Memory Segments

Embedded Software Essentials

C1M3V3

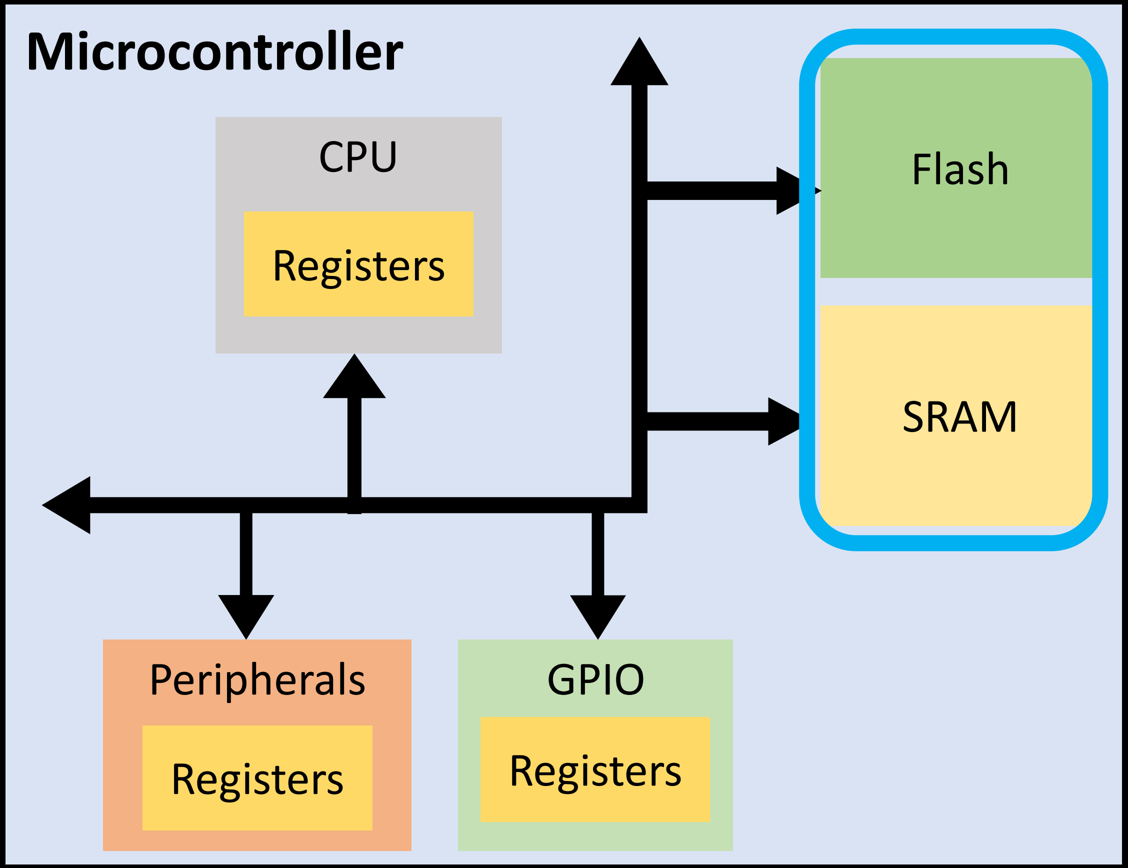
Be Boulder.

University of Colorado Boulder

Embedded System Memories

- Memories of an Embedded Systems
  - Code Memory (Flash)
  - Data Memory (SRAM)
  - Register Memory (internal to chip)
  - External Memory (if applicable)
- Compilation tracks and maps memory from a program into segments

