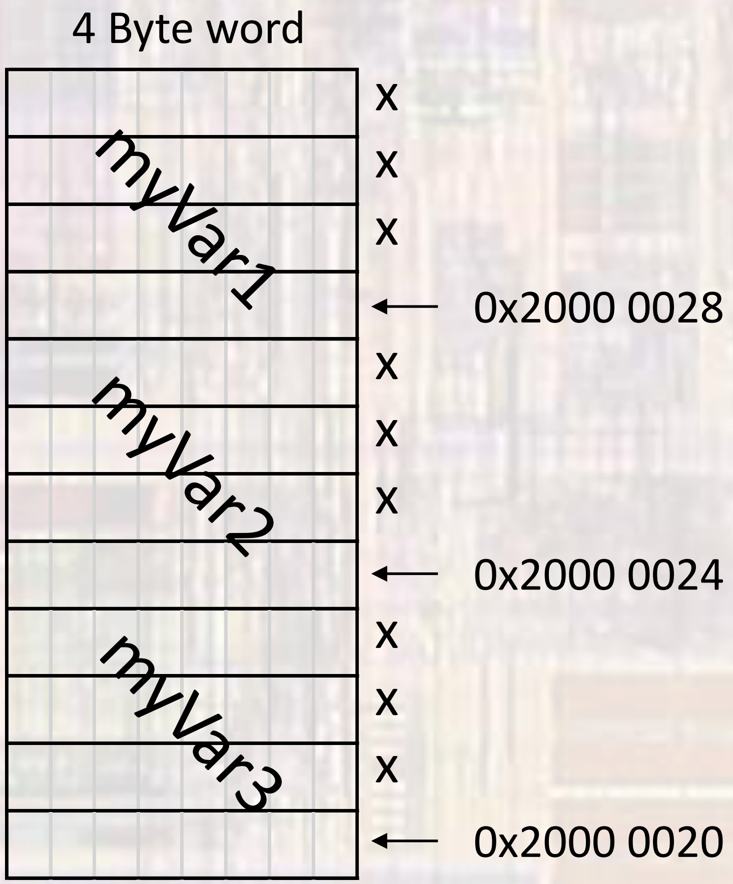
Pointer Basics

Last updated 6/16/23

These slides introduce pointers

Pointer Basics

- Pointer
  - Review variables in memory (stack)
    - Remember – the stack grows down



- address for myVar1

  0x2000 0028

- address for myVar2

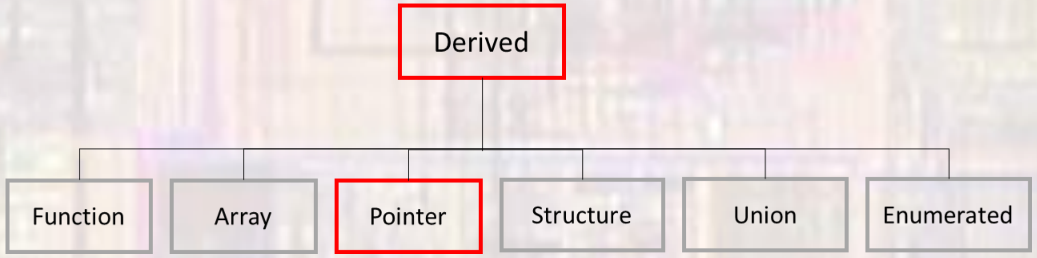
  0x2000 0024

- address for myVar3

  0x2000 0020

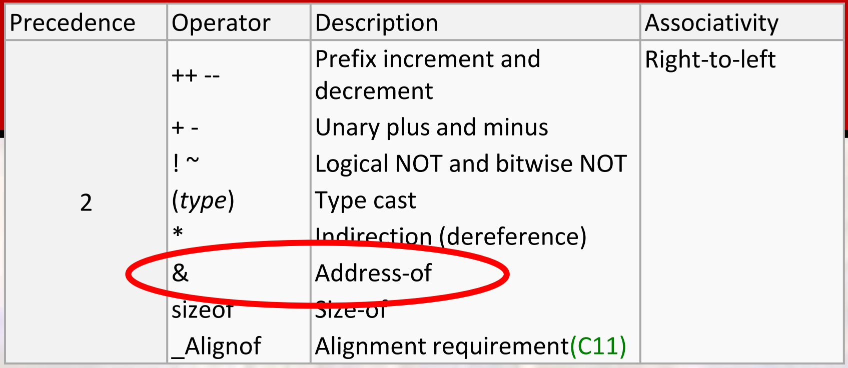
Pointer Basics

- Pointer



  - A special Type
  - A variable that holds the memory location of another variable
  - Holds an address – in our case 32 bits
    - All pointer variables are the same size
  - Each pointer must be tied to a specific data type
    - int, float, char, …

