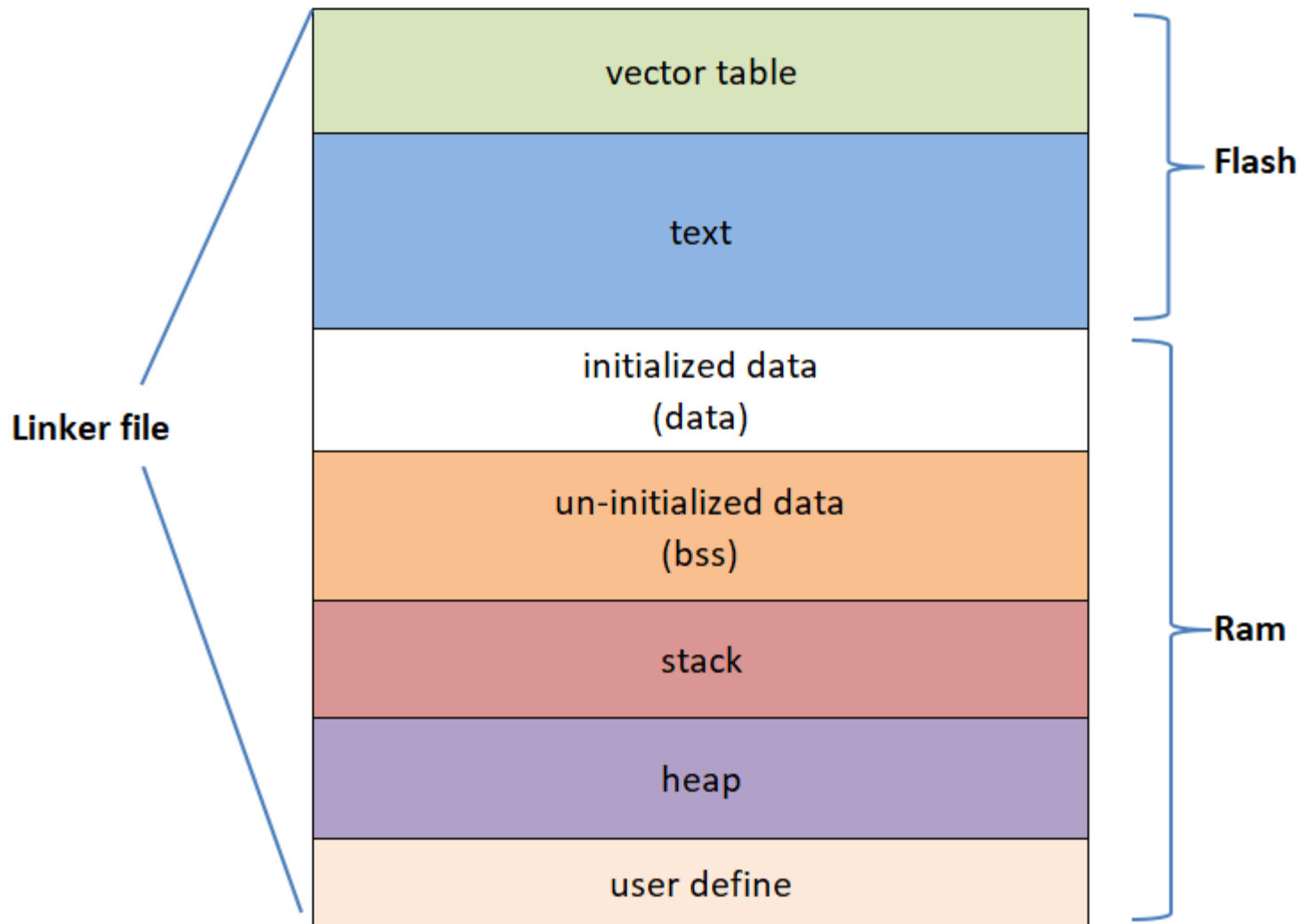


Memory management & Pointer basic

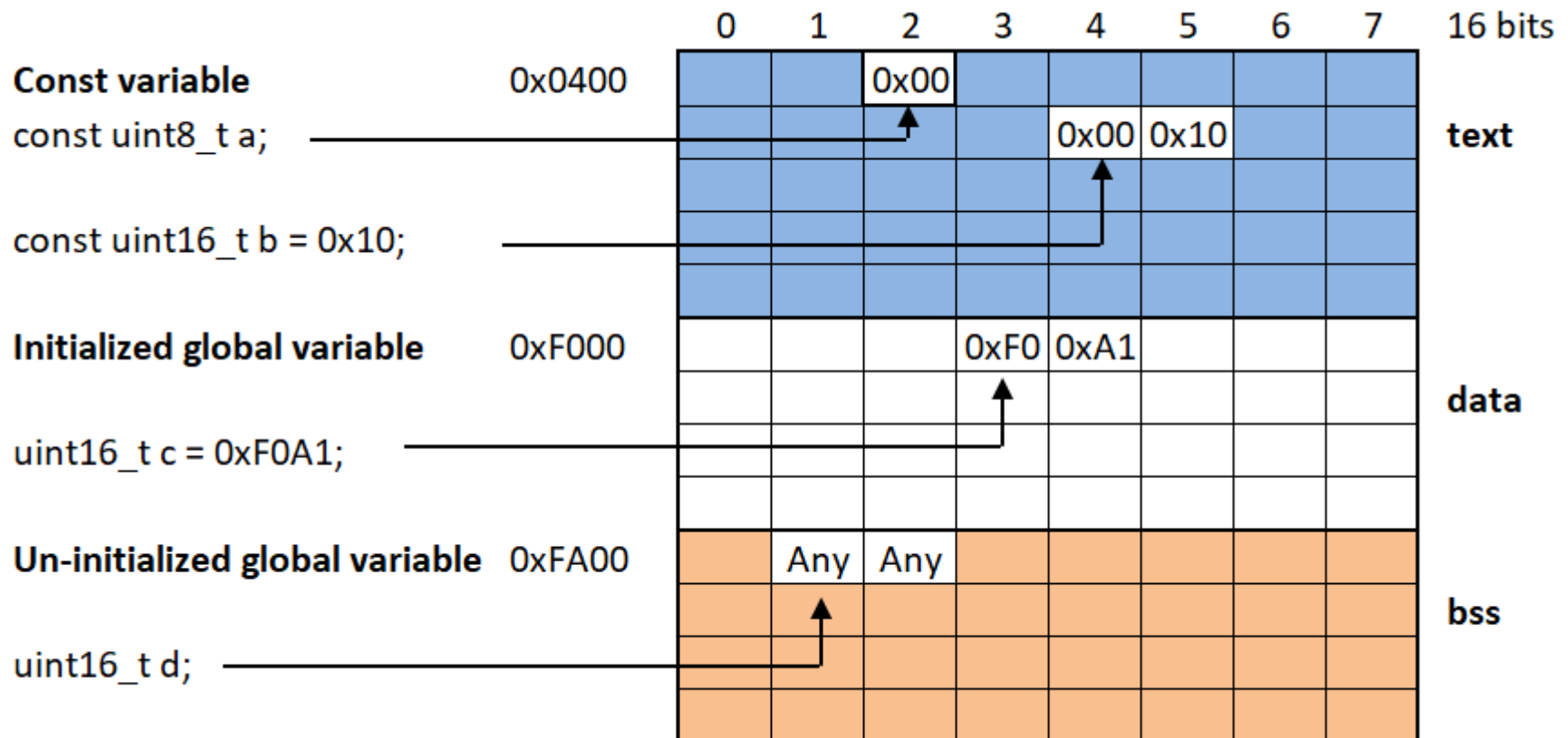
HaiND1

- Memory layout
- Variable and memory location
- Linker file and memory
- Pointer variable
- Assigning values to a pointer
- Memory allocation for a pointer
- Pointer arithmetic

