IIT Bombay

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

IIT Bombay

- 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

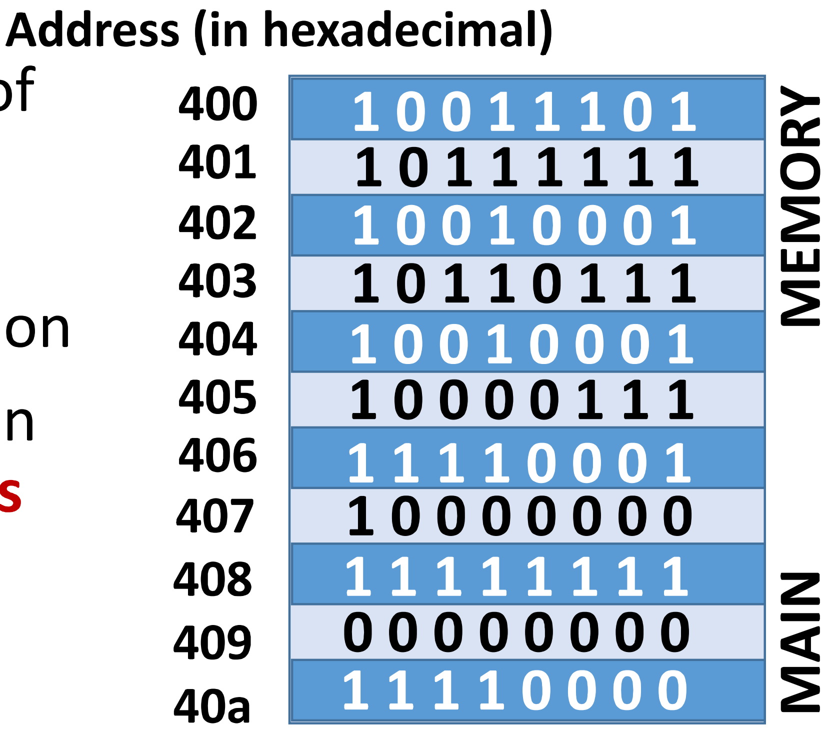
Overview of This Lecture

IIT Bombay

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

Memory and Addresses

- 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**



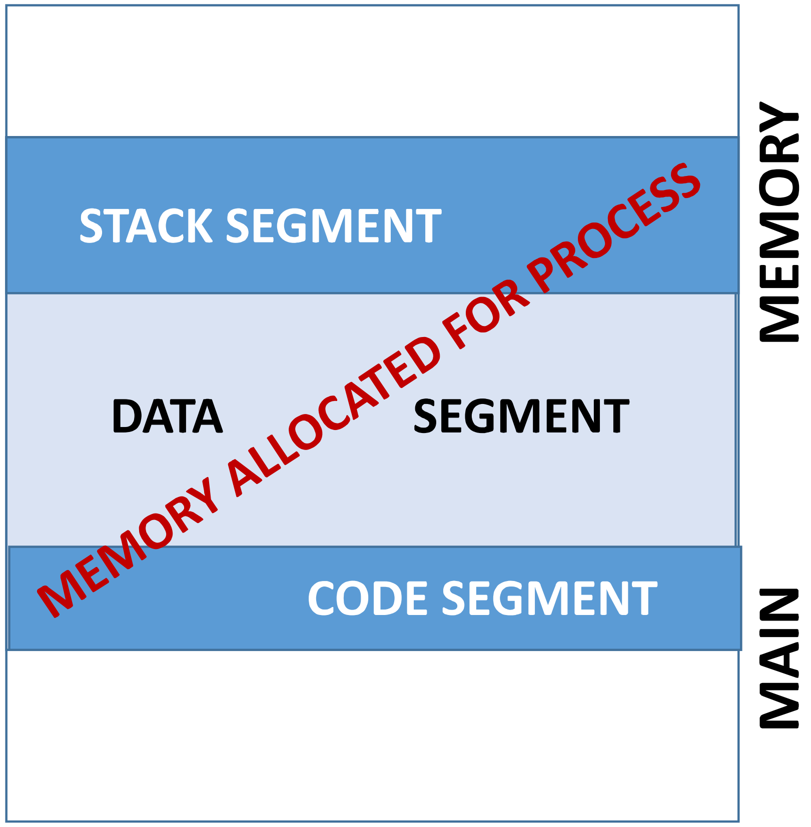
Memory For Executing A Program (Process)

- 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

IIT Bombay

```
int main()
{
  int a;
  float b;
  char c;

  // Rest of code
}
```

Named drawers of Dumbo

Named locations in stack segment

What are their addresses in memory?

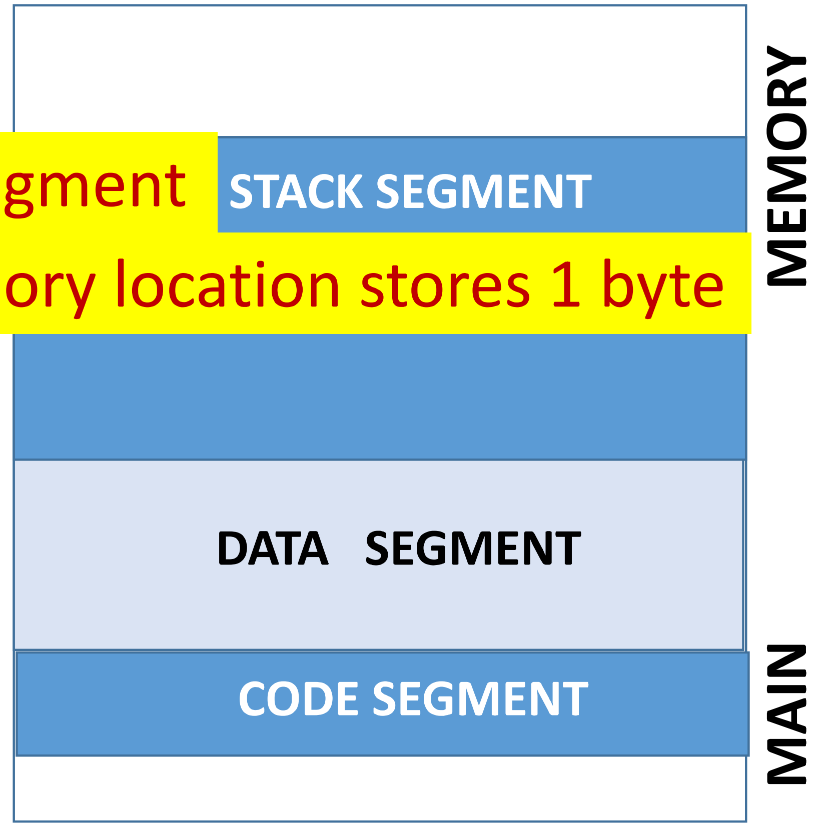
Memory and Program Variables

```
int main()
{
  int a;
  float b;
  char c;

  // Rest of code
}
```

