Special Keywords

Embedded Software Essentials

C1M3V5

Allocated Data Characteristics

- Allocated Data can have varying
  - Size
  - Access
  - Scope
  - Location
  - Creation time
  - Lifetime

Specified by utilizing

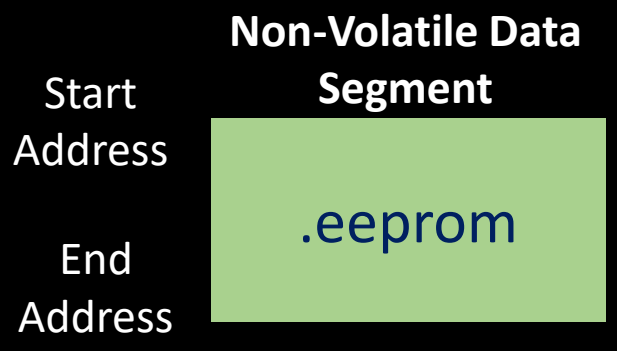
- **Variable types**
- **Type Qualifiers**
- **Type Modifiers**
- **Storage Classes**
- **Compiler Attributes**
- **Specialized Functions**

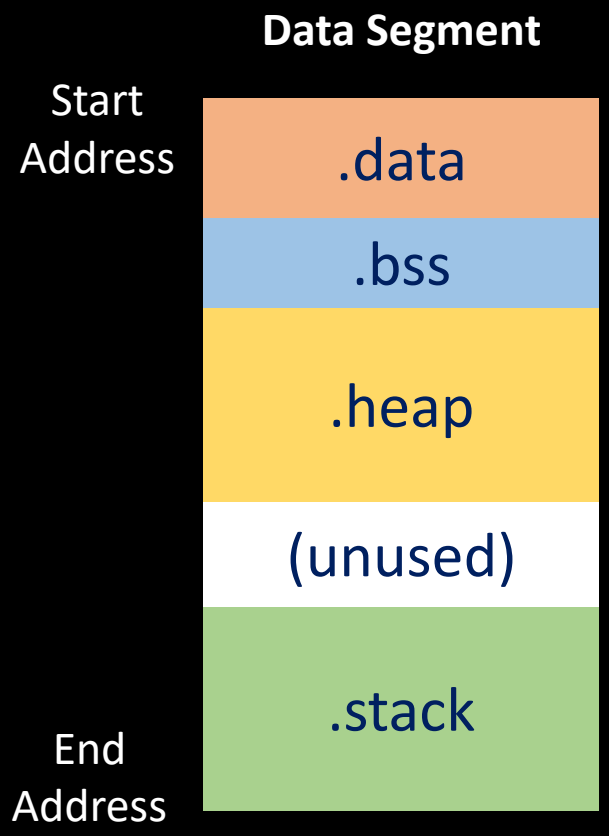
- Data allocation is not limited to static allocation at compile time, but also dynamic allocation at runtime.

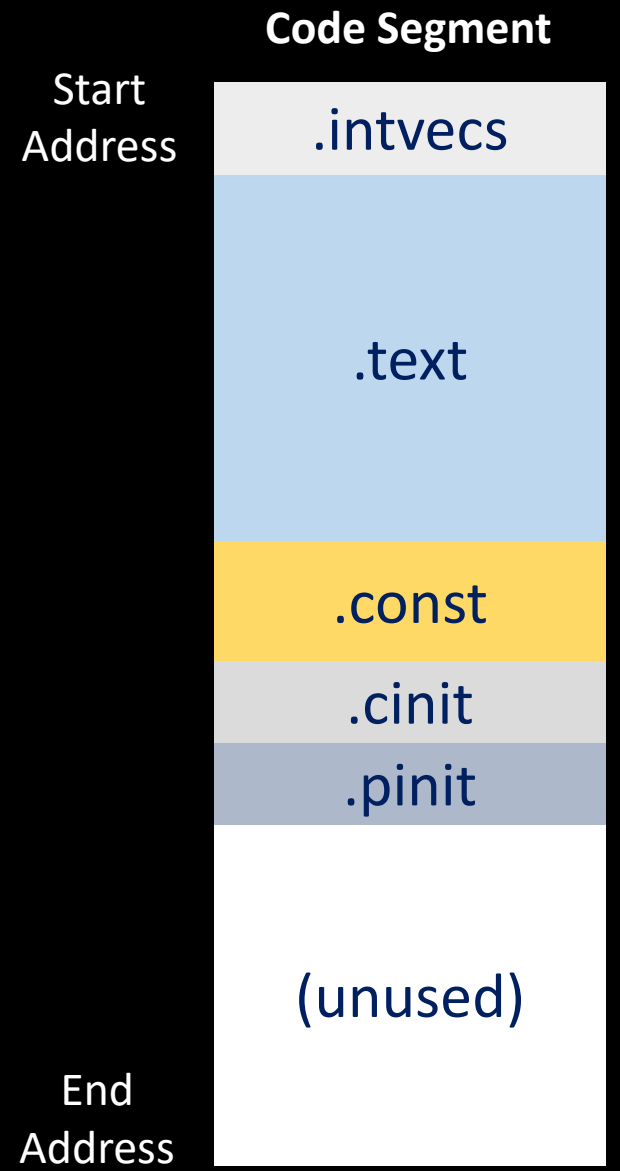
Linker File outlines the memory segments

