

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Introduction to Pointers – Part 1

Quick Recap of Relevant Topics



- Basic programming constructs
- Variables and basic data types
 - int, float, double, char, bool, void ...
- Arrays and matrices
- Programs to solve some interesting problems

Variables: Named memory locations Memory locations accessed through names

Overview of This Lecture



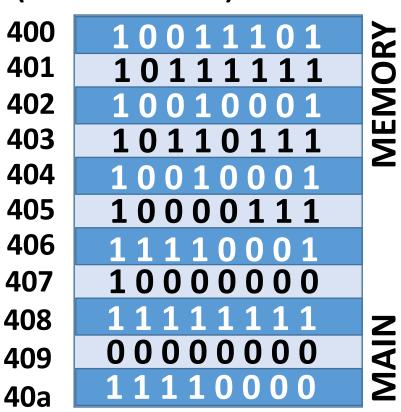
- Addresses of memory locations
- "Address of" operator in C++
- Pointer data type in C++
- Motivate accessing memory locations through addresses

Memory and Addresses



Address (in hexadecimal)

- Main memory is a sequence of physical storage locations
- Each location stores 1 byte (8 bits): Content/value of location
- Each physical memory location identified by a unique address
 - Index in sequence of memory locations



Memory For Executing A Program (Process)

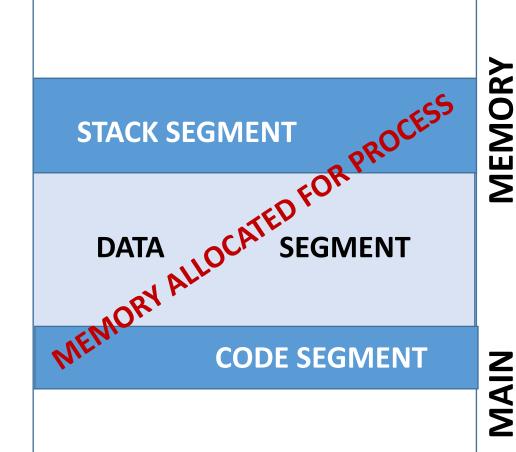


- Operating system allocates a part of main memory for use by a process
- Divided into:

Code segment: Stores executable instructions in program

Data segment: For dynamically allocated data (later lecture)

Stack segment: Call stack



Program Variables and Memory



```
int main()
                     Named drawers of Dumbo
 int a;
 float b;
 char c;
                    Named locations in stack segment
 // Rest of code
            What are their addresses in memory?
```

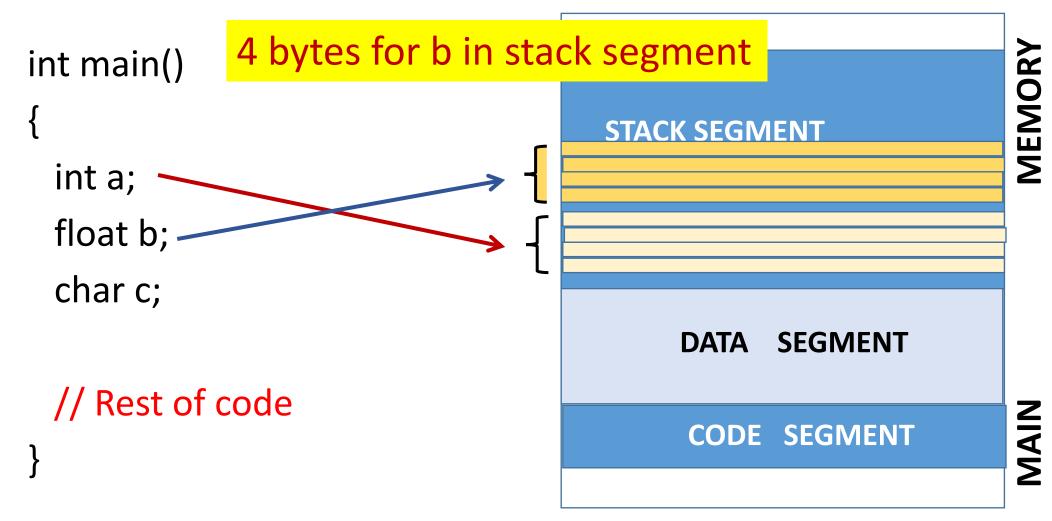


```
int main()
       4 bytes for a in stack segment stack segment
 int a; Each addressable memory location stores 1 byte
 float b;
 char c;
                                       DATA SEGMENT
 // Rest of code
                                        CODE SEGMENT
```