4 bytes for a in stack segment

Each addressable memory location stores 1 byte

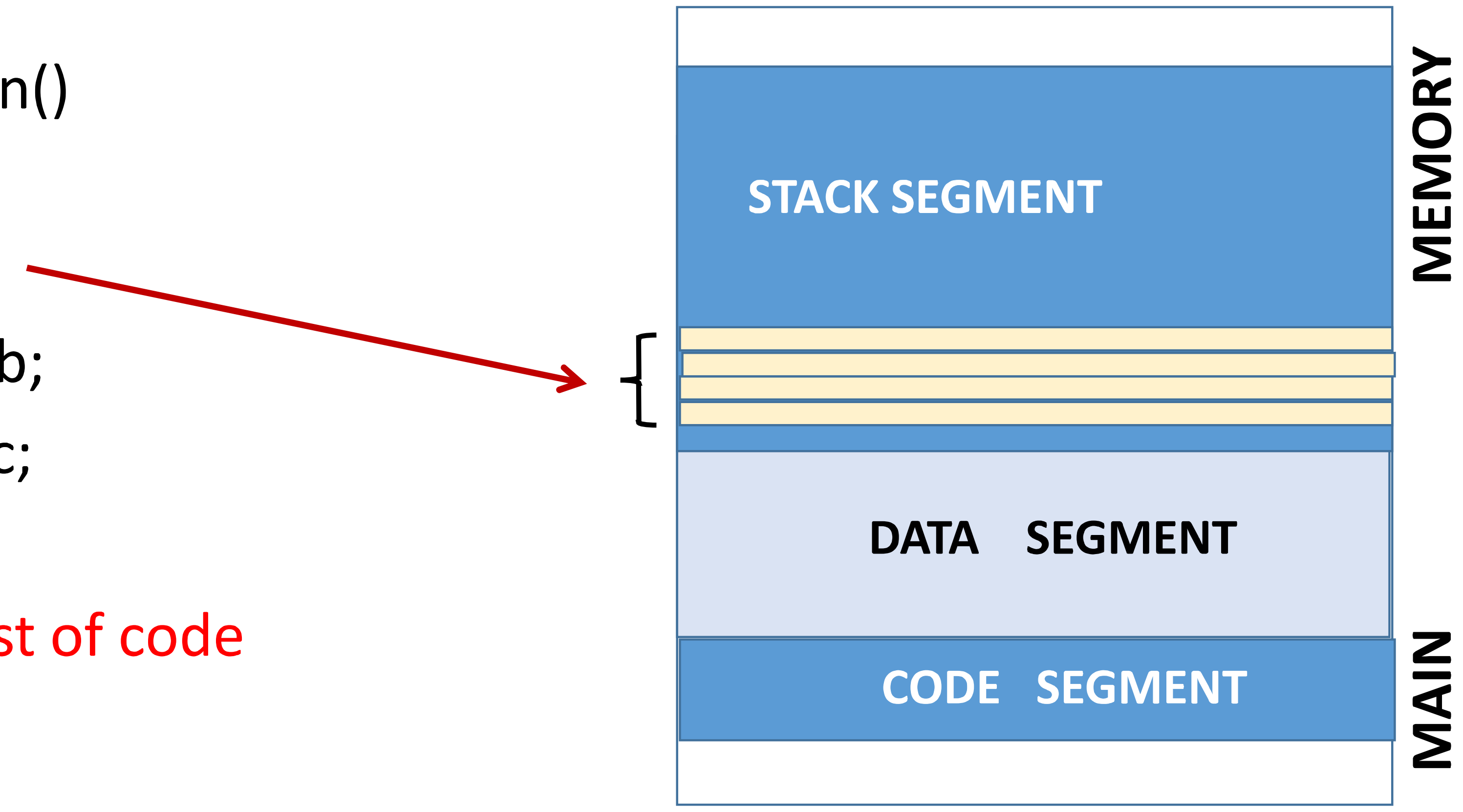


Memory and Program Variables

IIT Bombay

```
int main()
{
   int a;
   float b;
   char c;

   // Rest of code
}
```



