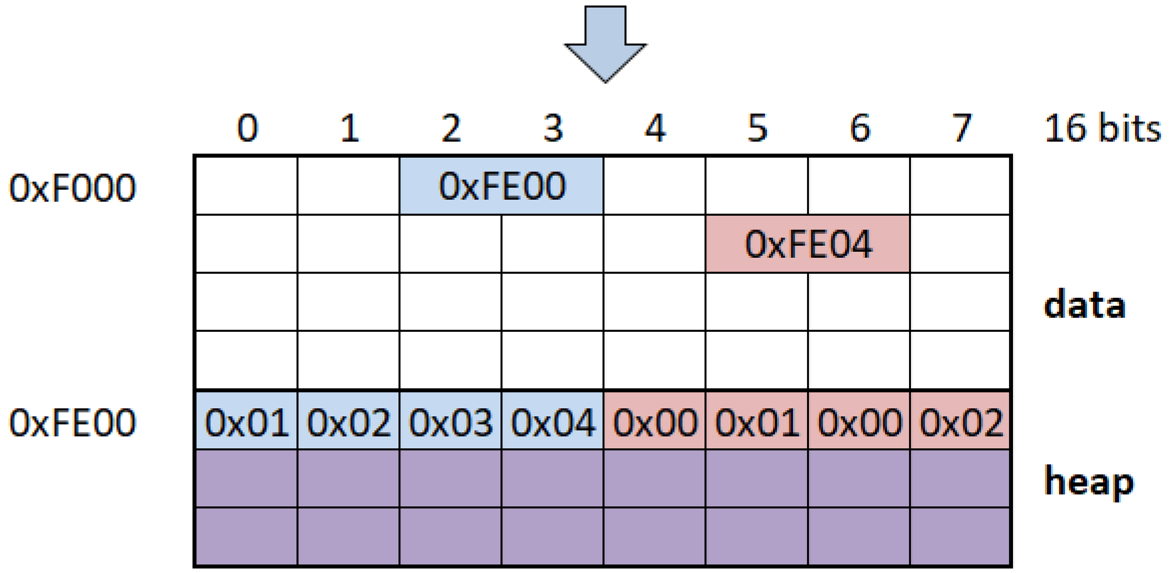
Pointer arithmetic

```
uint8_t *  aptr;
uint16_t *bptr;
```



```
for (i = 0; i < 4; i++)
{
 *(aptr + i) = i + 1;
}

for (i = 0; i < 2; i++)
{
 *(bptr + i) = i + 1;
}
```



Good bye