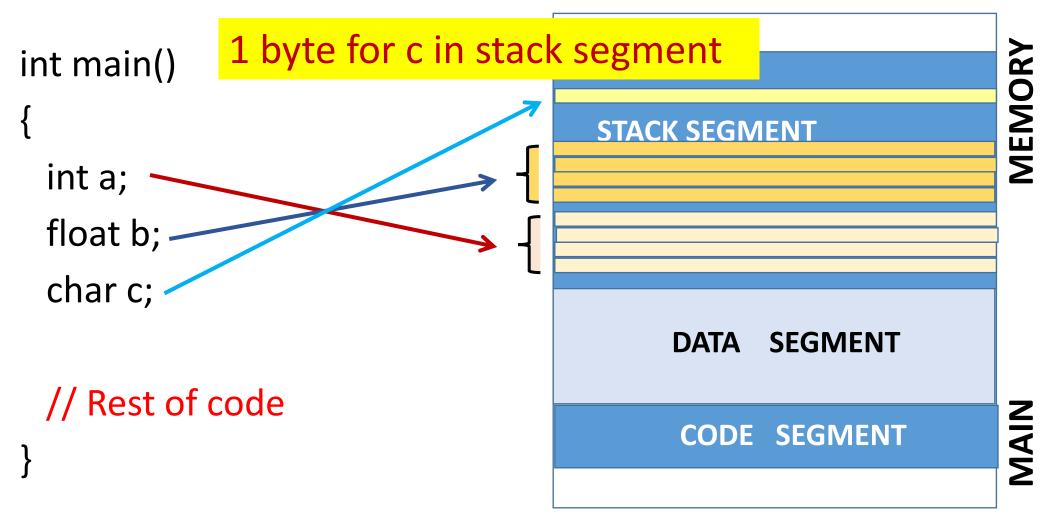


```
int main()
                                    STACK SEGMENT
 int a;
 float b;
 char c;
                                         DATA
                                               SEGMENT
 // Rest of code
                                          CODE SEGMENT
```