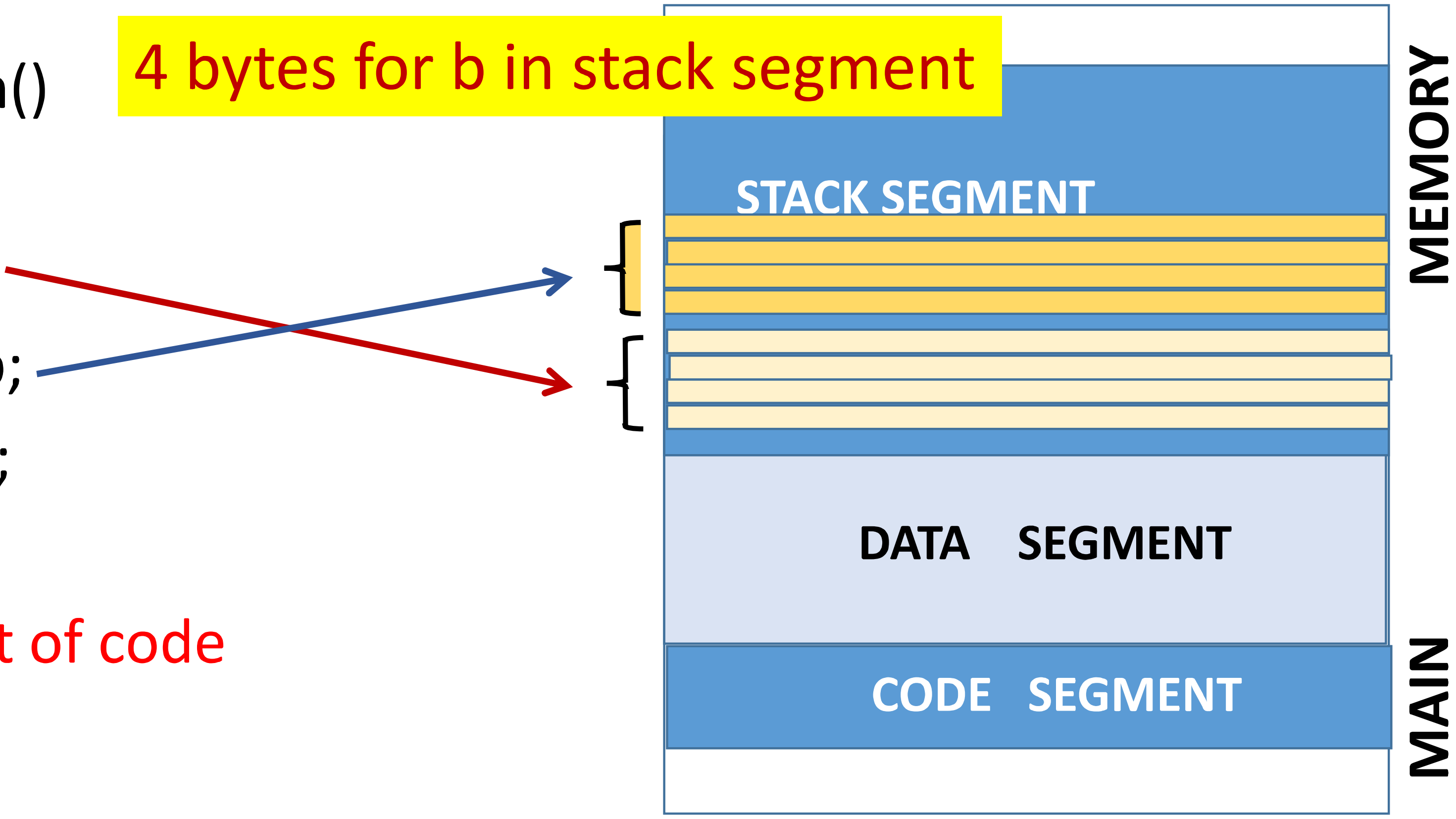
Memory and Program Variables

IIT Bombay

```
int main()
{
   int a;
   float b;
   char c;

   // Rest of code
}
```