Memory layout



Variable and memory location



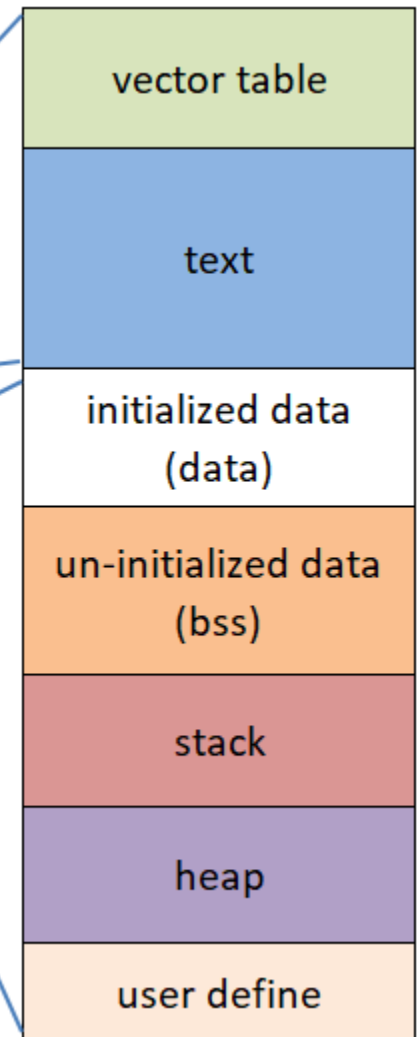
Linker file & Memory

Linker file specifies memory areas

MEMORY

```
{
  /* Flash */
  m_int   (RX) : ORIGIN = 0x00000000, LENGTH = 0x00000400
  m_text  (RX) : ORIGIN = 0x00000410, LENGTH = 0x0007FBF0

  /* Ram */
  m_data  (RW) : ORIGIN = 0x1FFF8000, LENGTH = 0x00008000
}
```



Linker file defines output sections, below is example:

<pre>.stack __StackLimit : { . = ALIGN(8); __stack_start__ = .; . += STACK_SIZE; __stack_end__ = .; } > m_data</pre>	<p>"," means begin at ".stack __StackLimit:" -> Stack section begins at __StackLimit address.</p>
<pre>.mySection : { __mySection_start__ = .; . += MY_SECTION_SIZE; __mySection_end__ = .; } > m_data</pre>	<p>".mySection" is user defined section. -> ".mySection :" means "user defined section begins after stack section</p>

Linker file & Memory

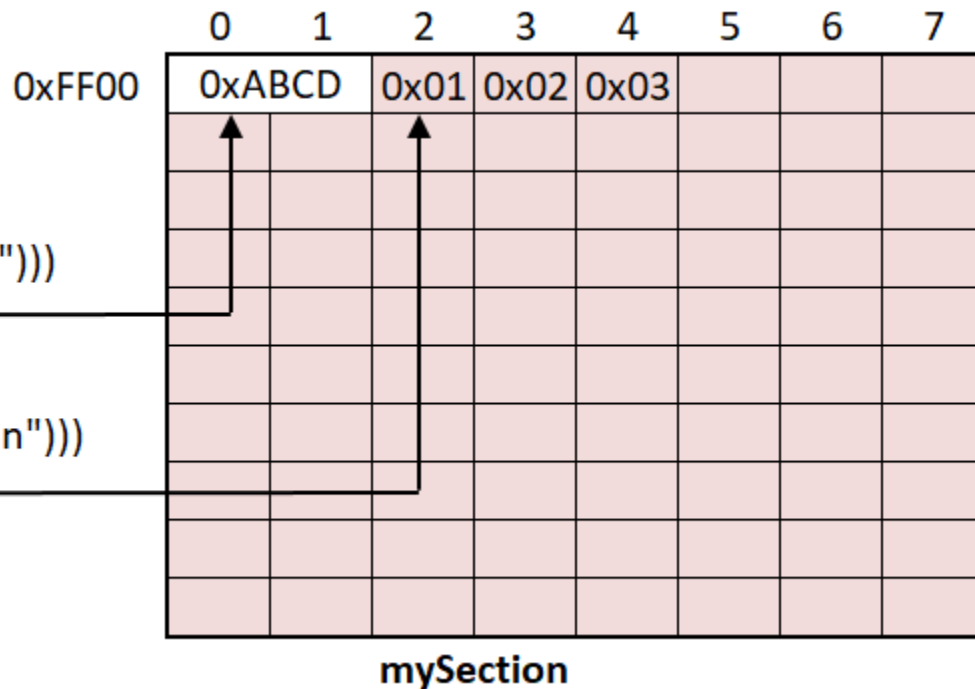
Variable in user defined section

Use instruction to put a variable into user defined section.