Data Locations

- Data Memory (Most Common)
  - RAM
- Code Memory
  - Read-Only Data
- External Non-volatile Memory
  - Non-Volatile Data







C Keywords

- Special C-Keywords can affect data allocation
  - Variable Types
  - Type Qualifiers
  - Type Modifiers
  - Storage Classes*

* Can apply to other references like functions

C Keywords

- Special C-Keywords can affect data allocation
  - Variable Types
  - Type Qualifiers
  - Type Modifiers
  - Storage Classes

Type most directly affect the size of the data allocated

Examples:

- **char, short, int, float, etc**
- **Derived types**
- **Enumerated Types**
- **_Bool, _Complex (c99)**

Sizes are Architecture Dependent

C Keywords

- Special C-Keywords can affect data allocation
  - Variable Types
  - Type Qualifiers
  - Type Modifiers
  - Storage Classes

Const Type Qualifier

- Allocates the data as a constant, will be mapped to Read-Only Memory

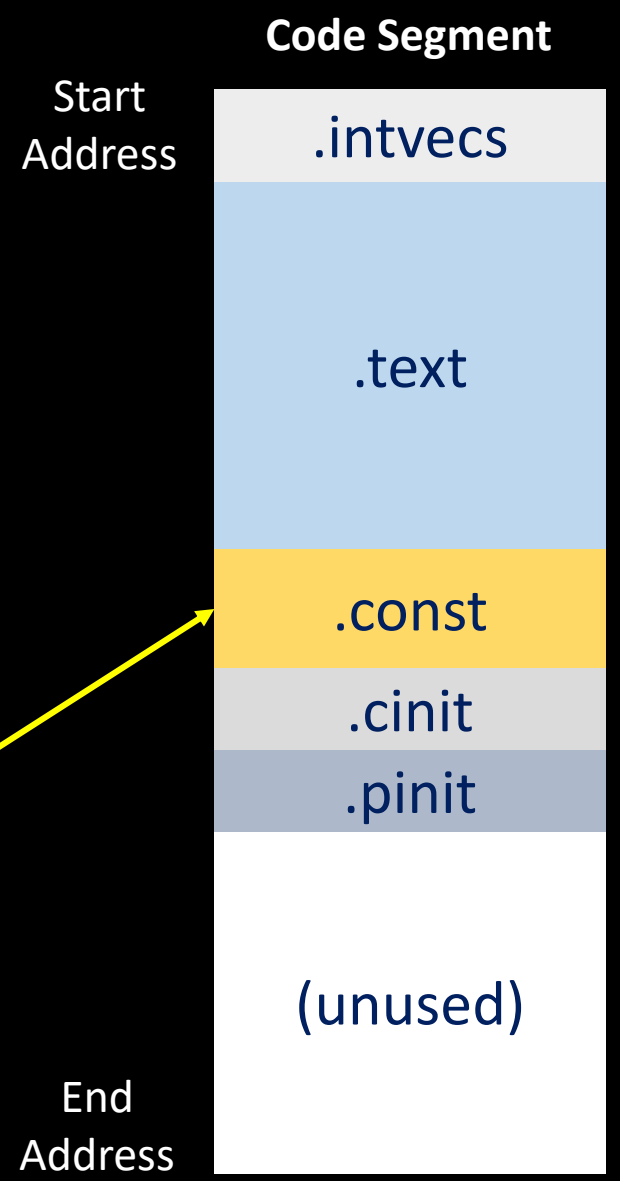
```
const char VARA = 'a';
const int VARB = 1;
```