4 bytes for b in stack segment



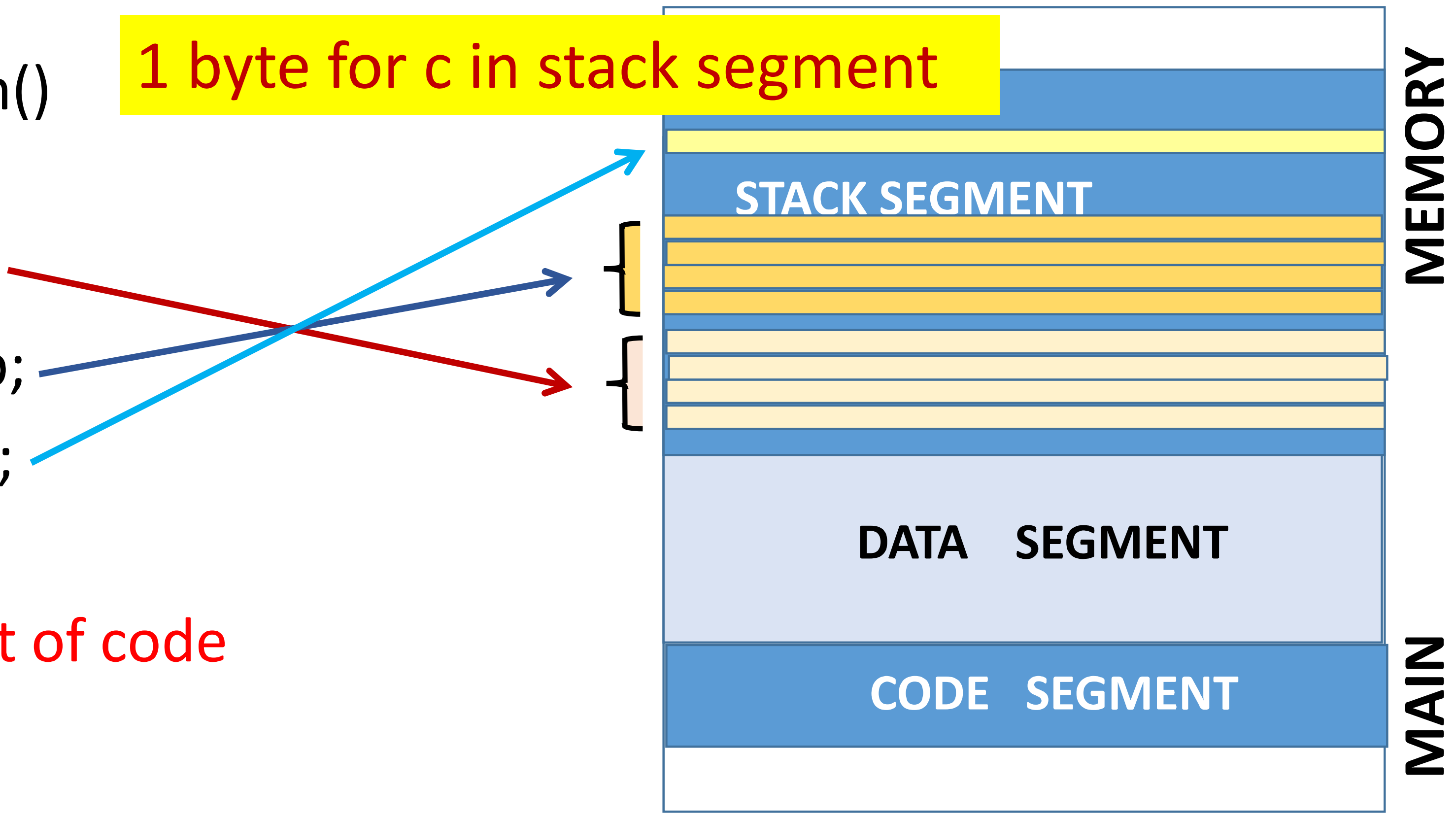
Memory and Program Variables

IIT Bombay

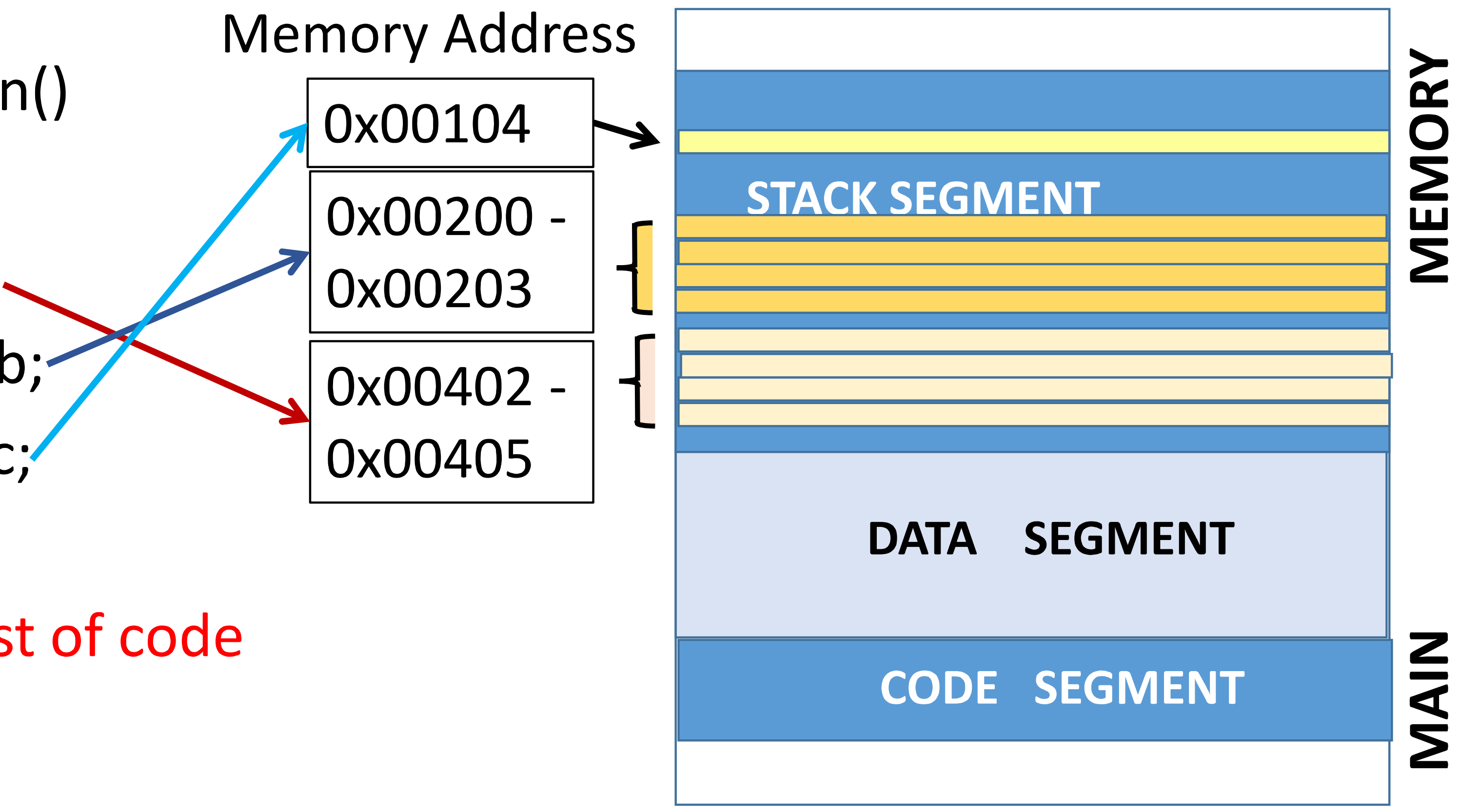
```
int main()
{
   int a;
   float b;
   char c;

   // Rest of code
}
```

1 byte for c in stack segment



Memory and Program Variables

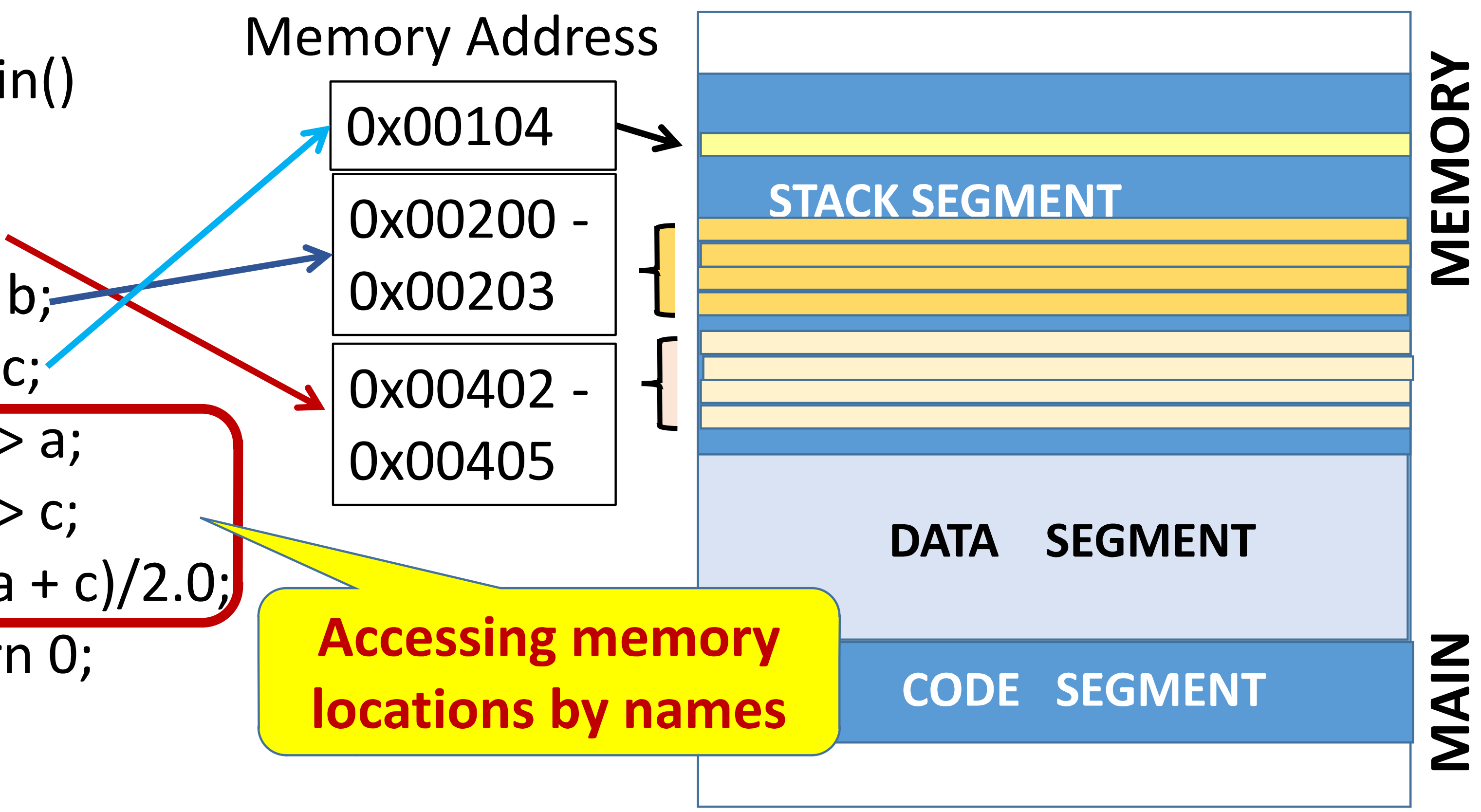


```
int main()
{
   int a;
   float b;
   char c;


   // Rest of code
}
```