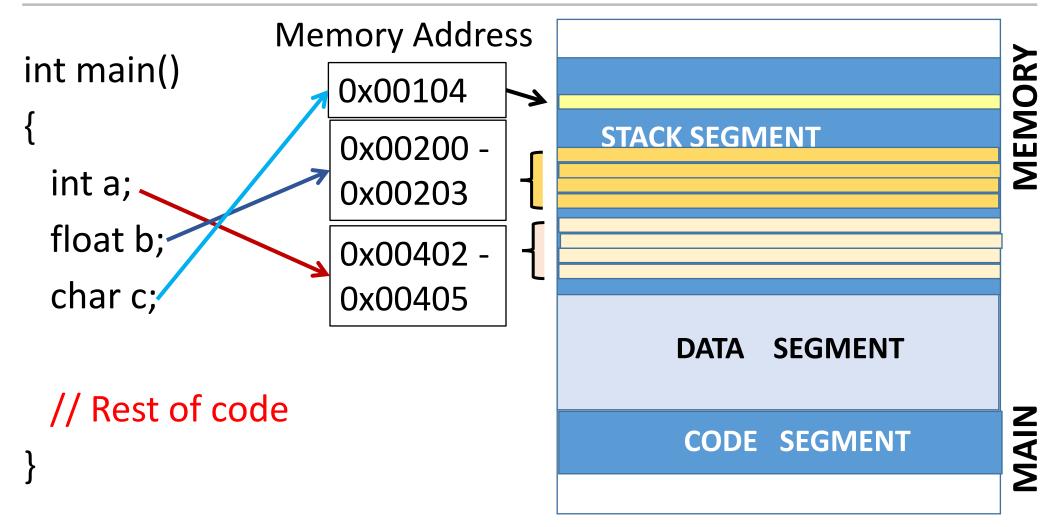




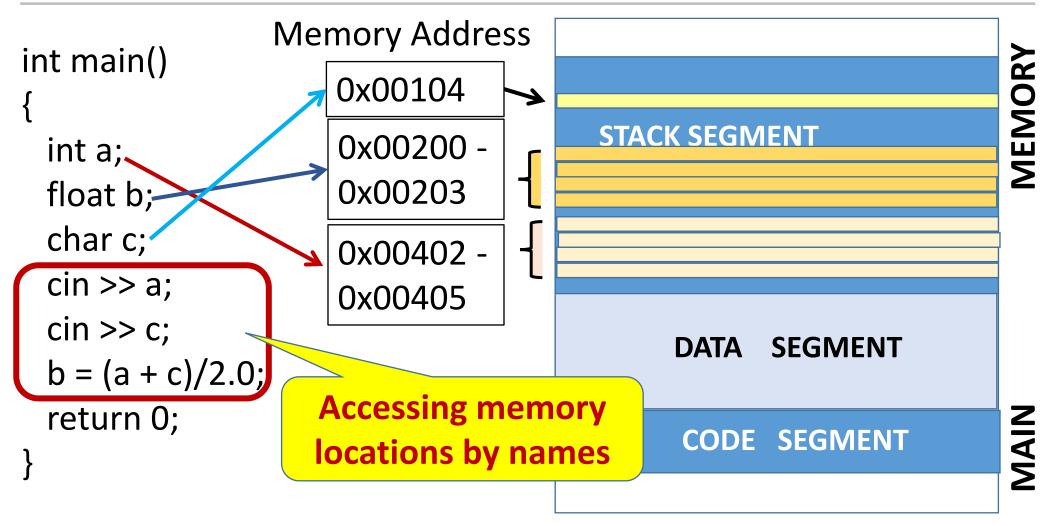




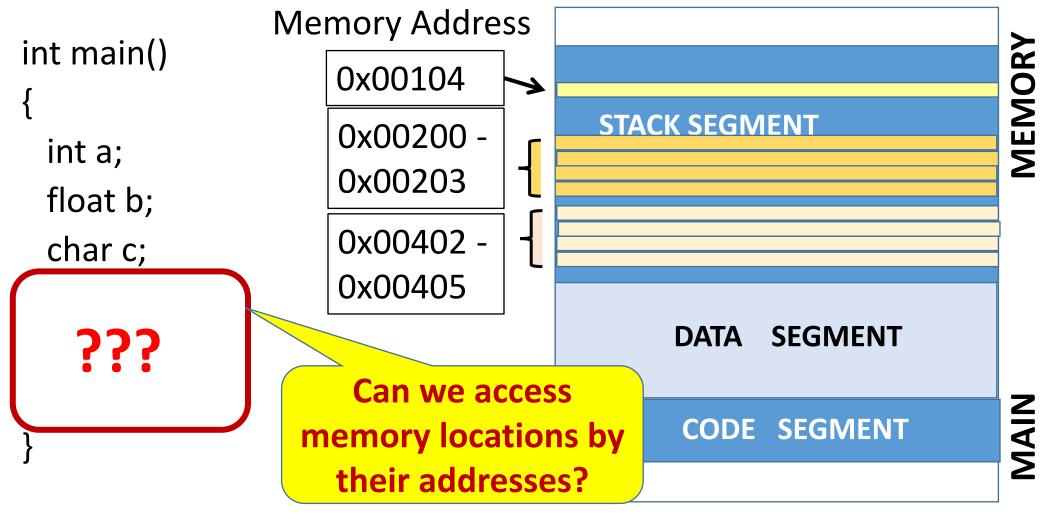




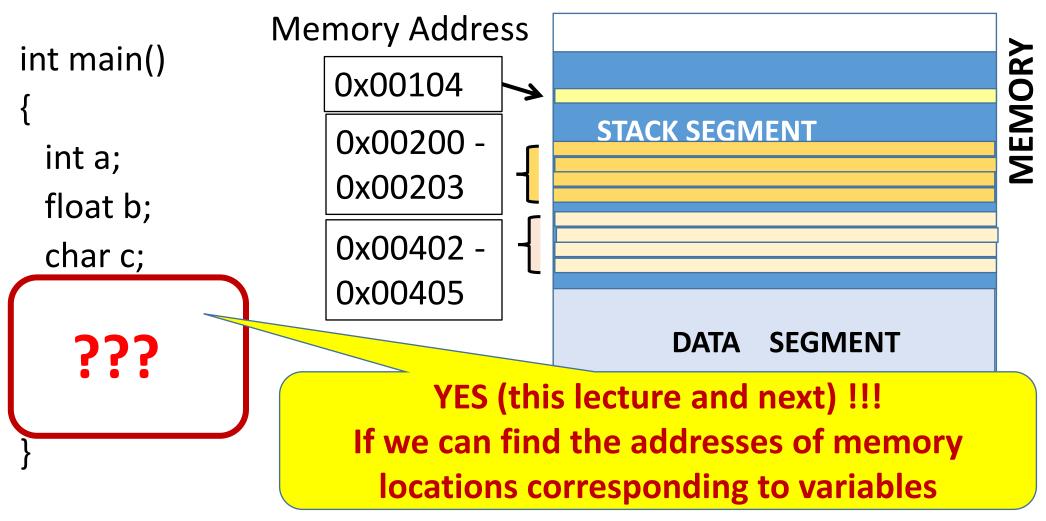












How Do We Get an Address?



- C++ provides an "address of" operator: unary &
 - If "a" is a program variable, "&a" gives address of "a" in memory
 - Unary operator: Takes a single argument
 - "&a" is a C++ expression

Worry about operator precedence, associativity???