Const Keyword

- Data will be constant & put in read-only memory like flash

- Sub-Segment name is application dependent
  - .rodata
  - .const

```
const char VARA = 'a';
const int VARB = 1;
```



C Keywords

- Special C-Keywords can affect data allocation
  - Variable Types
  - Type Qualifiers
  - Type Modifiers
  - Storage Classes

Type modifiers modify the size and sign of data type

Examples:

- **unsigned**
- **signed**
- **short**
- **long**

```
short int VARA;
long int VARB;
long long int VARC;
```

ARM:
2 Bytes
4 Bytes
8 Bytes

C Keywords

- Special C-Keywords can affect data allocation
  - Variable Types
  - Type Qualifiers
  - Type Modifiers
  - Storage Classes

Storage Classes specify lifetime and scope of a data type

Examples:

- **Auto**
- **Static**
- **Extern**
- **Register**

```
auto int VARA;
static int VARB;
extern int VARC;
register int VARD;
```

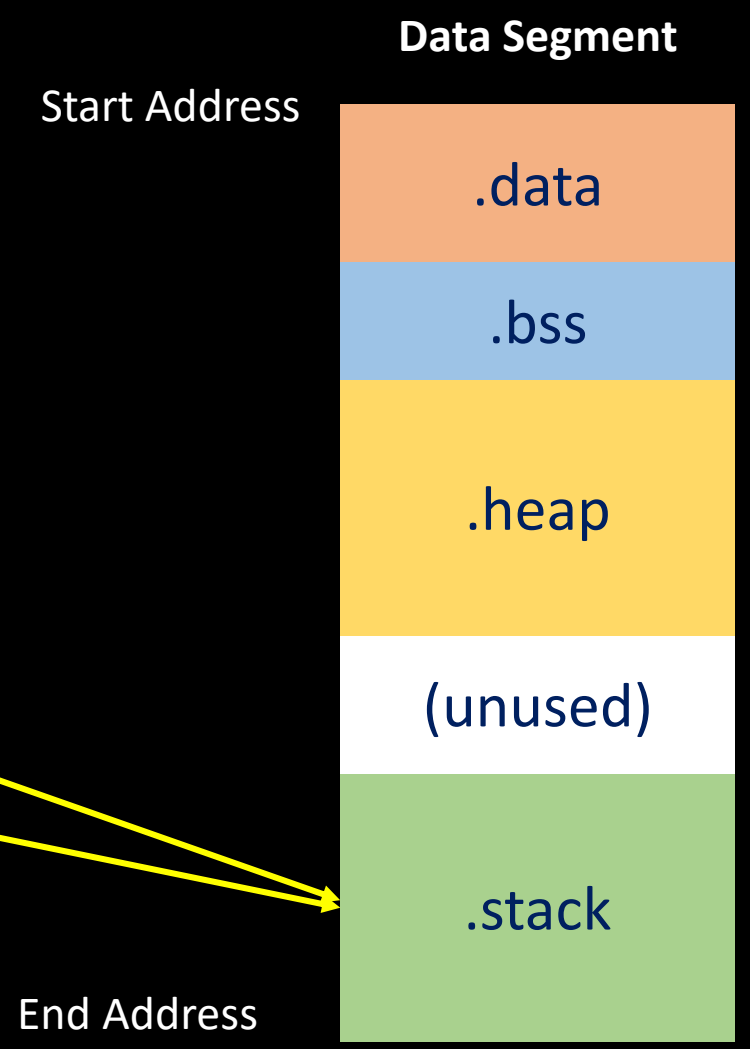
Auto Keyword

- Automatically allocated and deallocated data on the stack
  - Has a lifetime of a function or block

```
void foo(){
  auto int vara;
  int varb;

  /* Other Code */

  return;
}
```