Memory and Program Variables

```
int main()
{
   int a;
   float b;
   char c;
   cin >> a;
   cin >> c;
   b = (a + c)/2.0;
   return 0;
}
```

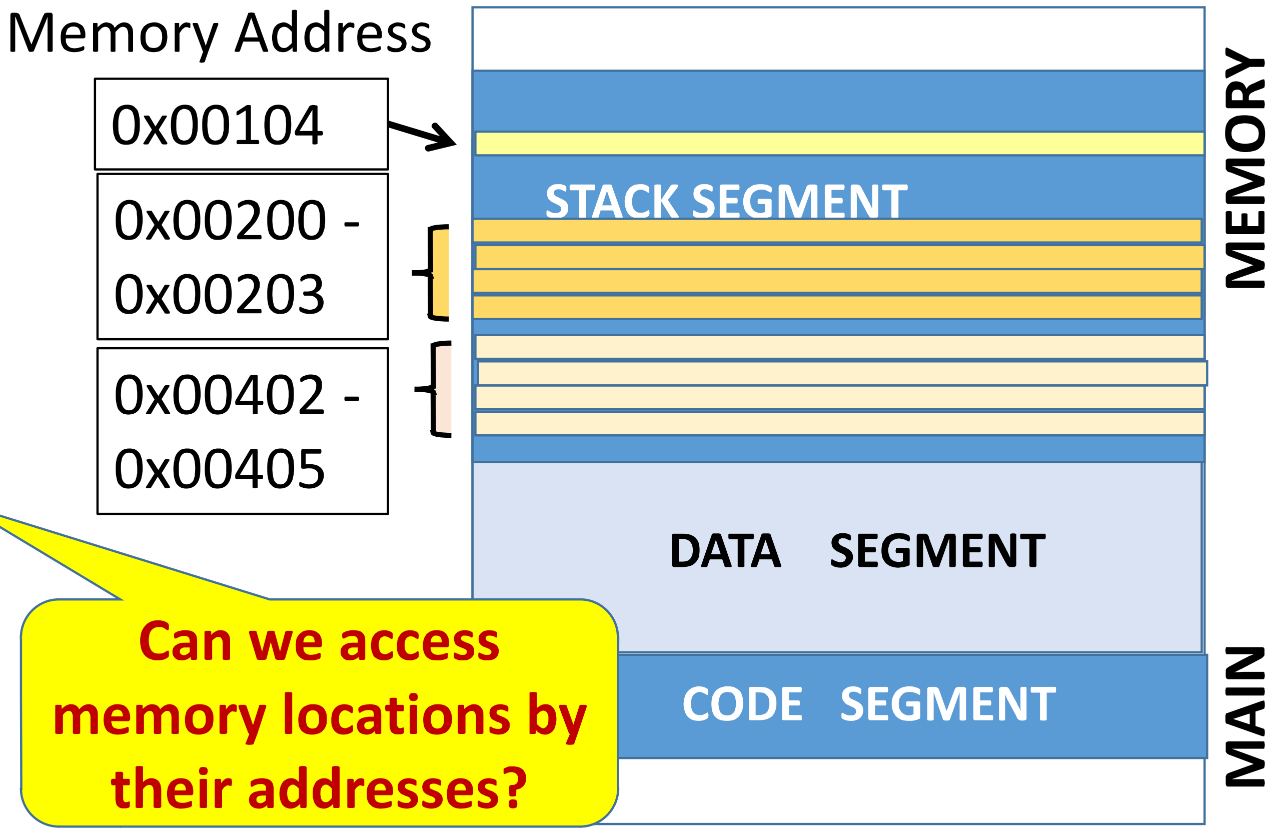


Memory and Program Variables

```
int main()
{
  int a;
  float b;
  char c;
```

???

```
}
```



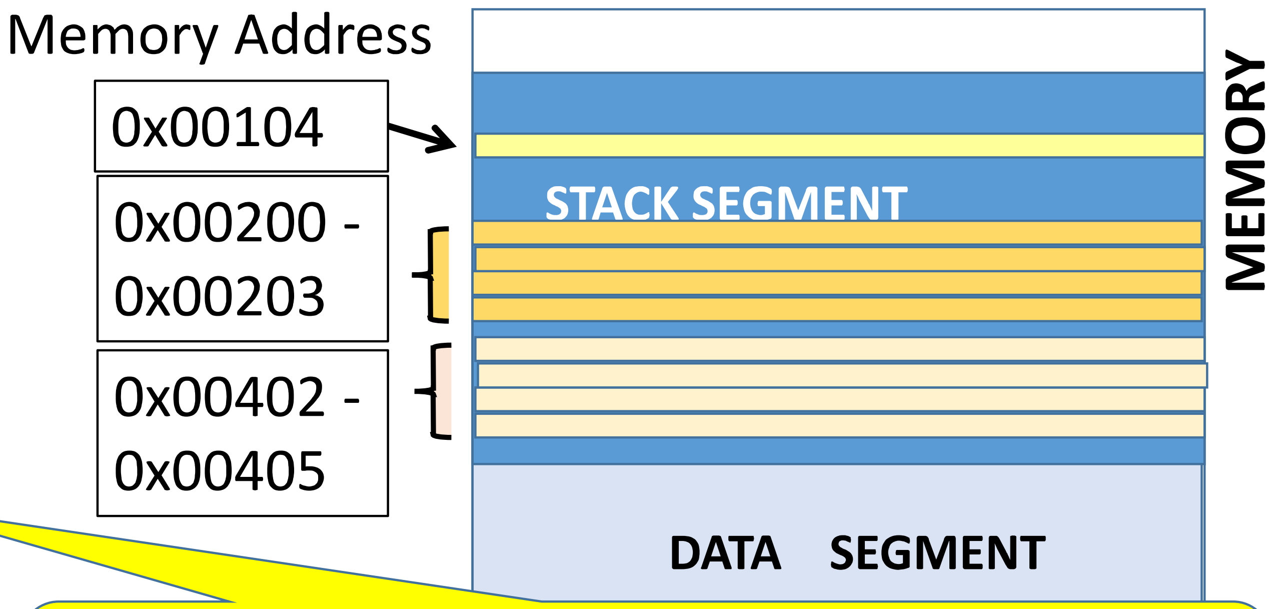
Can we access memory locations by their addresses?