Pointer Basics



Precedence	Operator	Description	Associativity
2	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of	
	_Alignof	Alignment requirement(C11)	

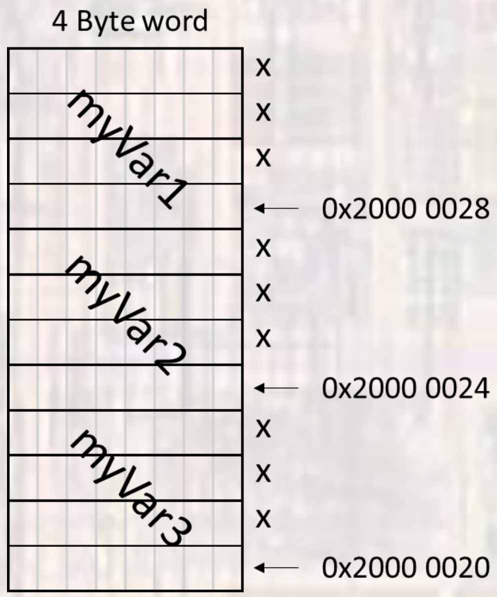
- Pointer – Address of

  - To find the memory location of a variable, use the address of operator: &

&myVar1 → 0x2000 0028

&myVar2 → 0x2000 0024

&myVar3 → 0x2000 0020



Pointer Basics

- Pointer - Declaration

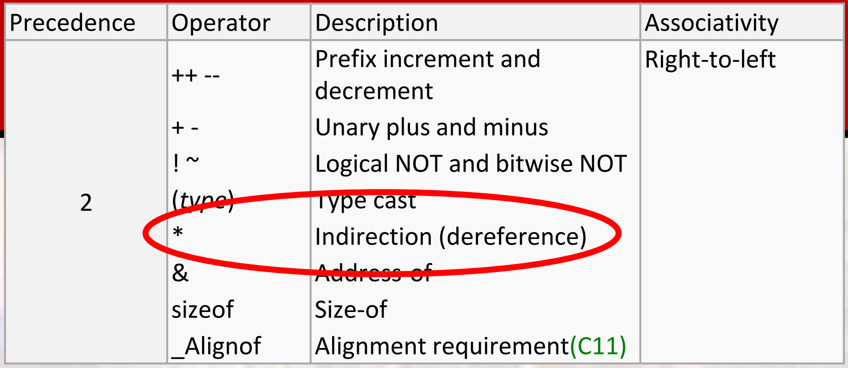
  - To declare a pointer variable
    - follow the type declaration with a *

The _ptr is for clarification
It is not required – but it does remind you that this variable is a pointer variable

```
int * myVar1_ptr;
// declare a pointer variable with name myVar1_ptr
// that holds the memory location of an integer variable

float * myVar2_ptr;
// declare a pointer variable with name myVar2_ptr
// that holds the memory location of a float variable
```

Note: int * my_ptr, int* my_ptr, and int *my_ptr all work – the location of the space is not critical

Pointer Basics



Precedence	Operator	Description	Associativity
2	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	Type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of	
	_Alignof	Alignment requirement(C11)	

- Pointer - Dereference
  - To determine the value of a variable pointed to by a pointer variable
    - precede the pointer variable with * (dereference operator)

```
int * myVar1_ptr;
float * myVar2_ptr;
```

```
*myVar1_ptr;
// provides the value held in the memory location
// pointed to by myVar1_ptr  (as an int)

*myVar2_ptr;
// provides the value held in the memory location
// pointed to by myVar2_ptr  (as a float)
```