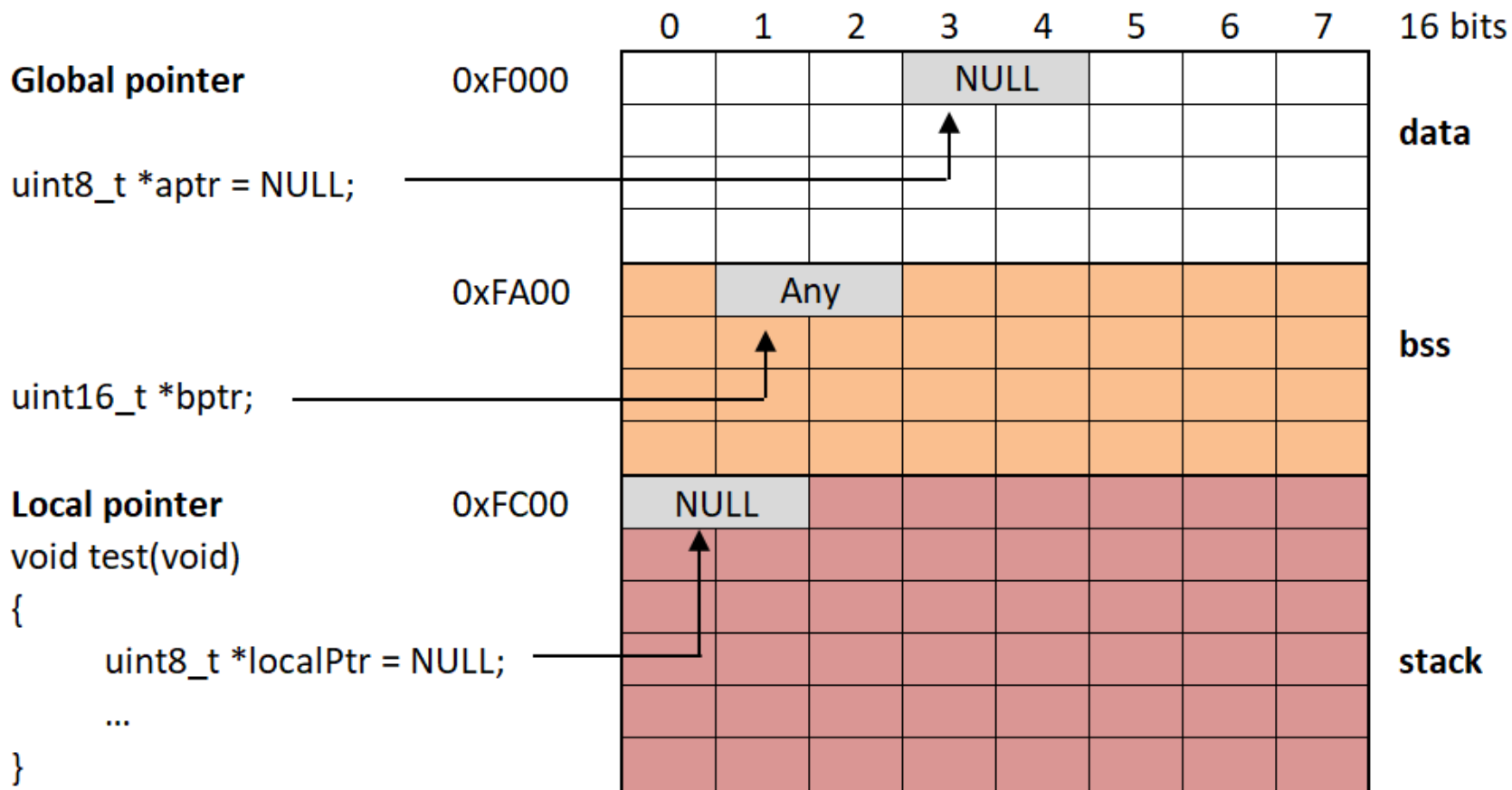
```
uint16_t x __attribute__((section (".mySection")))
    = 0xABCD;
```

```
uint8_t y[3] __attribute__((section (".mySection")))
    = { 0x01, 0x02, 0x03 };
```