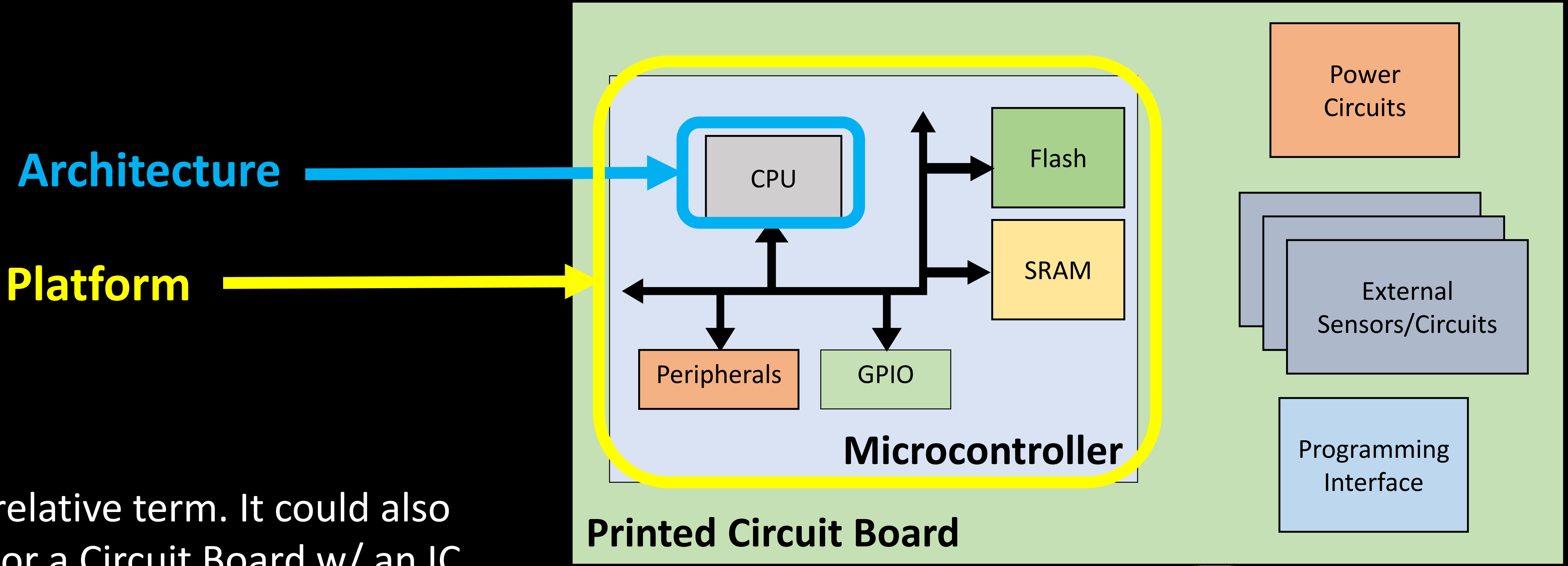
→ **Specified in the Linker File**



Be Boulder.
University of Colorado Boulder

Platforms

- **Platform**[1] - The underlying Integrated Circuit (IC) and the components surrounding the CPU (Peripherals)



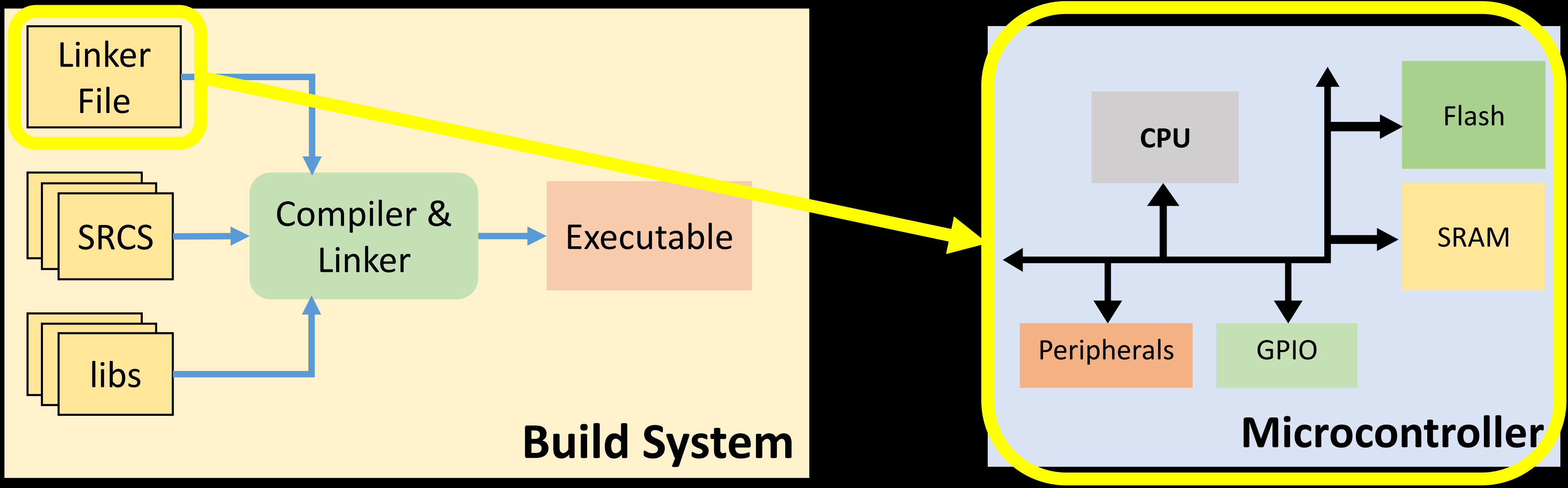
[1]Platform is a relative term. It could also refer to an OS or a Circuit Board w/ an IC

University of Colorado Boulder

Platforms

- **Platform**[1] - The underlying Integrated Circuit (IC) and the components surrounding the CPU (Peripherals)

Linker Files MUST be PLATFORM DEPENDENT!!!