Memory and Program Variables

```
int main()
{
  int a;
  float b;
  char c;
```

???

```
}
```



YES (this lecture and next) !!!
If we can find the addresses of memory locations corresponding to variables

How Do We Get an Address?

IIT Bombay

- 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

IIT Bombay

- 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”

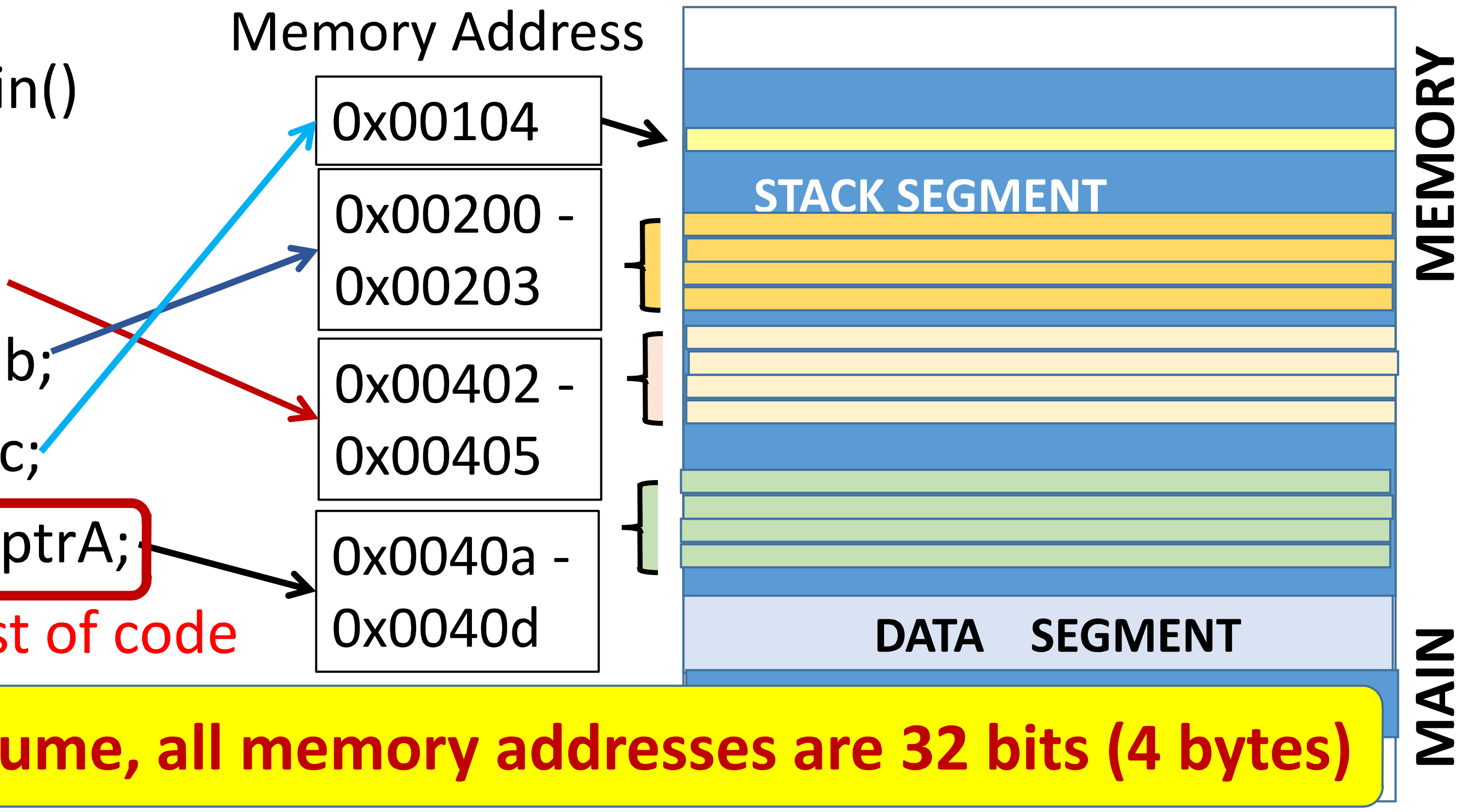
  “T *” is the type “pointer to T”