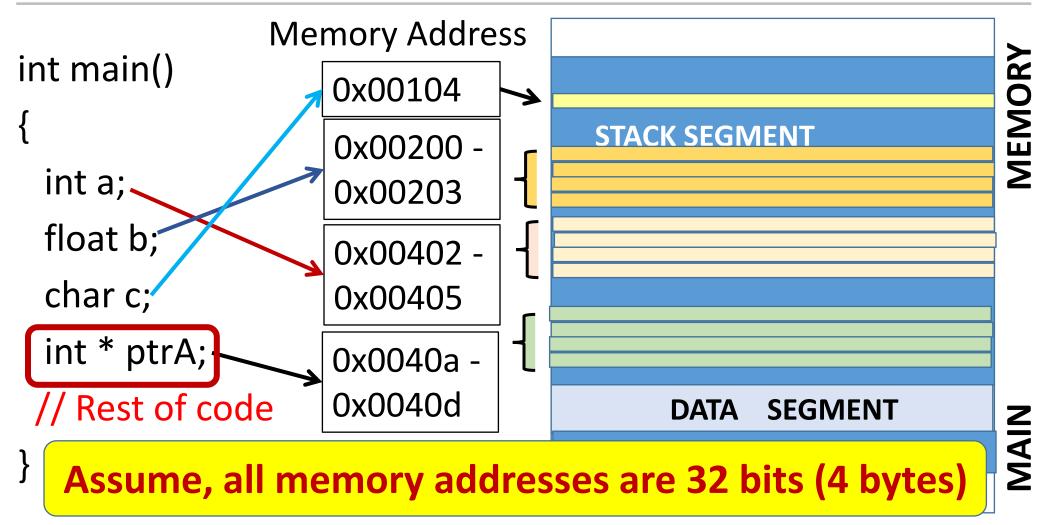
Simplify life: use parentheses

Pointer as a Type

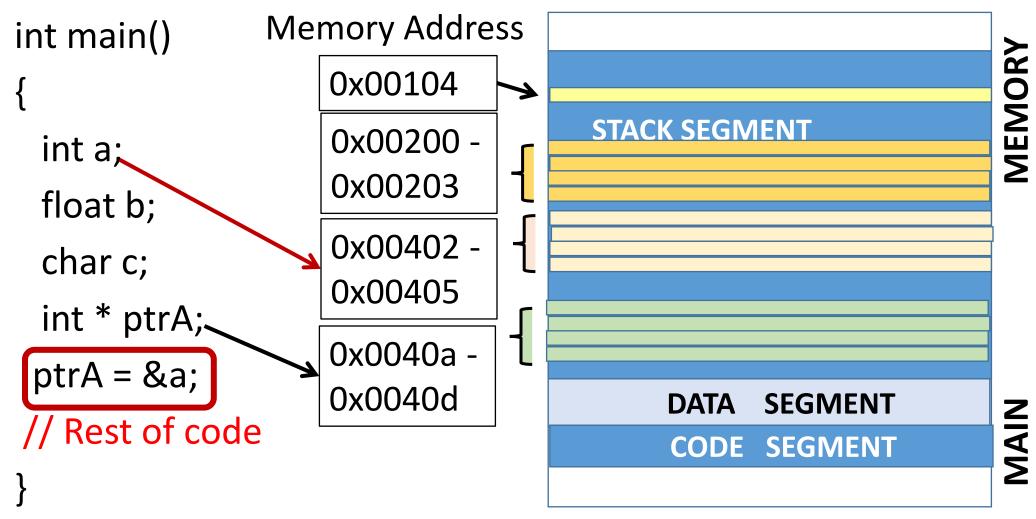


- If "a" is an int variable, what is the type of "&a"
 - "Pointer to int", written in C++ as the type "int *"
- If "b" is a float variable, type of "&b" is "pointer to float", written as "float *"
- In general, if "x" is a variable of type "T"
 "T *" is the type "pointer to T"
 "&x" is an expression of type "T *", gives address of "x"
- If "int *" is a type, can't we have a variable of type "int *"?