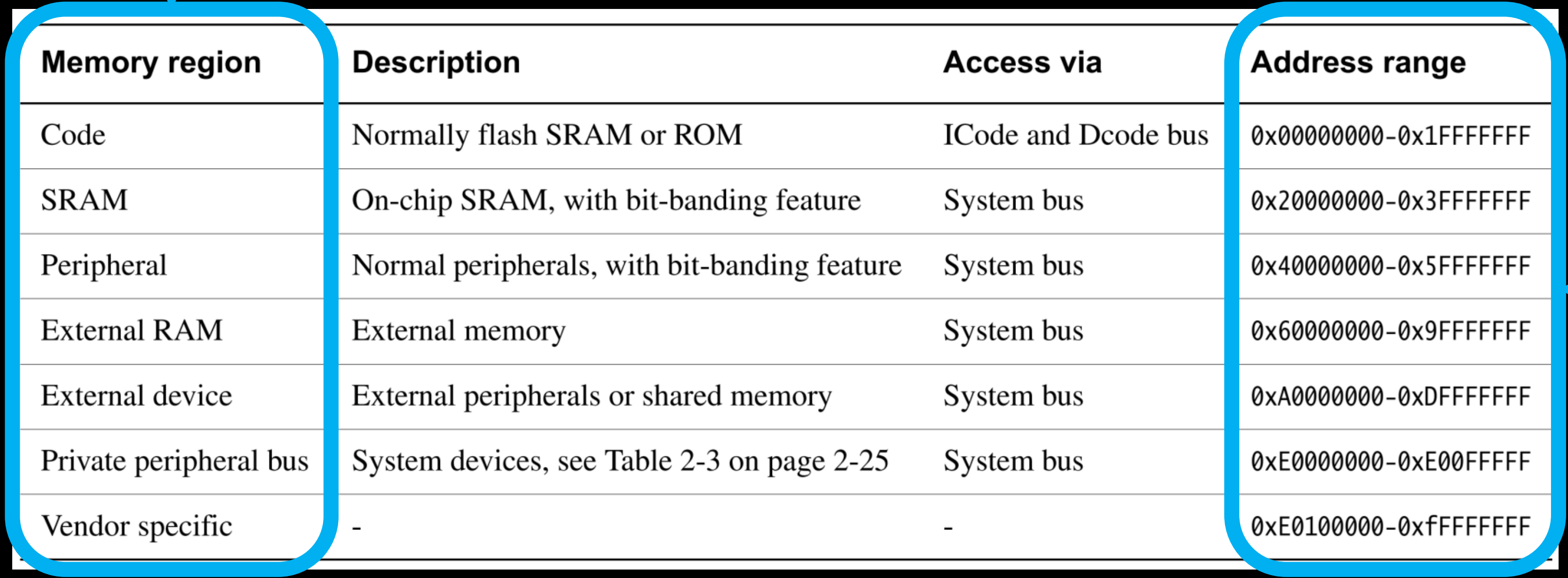
[1]Platform is a relative term. It could also refer to an OS or a Circuit Board w/ an IC

Plaform may affect the Address Space of the microcontroller (different memories)

Memory Map

- **Memory Map**: Provides a memory address to physical device mapping within an address space for use in programming

Mapped Components (Memory, Peripherals, System Config, etc)



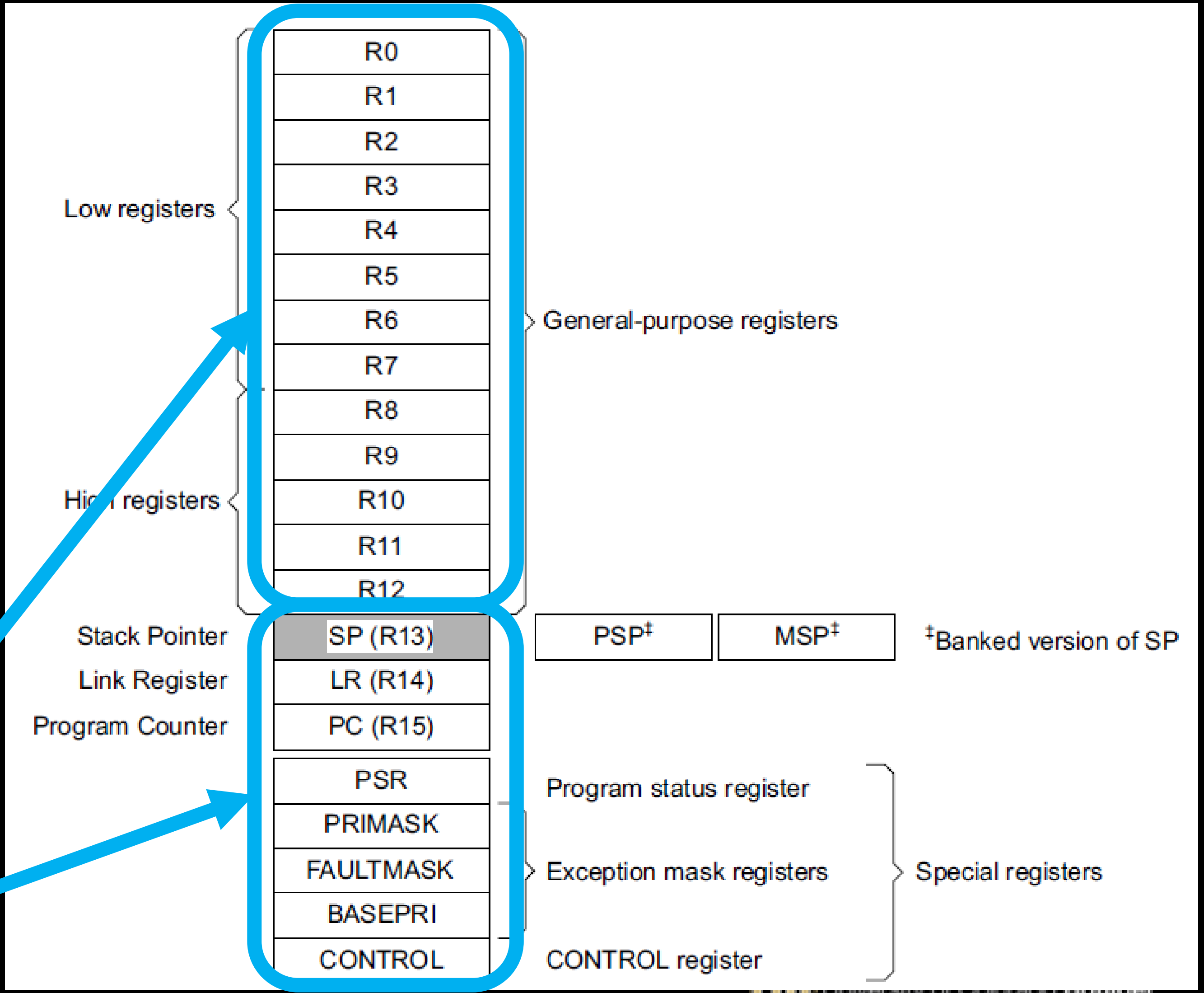
Memory region	Description	Access via	Address range
Code	Normally flash SRAM or ROM	ICode and Dcode bus	0x00000000-0x1FFFFFFF
SRAM	On-chip SRAM, with bit-banding feature	System bus	0x20000000-0x3FFFFFFF
Peripheral	Normal peripherals, with bit-banding feature	System bus	0x40000000-0x5FFFFFFF
External RAM	External memory	System bus	0x60000000-0x9FFFFFFF
External device	External peripherals or shared memory	System bus	0xA0000000-0xDFFFFFFF
Private peripheral bus	System devices, see Table 2-3 on page 2-25	System bus	0xE0000000-0xE00FFFFF
Vendor specific	-	-	0xE0100000-0xfFFFFFFF