  “&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 *”?**

Pointer Variables in a C++ Program

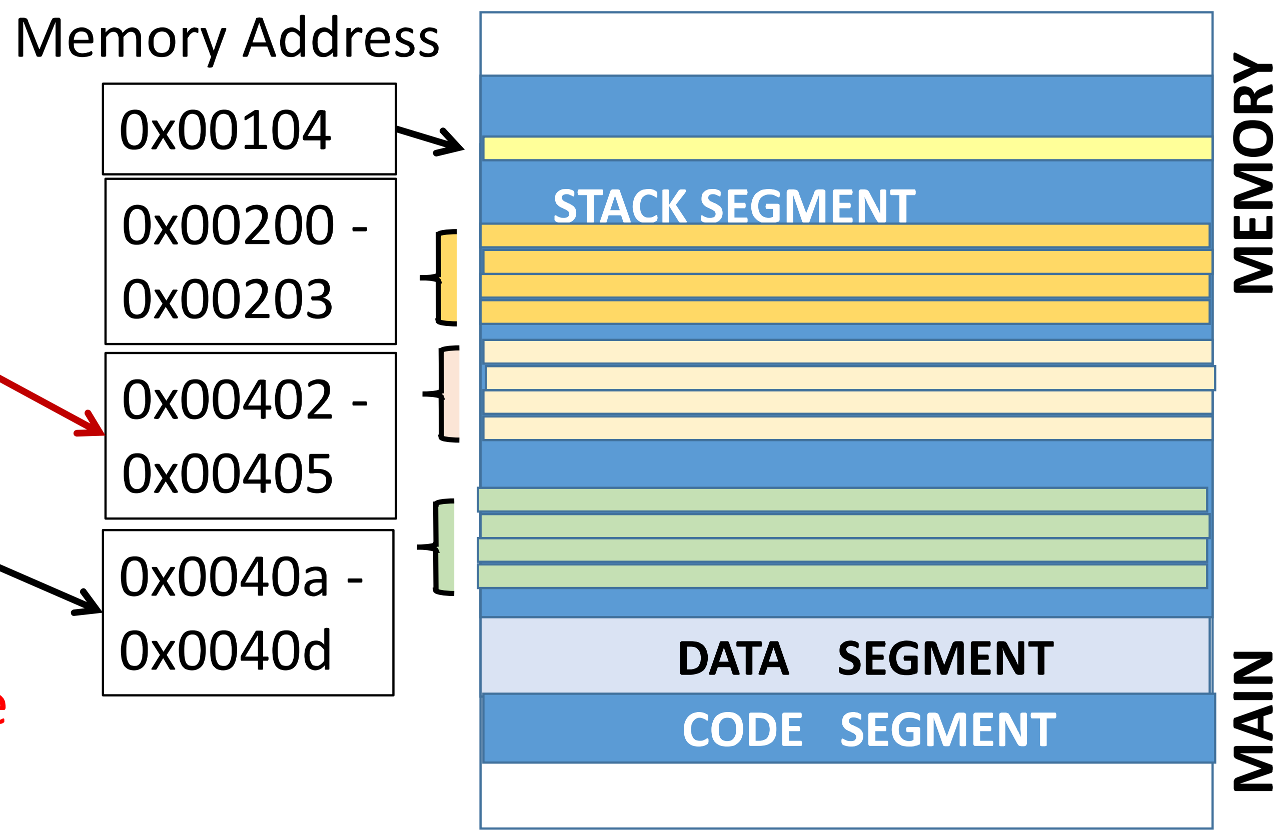
```
int main()
{
   int a;
   float b;
   char c;
   int * ptrA;
 // Rest of code
}
```



Assume, all memory addresses are 32 bits (4 bytes)

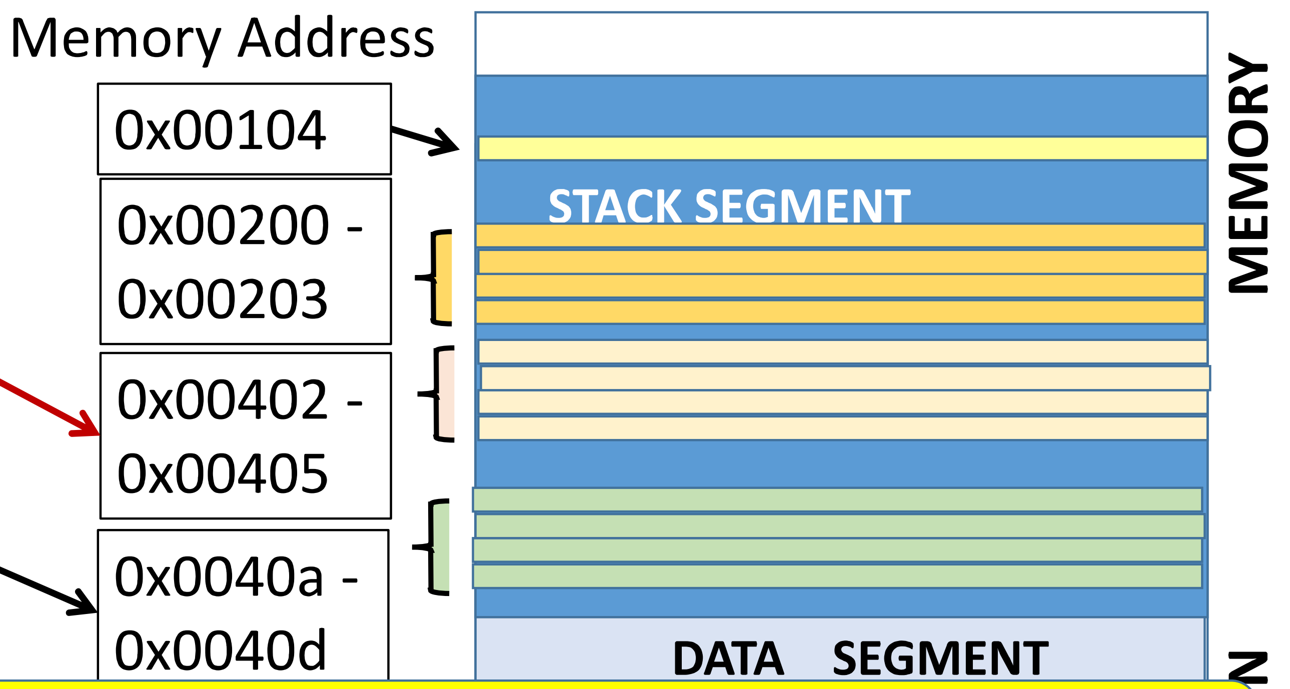
Pointer Variables in a C++ Program

```
int main()
{
   int a;
   float b;
   char c;
   int * ptrA;
   ptrA = &a;
   // Rest of code
}
```



Pointer Variables in a C++ Program

```
int main()
{
   int a;
   float b;
   char c;
   int * ptrA;
  ptrA = &a;
```