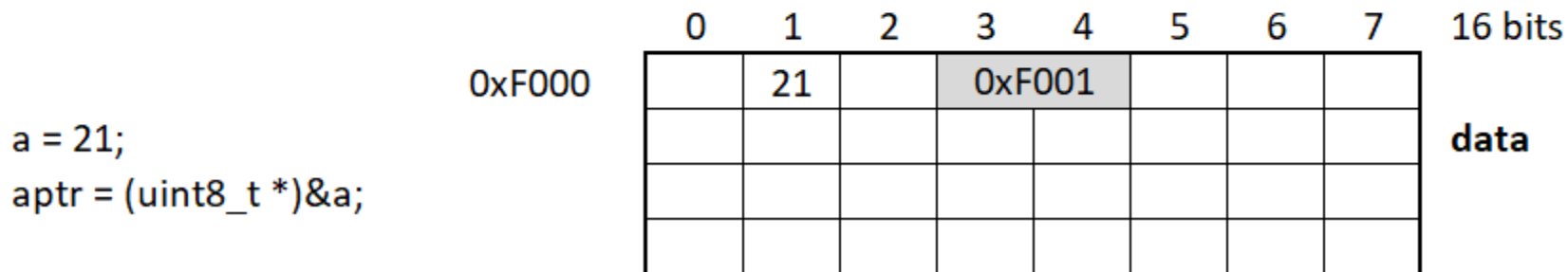
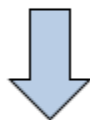
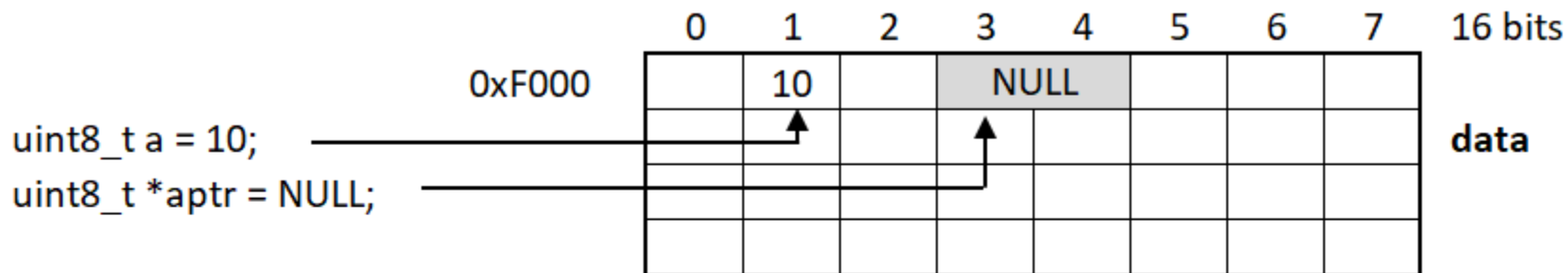
instruction is different between toolchains.