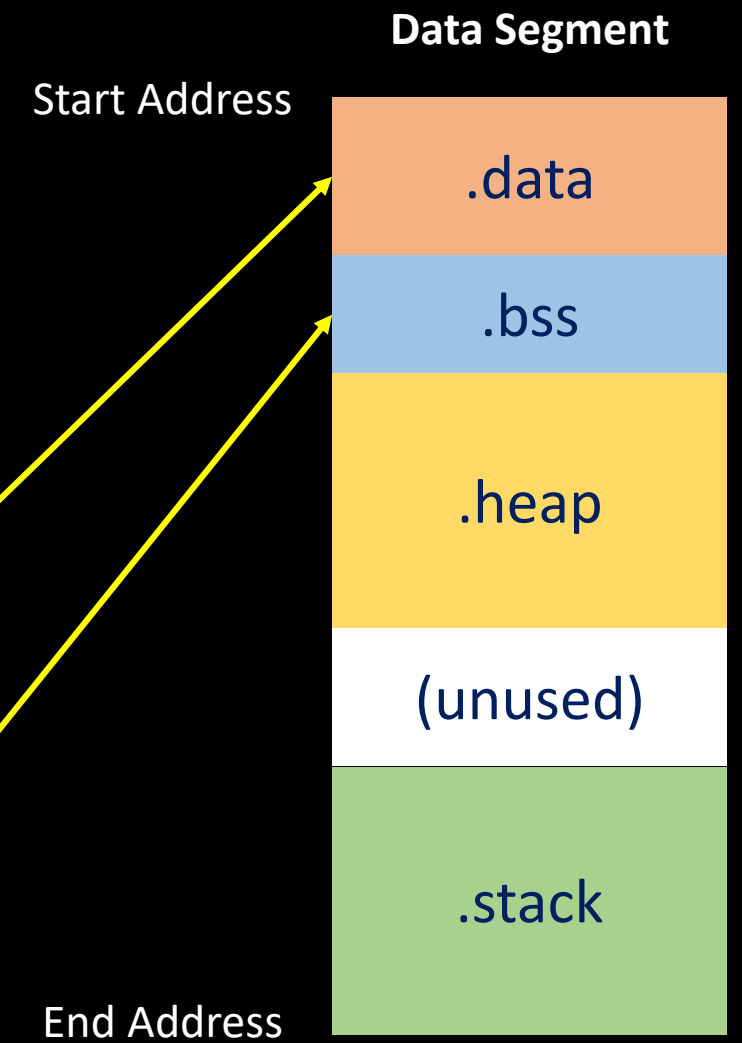
Static Keyword

- Data will persist in memory until the end of the program

- Static data can get stored in both .data or .bss

```
static int VARA = 1;

static char VARB;
static int VARC = 0;
```



Extern Keyword

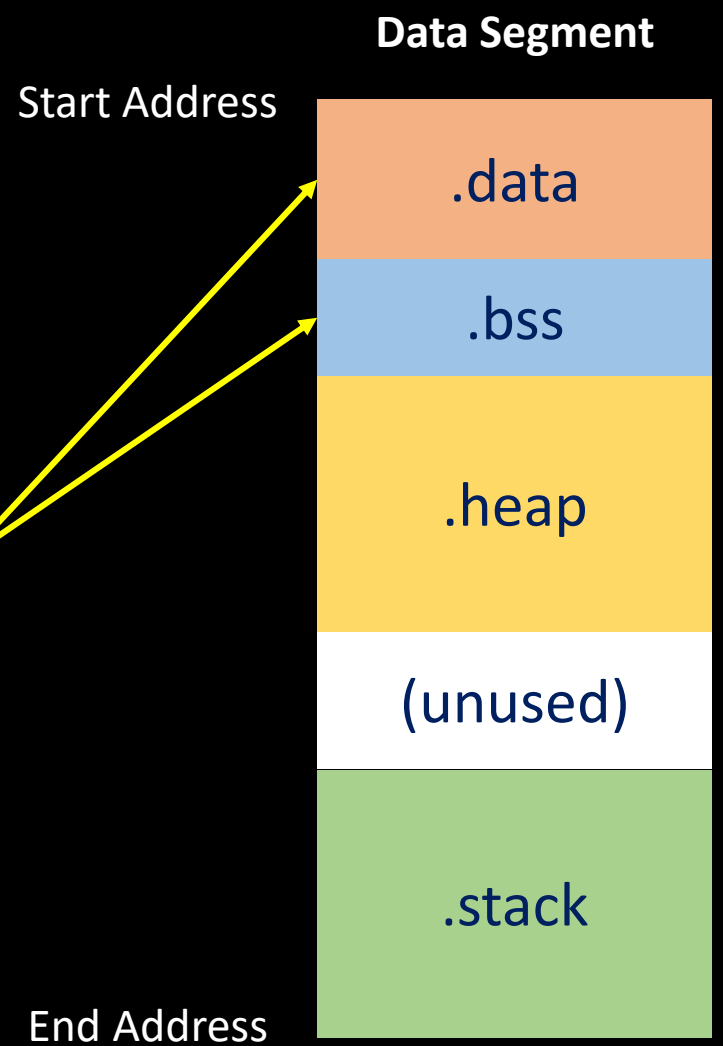
- Declares a global reference defined in another file to be visible by current file
  - Can be bss or data
  - Initial definition must be a global variable

```
extern int VARA;
extern char VARB;
extern int VARC;
```