Pointer Basics

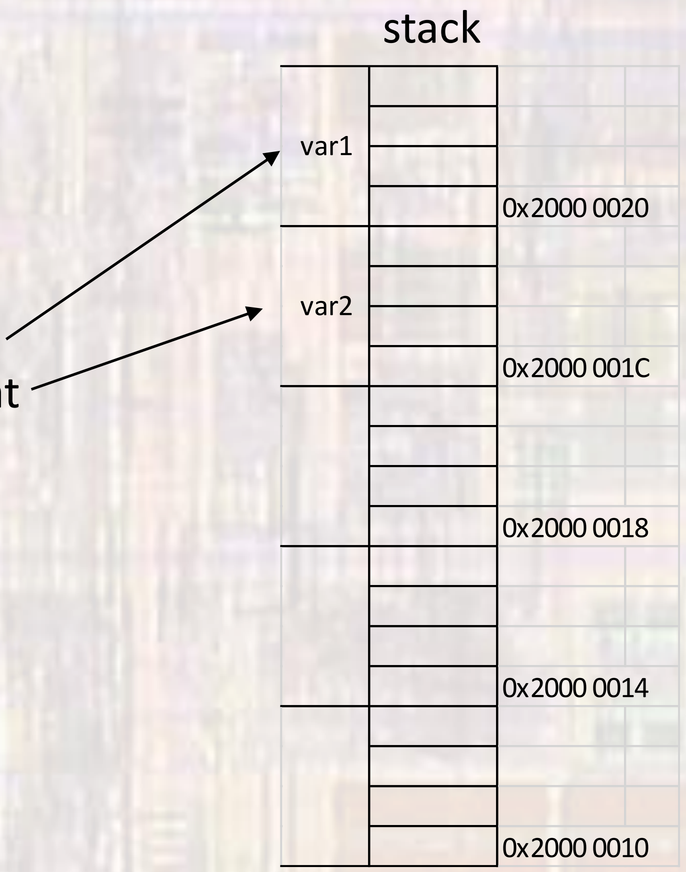
- Pointer - terminology
  - It helps to remember what is happening if we use specific terminology
  - Read ‘int * foo_ptr’ as ‘pointer variable to a variable of type int’
  - Read ‘&foo’ as ‘the address of foo’
  - Read ‘*foo_ptr’ as ‘the value pointed to by foo_ptr’

The _ptr is for clarification. It is not required – but it does remind you that this variable is a pointer variable

Pointer Basics

- Pointers in Memory

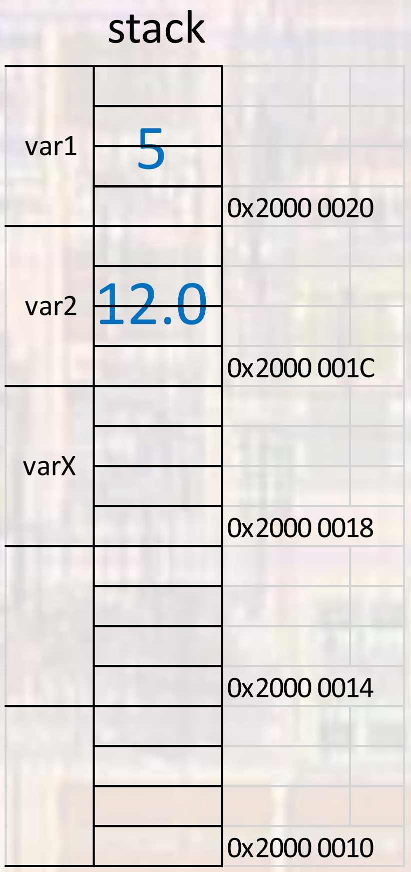
```
int foo1;        // stored earlier so not visible
float foo2;      // in this section of the stack


int var1;        // declare a variable of type int
float var2;      // declare a variable of type float
```