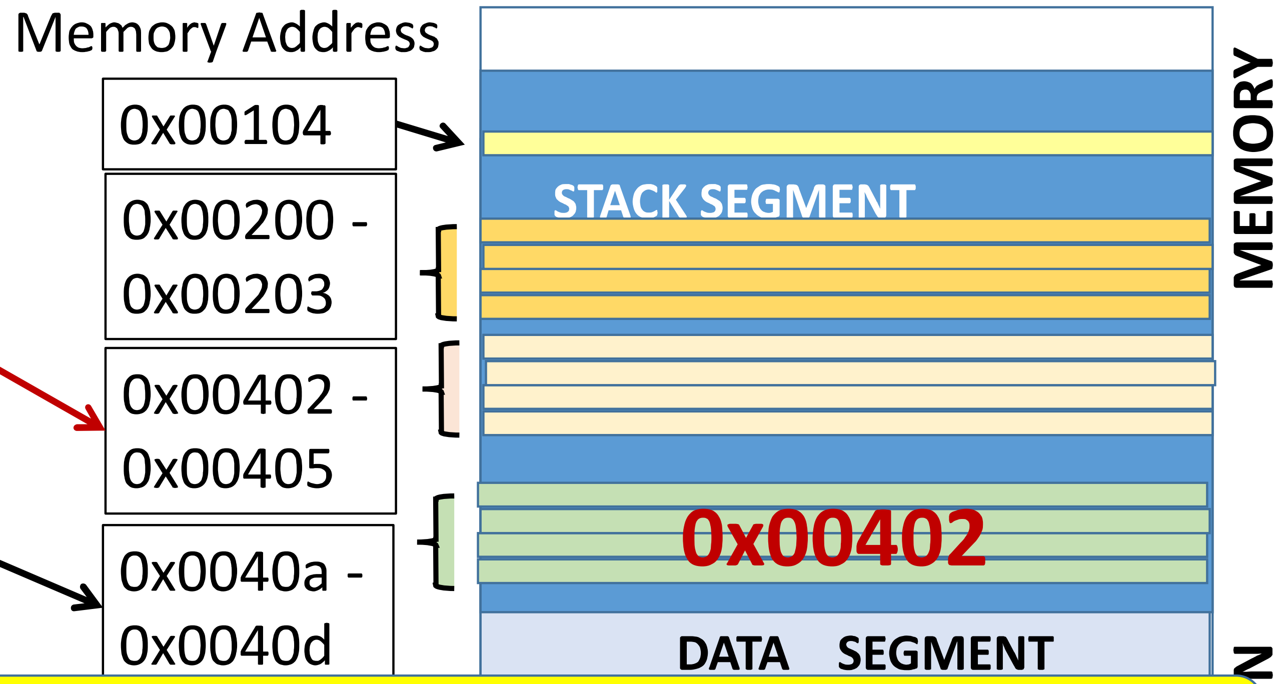


What is the value assigned to ptrA ?

0x00402, 0x00403, 0x00404 or 0x00405 ?

Pointer Variables in a C++ Program

```
int main()
{
   int a;
   float b;
   char c;
   int * ptrA;
ptrA = &a;
```

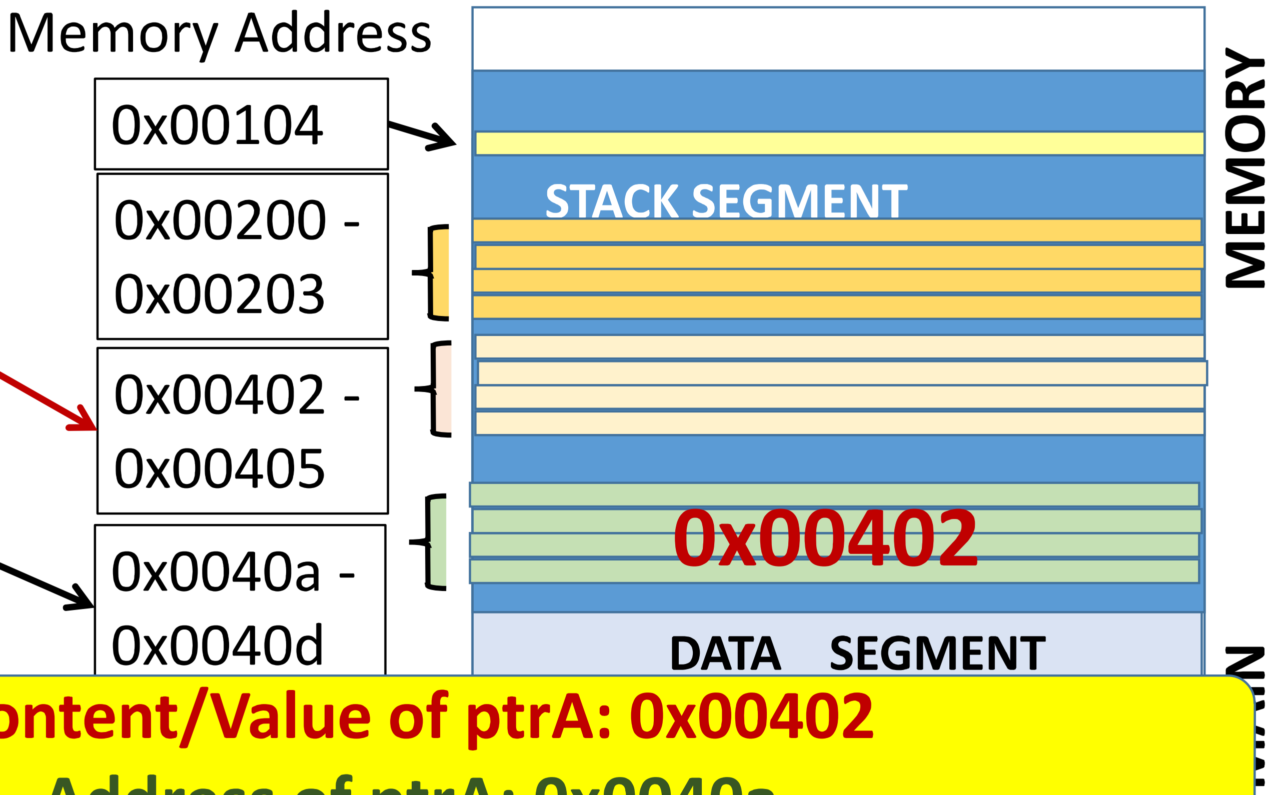


What is the value assigned to ptrA ?

0x00402: Address of first among bytes allocated for a

Value and Address of Pointer Variables

```
int main()
{
   int a;
   float b;
   char c;
   int * ptrA;
ptrA = &a;
```



Content/Value of ptrA: 0x00402

Address of ptrA: 0x0040a

Can We Have Pointers to Pointers?

IIT Bombay

- "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 ****" is a legitimate pointer type in C++

  pointer to pointer to pointer to pointer to int

A Note About Pointer Declarations

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

Types of x, y and z: int

A Note About Pointer Declarations

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

Type of a: pointer to int

Type of c: pointer to int, not pointer to pointer to int

Type of b: int, not pointer to int

A C++ Program For Printing Addresses

IIT Bombay

```
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

IIT Bombay

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