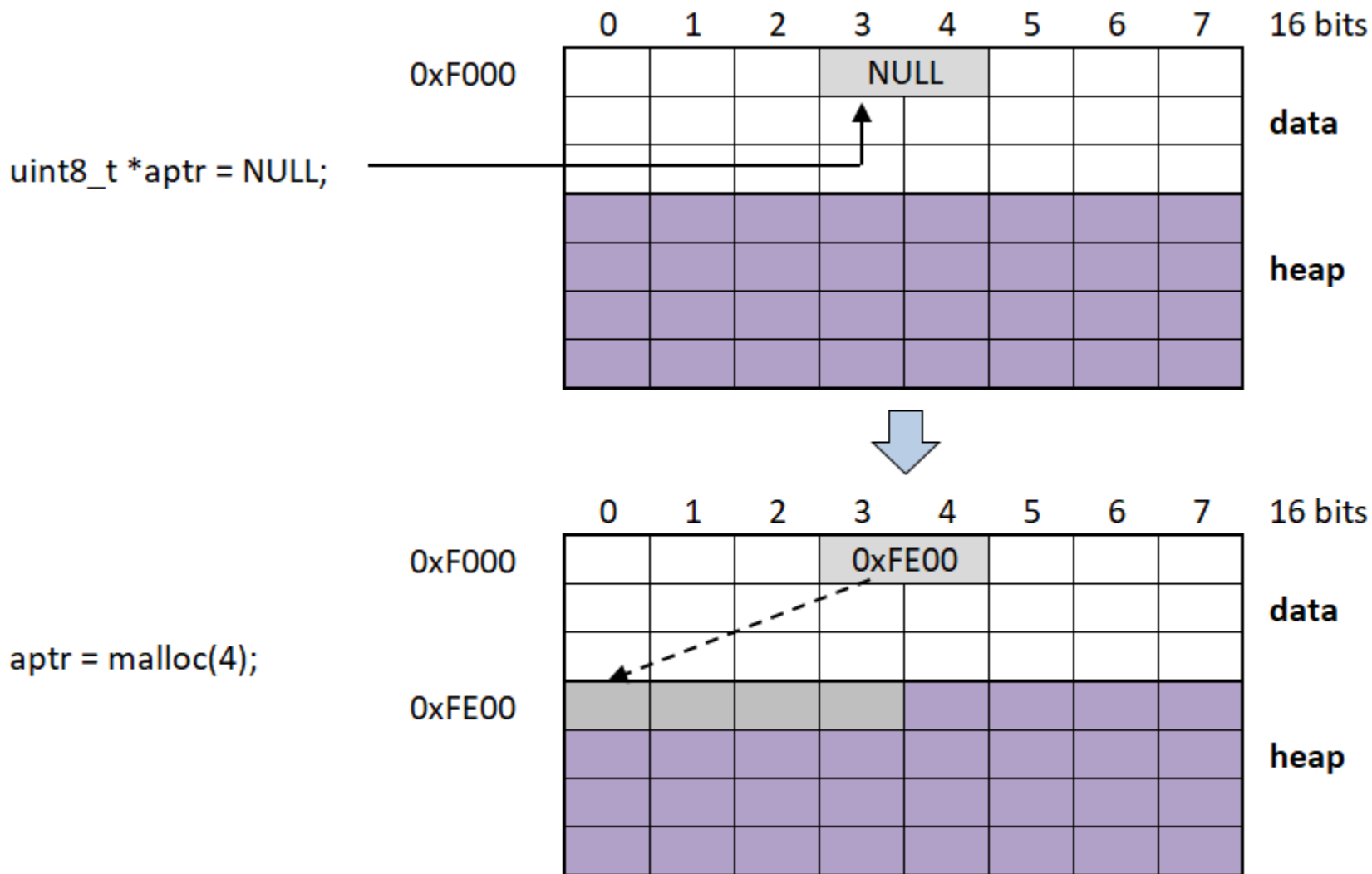
Pointer variable



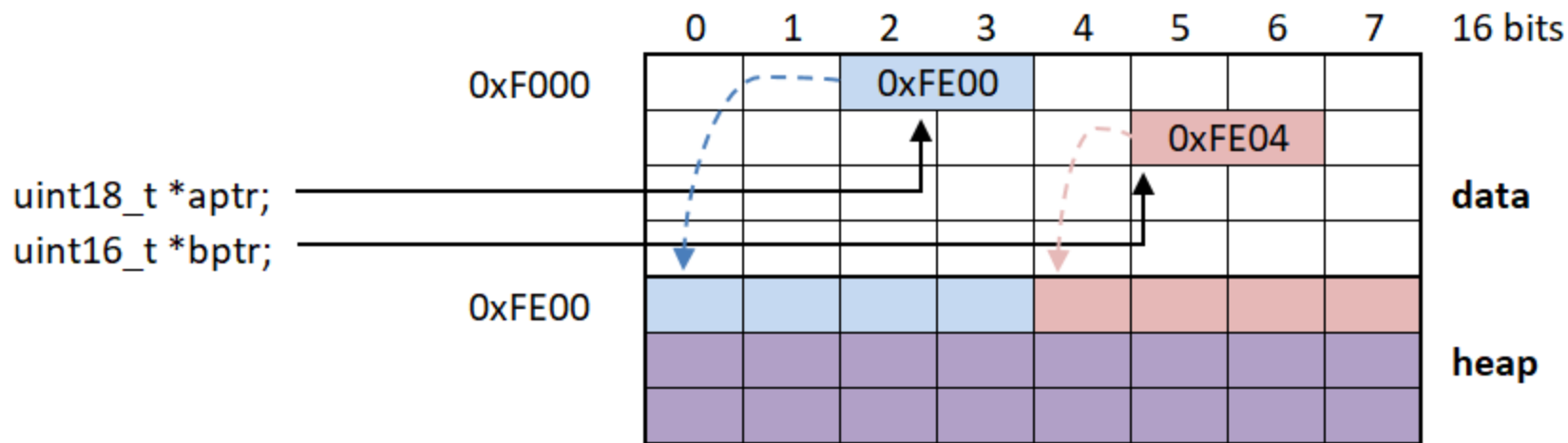
Assigning values to a pointer



Memory allocation for a pointer

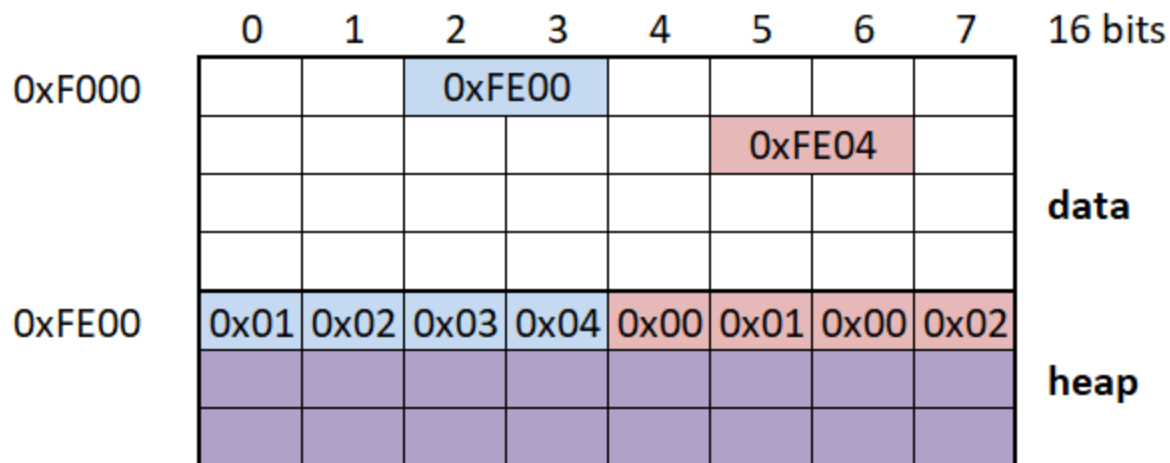