Pointer Basics

- Pointers in Memory

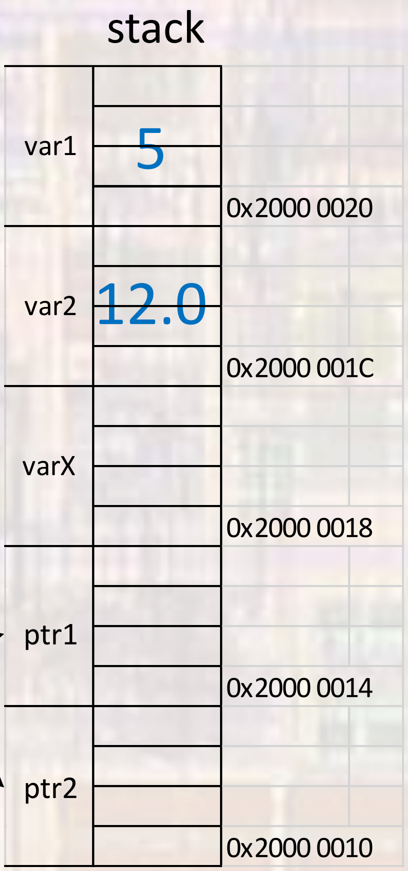
```
int foo1;           // stored earlier so not visible
float foo2;         // in this section of the stack


int var1;           // declare a variable of type int
float var2;         // declare a variable of type float

var1 = 5;           // assign 5 to var1 (0x2000 0020)
var2 = 12.0;        // assign 12.0 to var2 (0x2000 001C)
```



Pointer Basics

- Pointers in Memory

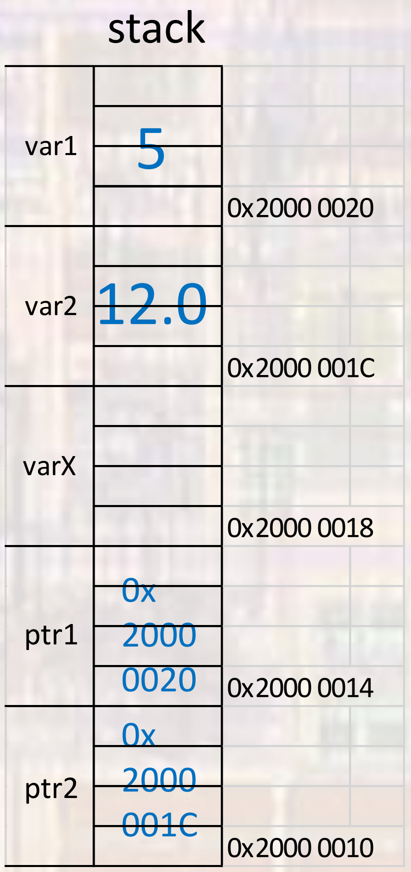
```
int foo1;        // stored earlier so not visible
float foo2;      // in this section of the stack


int var1;        // declare a variable of type int
float var2;      // declare a variable of type float

var1 = 5;        // assign 5 to var1 (0x2000 0020)
var2 = 12.0;     // assign 12.0 to var2 (0x2000 001C)

int * ptr1;      // declare a pointer variable to a variable of type int
float * ptr2;    // declare a pointer variable to a variable of type float
```



Pointer Basics

- Pointers in Memory

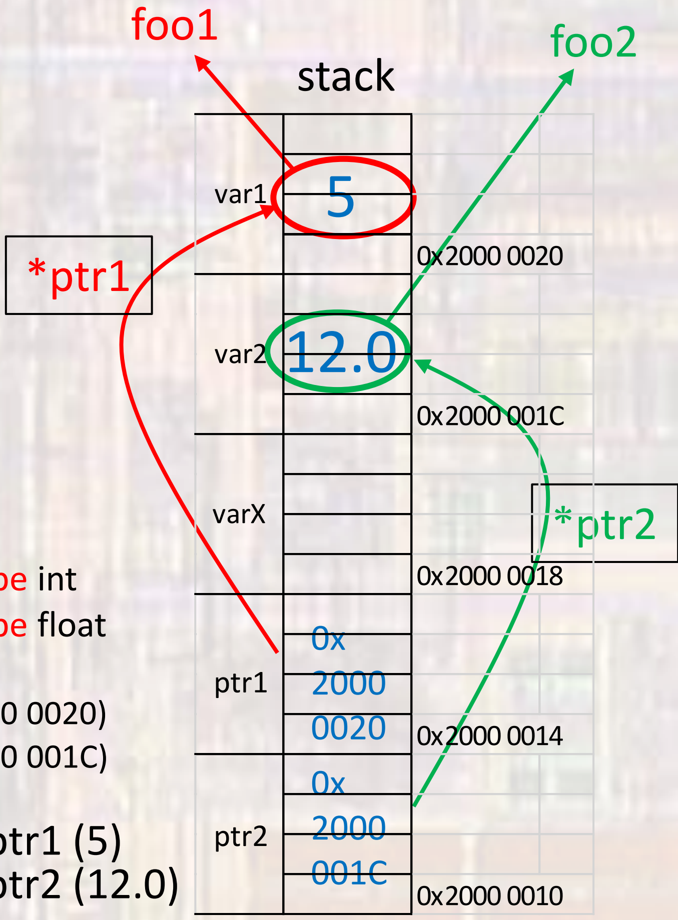
```
int foo1;           // stored earlier so not visible
float foo2;         // in this section of the stack


int var1;           // declare a variable of type int
float var2;         // declare a variable of type float

var1 = 5;           // assign 5 to var1 (0x2000 0020)
var2 = 12.0;        // assign 12.0 to var2 (0x2000 001C)

int * ptr1;         // declare a pointer variable to a variable of type int
float * ptr2;       // declare a pointer variable to a variable of type float

ptr1 = &var1;       // set ptr1 to the address of var1 (0x2000 0020)
ptr2 = &var2;       // set ptr2 to the address of var1 (0x2000 001C)
```



