

Computer Programming

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

Session: Introduction to Pointers – Part 2

Quick Recap of Relevant Topics



- Basic programming constructs
- Pointer data type in C++
- “Address of” operator in C++
- Caveats when using “address of” operator

Overview of This Lecture



- Dereferencing a memory address
 - Finding content at a given memory address
- “Contents of” operator in C++

Memory and Addresses

- 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

Address (in hexadecimal)

400	1 0 0 1 1 1 0 1
401	1 0 1 1 1 1 1 1
402	1 0 0 1 0 0 0 1
403	1 0 1 1 0 1 1 1
404	1 0 0 1 0 0 0 1
405	1 0 0 0 0 1 1 1
406	1 1 1 1 0 0 0 1
407	1 0 0 0 0 0 0 0
408	1 1 1 1 1 1 1 1
409	0 0 0 0 0 0 0 0
40a	1 1 1 1 0 0 0 0

Memory For Executing A Program (Process)

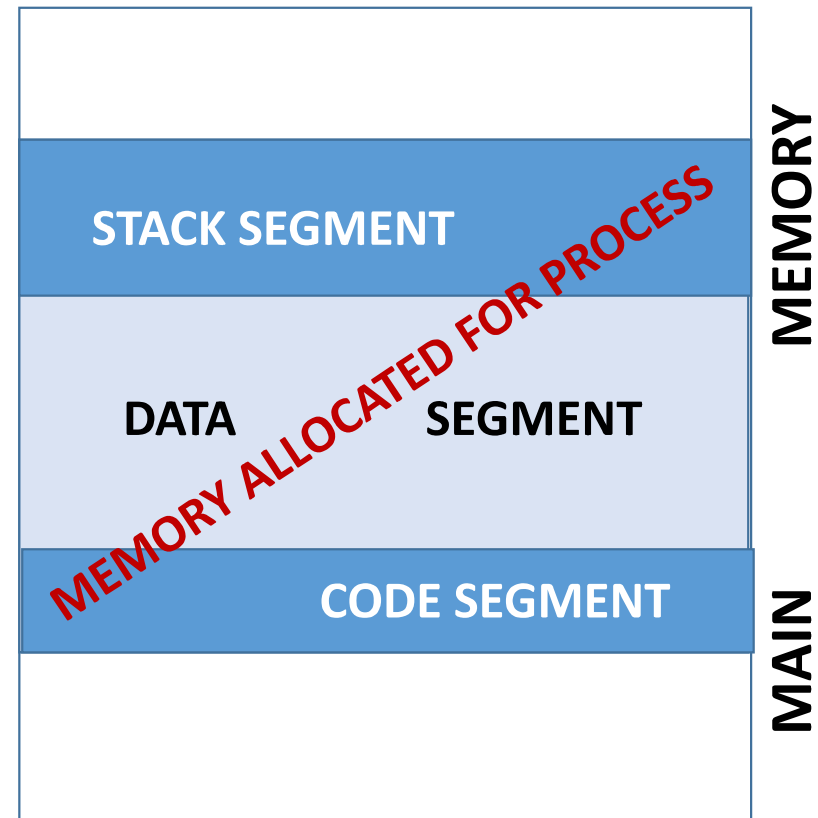
- Operating system allocates a part of main memory for use by a process

- Divided into:

Code segment: Stores executable instructions in program

Data segment: For dynamically allocated data (later lecture)

Stack segment: Call stack



Accessing Content At Given Address

int main() Memory Address

{

int a;

int * ptrA;

a = 0x02abc;