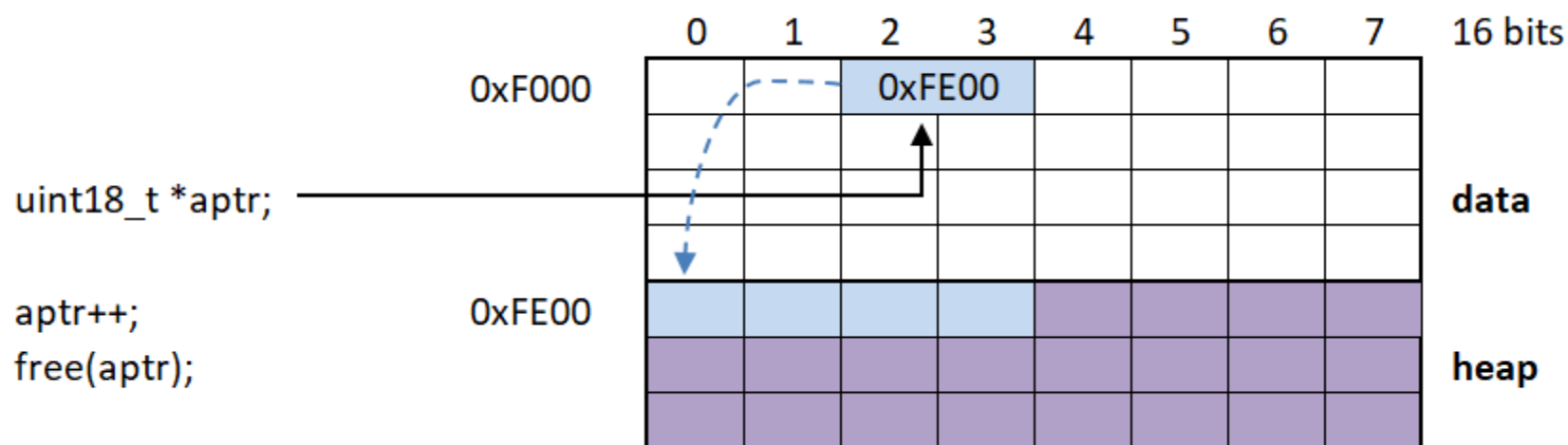
Pointer arithmetic



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

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





Q & A