Address Space (Ranges for each device)

Be Boulder.
University of Colorado Boulder

CPU Registers

- General Purpose store operation operands
  - R0-R12
- Special Purpose Track and Control CPU state



General Purpose Registers

Special Purpose Registers

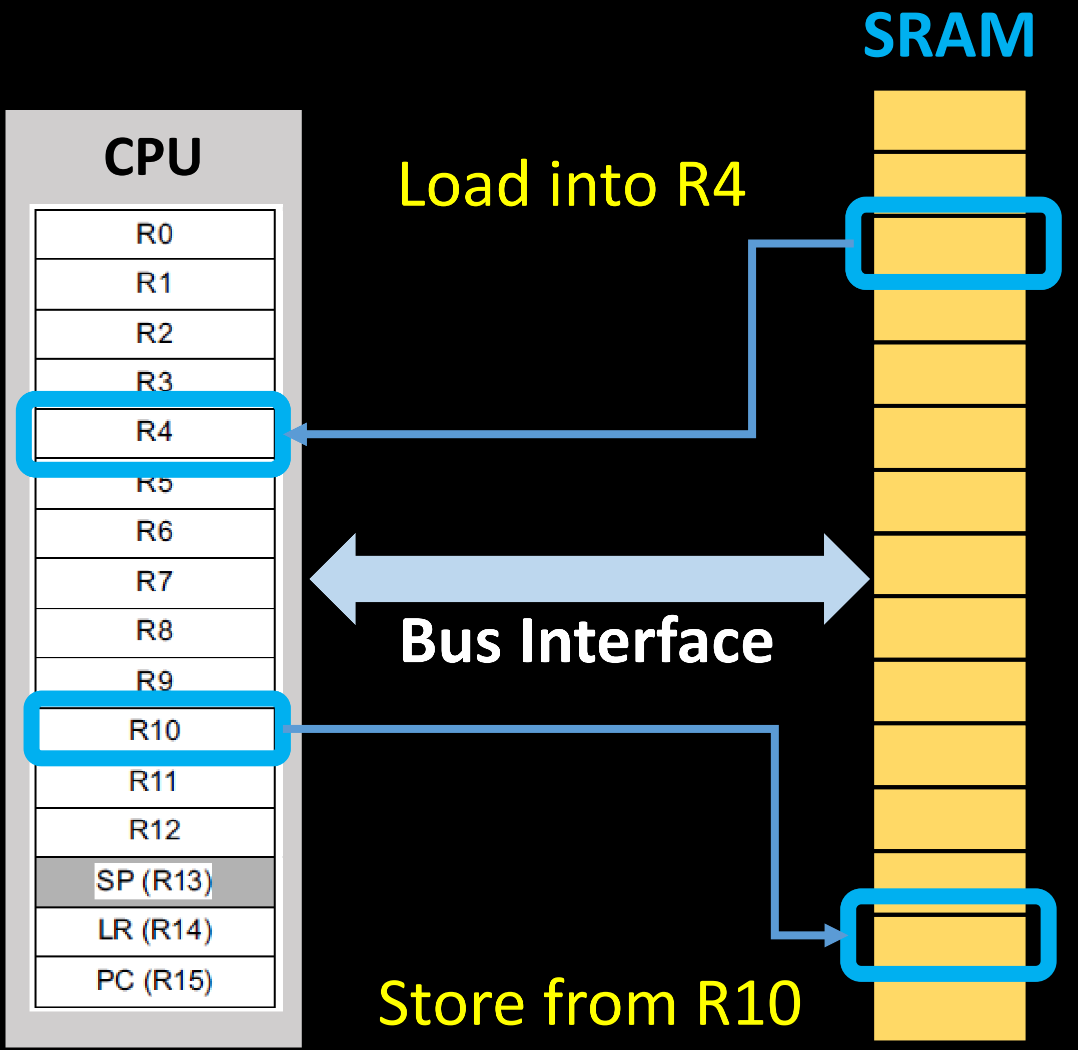
Register Contents

- Register data constantly changes
  - Data is **loaded** in from memory
  - Results are **stored** back to memory