File1.c

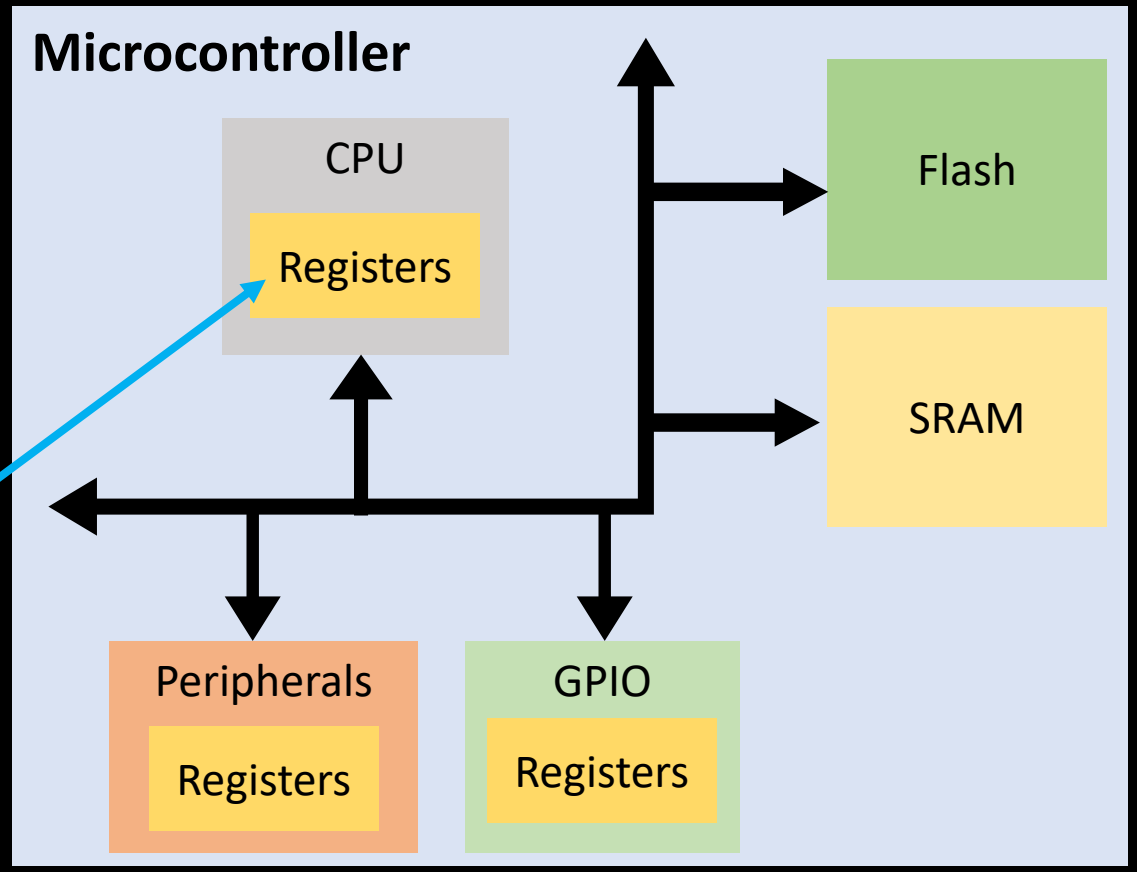
```
int VARA = 1;
char VARB;
int VARC = 0;
```

File2.c



Register Keyword

- Allocates Data directly in the CPU
  - Used for repeated variable use with high speed
  - Not a guaranteed
  - Not commonly used




```
register int VAR = 1;
```

Data Segment

- Stack: Temporary Data Storage like local variables
- Heap: Dynamic data storage
- Data: Non-Zero Initialized global and static data
- BSS: Zero initialized and Uninitialized global and static data

```
int A_BSS;
int B_BSS = 0;
int C_DATA = 1;
const int D_RODATA = 1;

void foo(int D_STACK_REG){
  int F_STACK_REG;
  int G_STACK_REG = 1;
  static int H_BSS;
  static int I_BSS = 0;
  static int J_DATA = 1;
  char * ptr_STACK_REG;
  ptr_STACK_REG = (char *)malloc(8);

  /* More Code Here */

  free((void *)ptr_STACK_REG);

  return;
}
```