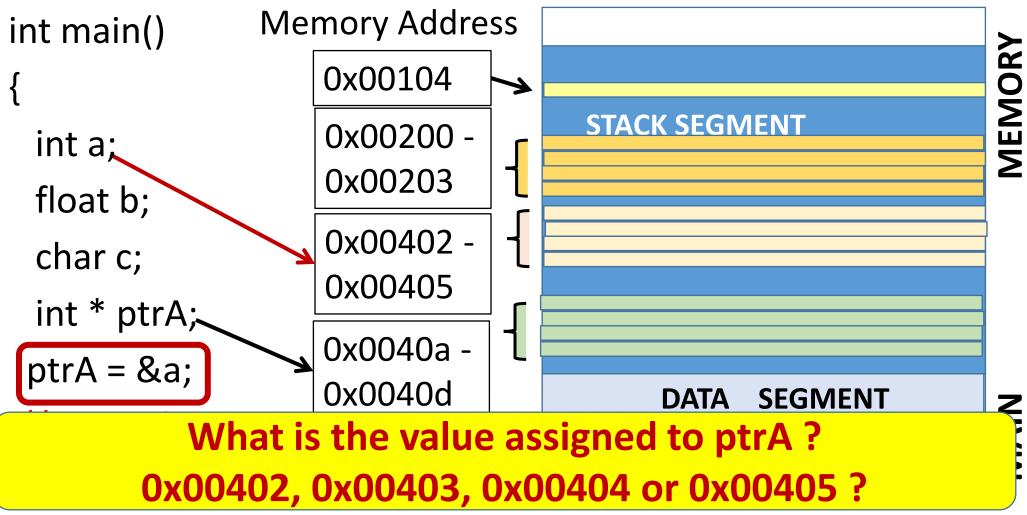




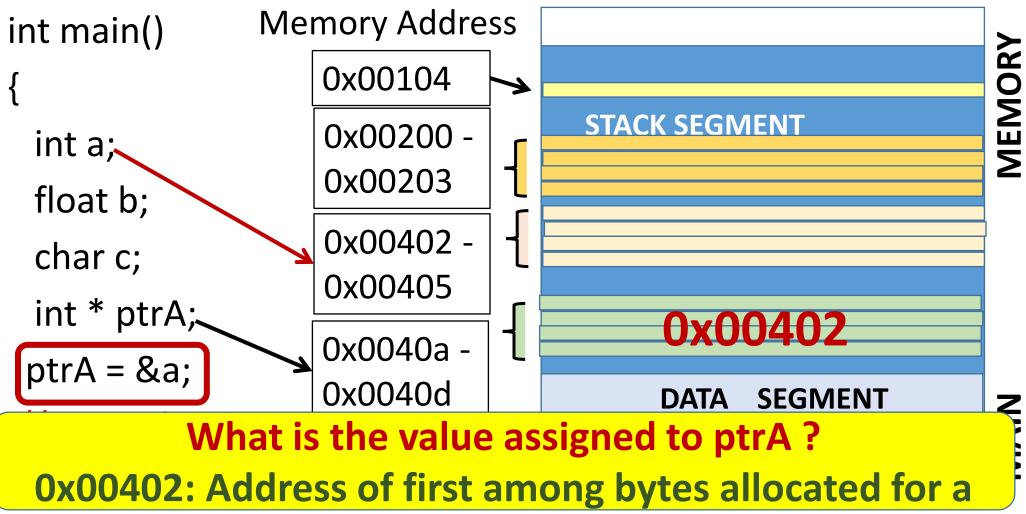






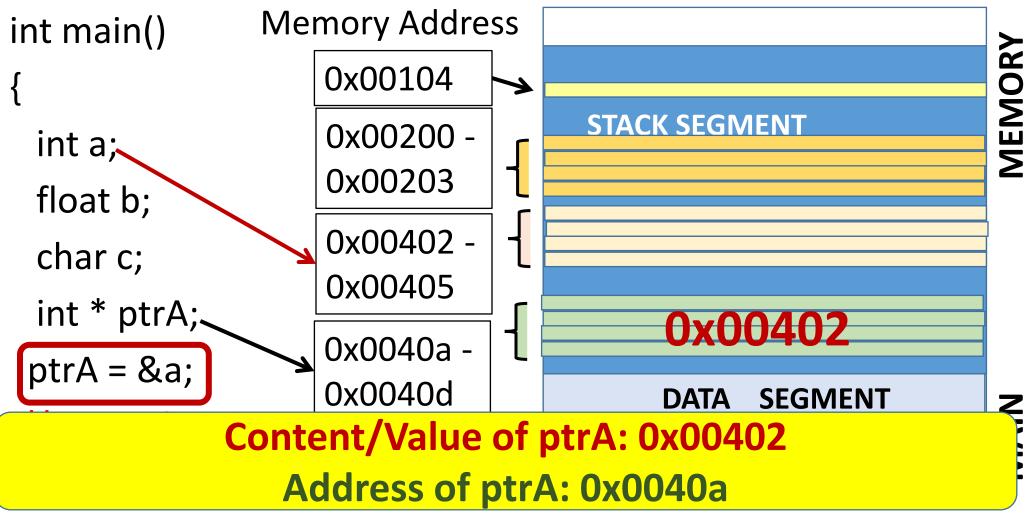






Value and Address of Pointer Variables





Can We Have Pointers to Pointers?



- "ptrA" is a variable of type "int *" (pointer to int)
- What is the type of the expression "&ptrA"?
 Recall: If variable "x" is of type "T", "&x" is of type "T *"
 Type of "&ptrA" is "int **": Pointer to pointer to int
 Note: We don't write "(int *) *", but "int **"
- How far can we take this?

```
As far as we want

'int ***

int ***

is a legitimate pointer type in C++

pointer to pointer to pointer to int
```

A Note About Pointer Declarations



```
int main()
{
         Types of x, y and z: int
    int x, y, z;
    int *a, b, *c;
    // Rest of code
}
```

A Note About Pointer Declarations



```
int main()
                 Type of a: pointer to int
 int x, y, z;
                                    Type of c: pointer to int,
 int *a, b,
                                  not pointer to pointer to int
// Rest of
                     Type of b: int,
                   not pointer to int
```

A C++ Program For Printing Addresses



```
int main()
 int a; float b; char c;
 int * ptrA; float * ptrB; char * ptrC;
 ptrA = &a; cout << "Address of a is: " << ptrA;
 ptrB = &b; cout << "Address of b is: " << ptrB;
 ptrC = &c; cout << Address of c is: " << ptrC;
 return 0;
                      Compile and run this program
                See how memory addresses look like !!!
```

A C++ Program For Printing Addresses



```
int main()
 int a; float b; char c;
 int * ptrA; float * ptrB; char * ptrC;
 ptrA = &a; cout << "Address of a is: " << ptrA;
 ptrB = &b; cout << "Address of b is: " << ptrB;
 ptrC = &c; cout << Address of c is: " << ptrC;
 return 0;
                 What do we do with these addresses?
                     Wait for a few more lectures !!!
```

Summary



- Memory and addresses
- "Address of" operator in C++
- Pointer data type in C++
- Some simple usage in programs