- Application Binary Interface (ABI) provides architecture details to Compiler / Software Programmer

Example Assembly relative load/store:

```
ldr r1, [r7,#8]
str r1, [r7,#8]
```

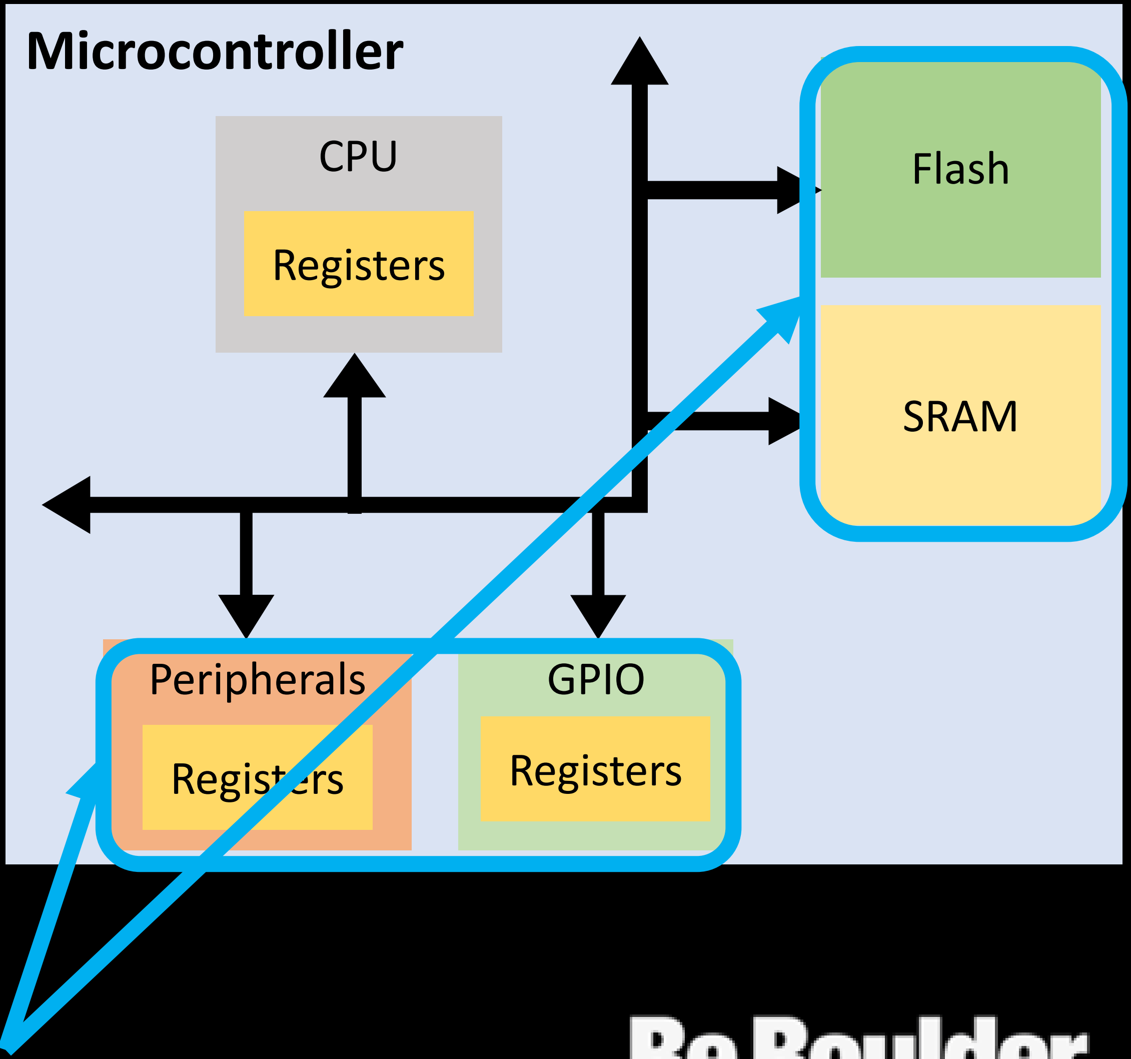


Be Boulder.

University of Colorado Boulder

Platform

- Architecture Families have many different chip sets
  - KL24z vs. KL25z vs KL26z
- These have the same Architecture (ARM) but different memory size and peripheral support