Pointer Basics

- Pointers in Memory

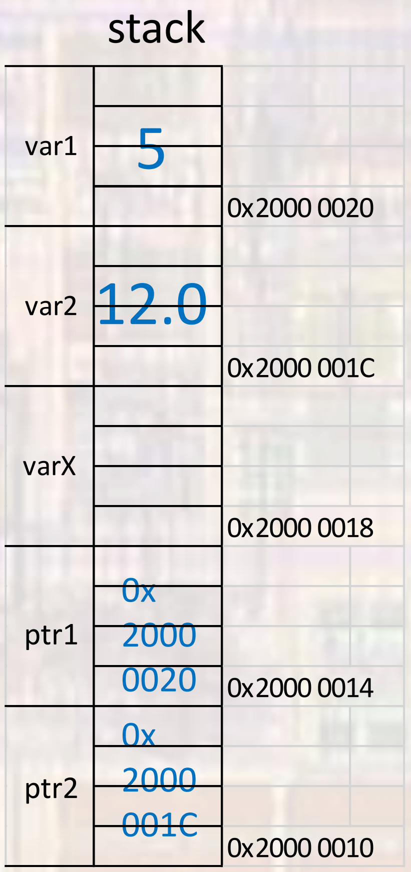
```
int foo1;          // stored earlier so not visible
float foo2;        // in this section of the stack

int var1;          // declare a variable of type int
float var2;        // declare a variable of type float

var1 = 5;          // assign 5 to var1 (0x2000 0020)
var2 = 12.0;       // assign 12.0 to var2 (0x2000 001C)

int * ptr1;        // declare a pointer variable to a variable of type int
float * ptr2;      // declare a pointer variable to a variable of type float

ptr1 = &var1;      // set ptr1 to the address of var1 (0x2000 0020)
ptr2 = &var2;      // set ptr2 to the address of var1 (0x2000 001C)

foo1 = *ptr1;      // set foo1 to the value pointed to by ptr1 (5)
foo2 = *ptr2;      // set foo2 to the value pointed to by ptr2 (12.0)
```



Pointer Basics

- Pointers in Memory

```
int foo1;
float foo2;

int var1;              // declare a variable of type int
float var2;            // declare a variable of type float

var1 = 5;              // assign 5 to var1 (0x2000 0020)
var2 = 12.0;           // assign 12.0 to var2 (0x2000 001C)

int * ptr1;            // declare a pointer variable to a variable of type int
float * ptr2;          // declare a pointer variable to a variable of type float

ptr1 = &var1;          // set ptr1 to the address of var1 (0x2000 0020)
ptr2 = &var2;          // set ptr2 to the address of var1 (0x2000 001C)

foo1 = *ptr1;          // set foo1 to the value pointed to by ptr1 (5)
foo2 = *ptr2;          // set foo2 to the value pointed to by ptr2 (12.0)

Note:     &ptr1        // the address of ptr1 (0x2000 0014)
          &ptr2        // the address of ptr2 (0x2000 0010)
```


stack
var1
5
0x2000 0020
var2
12.0
0x2000 001C
varX
0x2000 0018
ptr1
0x
2000
0020
0x2000 0014
ptr2
0x
2000
001C
0x2000 0010