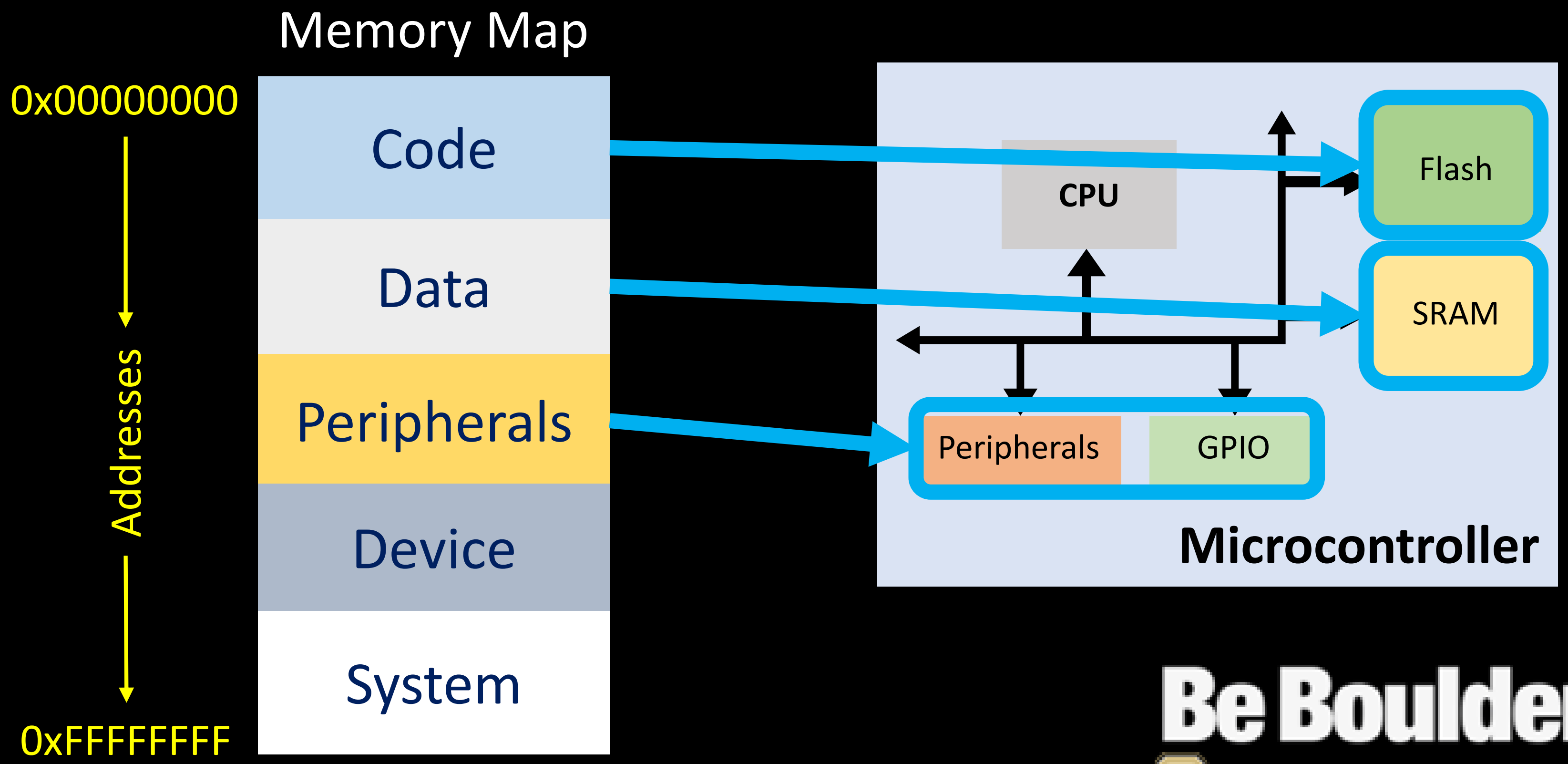
Platform Dependent

Be Boulder.
University of Colorado Boulder

Register Definition Files

- Details on platform specific registers can be put in C-Programming source files (Register Definition Files)

- You do not need to know physical locaitons, just an address
  - Use Pointers!!!



Be Boulder.
University of Colorado Boulder

Linker File

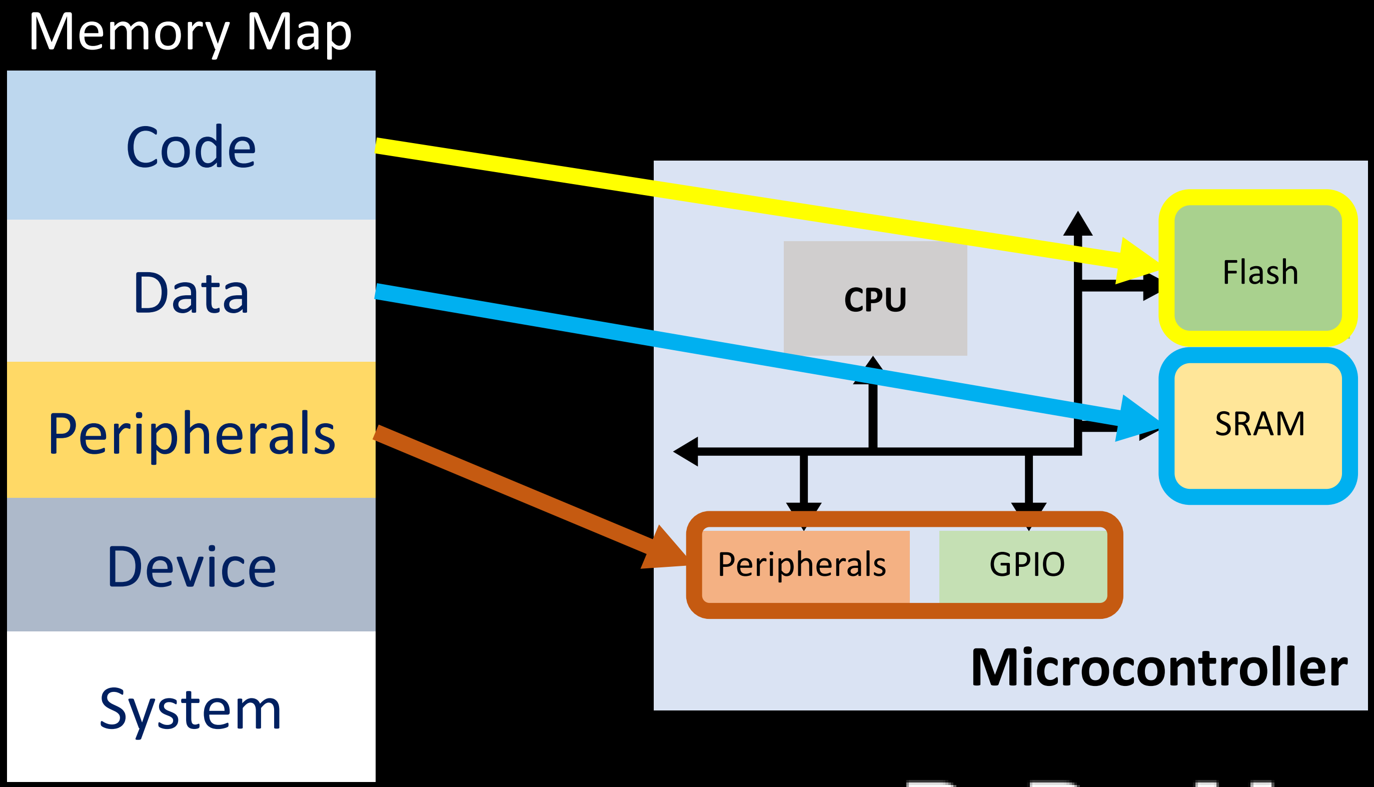
MEMORY **Physical Memory Regions**

```
{
    MAIN  (RX) : origin = 0x00000000, length = 0x00040000
    DATA  (RW) : origin = 0x20000000, length = 0x00010000
}
```

SECTIONS **Compiled Memory Sections**

```
{
    .intvecs :   > 0x00000000
    .text :   > MAIN
    .const :   > MAIN
    .cinit :   > MAIN
    .pinit :   > MAIN
    .data :   > DATA
    .bss :   > DATA
    .heap :   > DATA
    .stack :   > DATA (HIGH)
}
```

- **Memory Map allows access to platform through addresses**



Be Boulder.
University of Colorado Boulder

Linker File

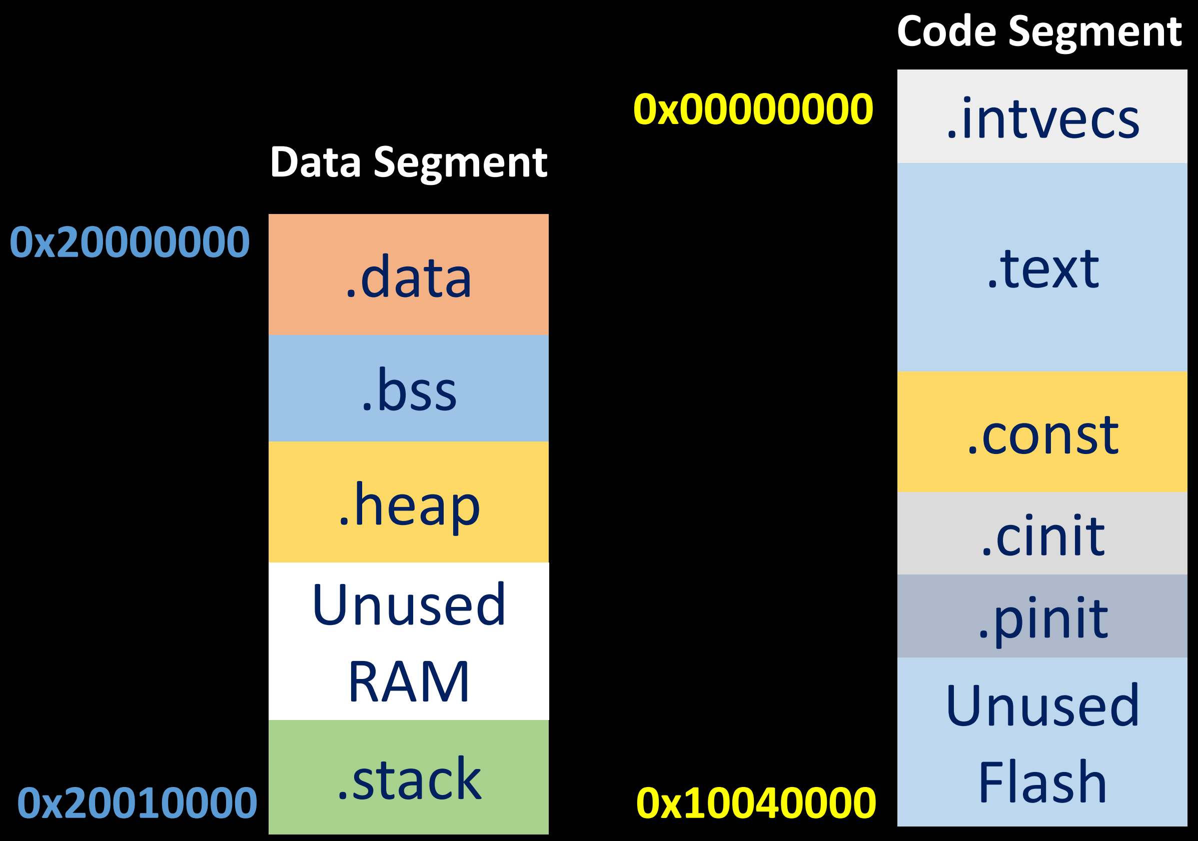
```
MEMORY    Physical Memory Regions
{
    MAIN  (RX) : origin = 0x00000000, length = 0x00040000
    DATA  (RW) : origin = 0x20000000, length = 0x00010000
}

SECTIONS    Compiled Memory Sections
{
    .intvecs :   > 0x00000000
    .text :   > MAIN
    .const :   > MAIN
    .cinit :   > MAIN
    .pinit :   > MAIN
    .data :   > DATA
    .bss :   > DATA
    .heap :   > DATA
    .stack :   > DATA (HIGH)
}
```

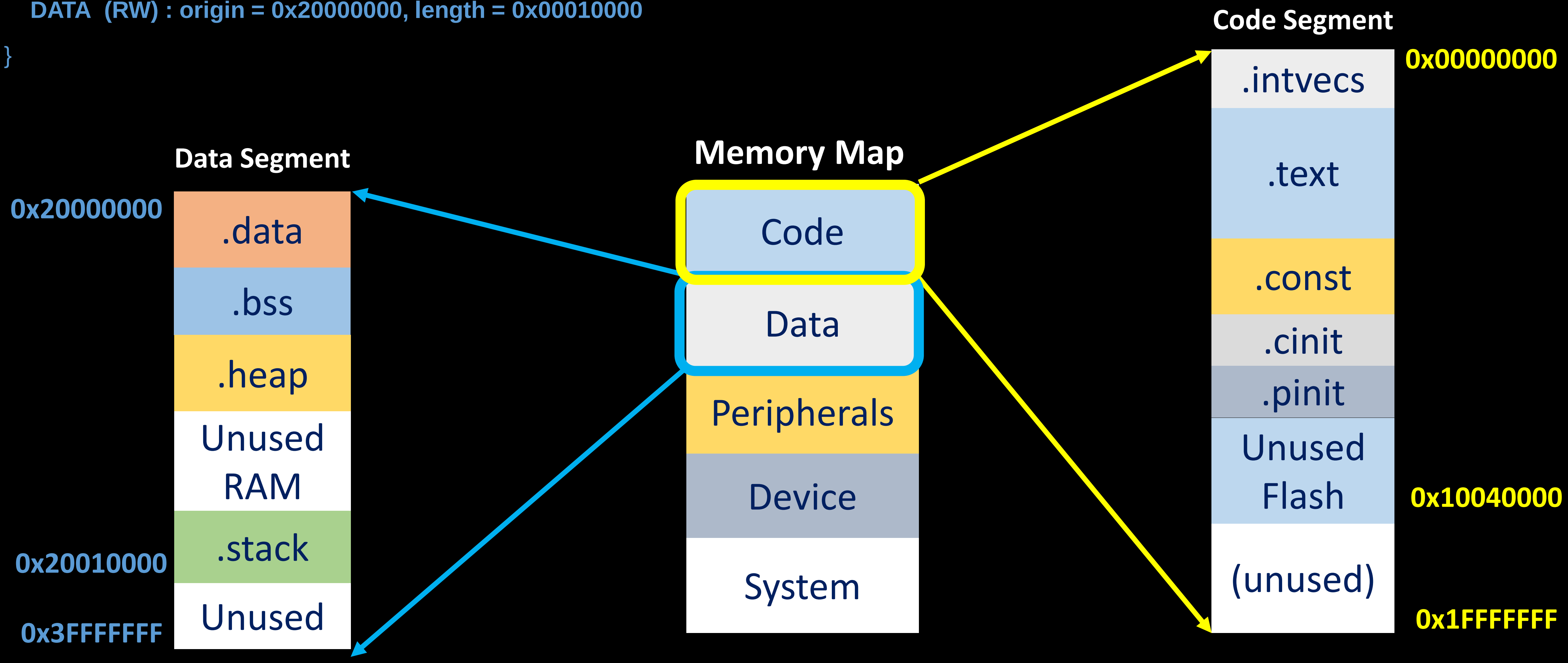
Code Sub-Segments (.intvecs, .text, .const, .cinit, .pinit)

Data Sub-Segments (.data, .bss, .heap, .stack)

- **Linker File provides physical address to symbol mapping**



```
MEMORY
{
   MAIN  (RX) : origin = 0x00000000, length = 0x00040000
   DATA  (RW) : origin = 0x20000000, length = 0x00010000
}
```



Linker File Sub-Segments

- The compiled memory sections of a compiled executable will be relocated by referencing a symbol name

- These symbol names are referred to as memory sub-segments

```
MEMORY   Physical Memory Regions
{
    MAIN  (RX) : origin = 0x00000000, length = 0x00040000
    DATA  (RW) : origin = 0x20000000, length = 0x00010000
}

SECTIONS   Compiled Memory Sections
{
    .intvecs :   > 0x00000000
    .text :   > MAIN
    .const :   > MAIN
    .cinit :   > MAIN
    .pinit :   > MAIN
    .data :   > DATA
    .bss :   > DATA
    .heap :   > DATA
    .stack :   > DATA (HIGH)
}
```

Be Boulder.
University of Colorado Boulder