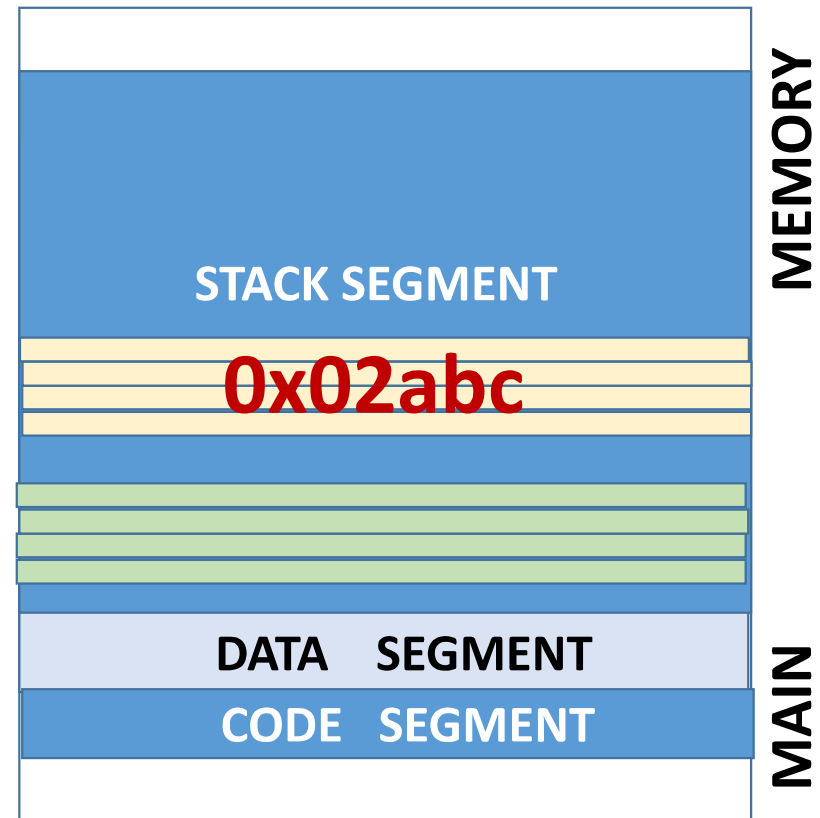
ptrA = &a;

// Rest of code

}

0x00402 -
0x00405

0x0040a -
0x0040d



Accessing Content At Given Address

int main() Memory Address

{

int a;

int * ptrA;

a = 0x02abc;

ptrA = &a;

// Rest of code

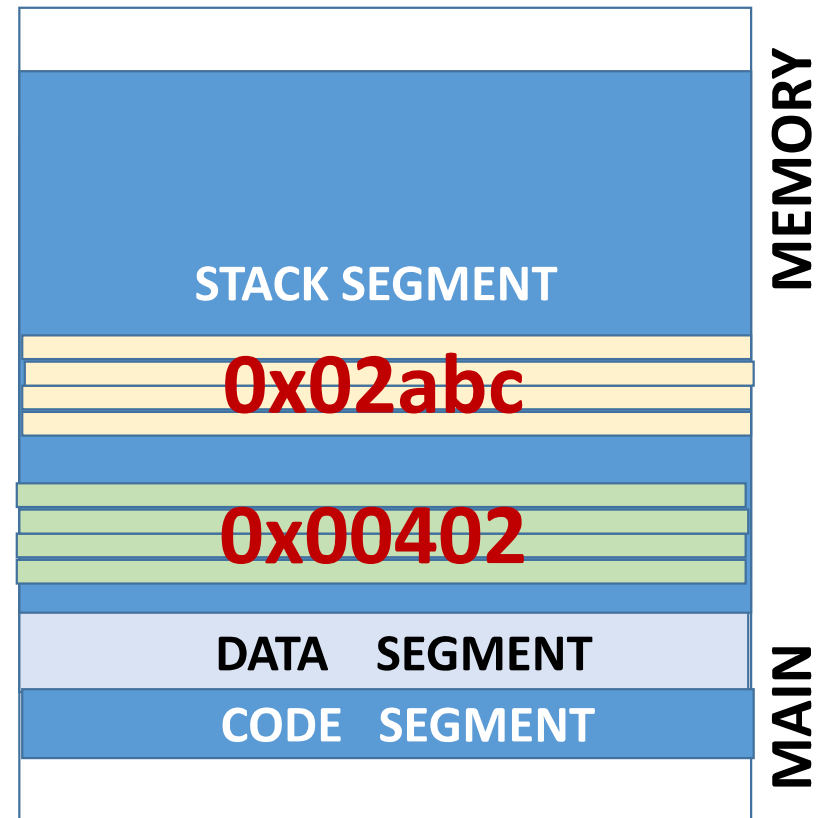
}

0x00402 -
0x00405

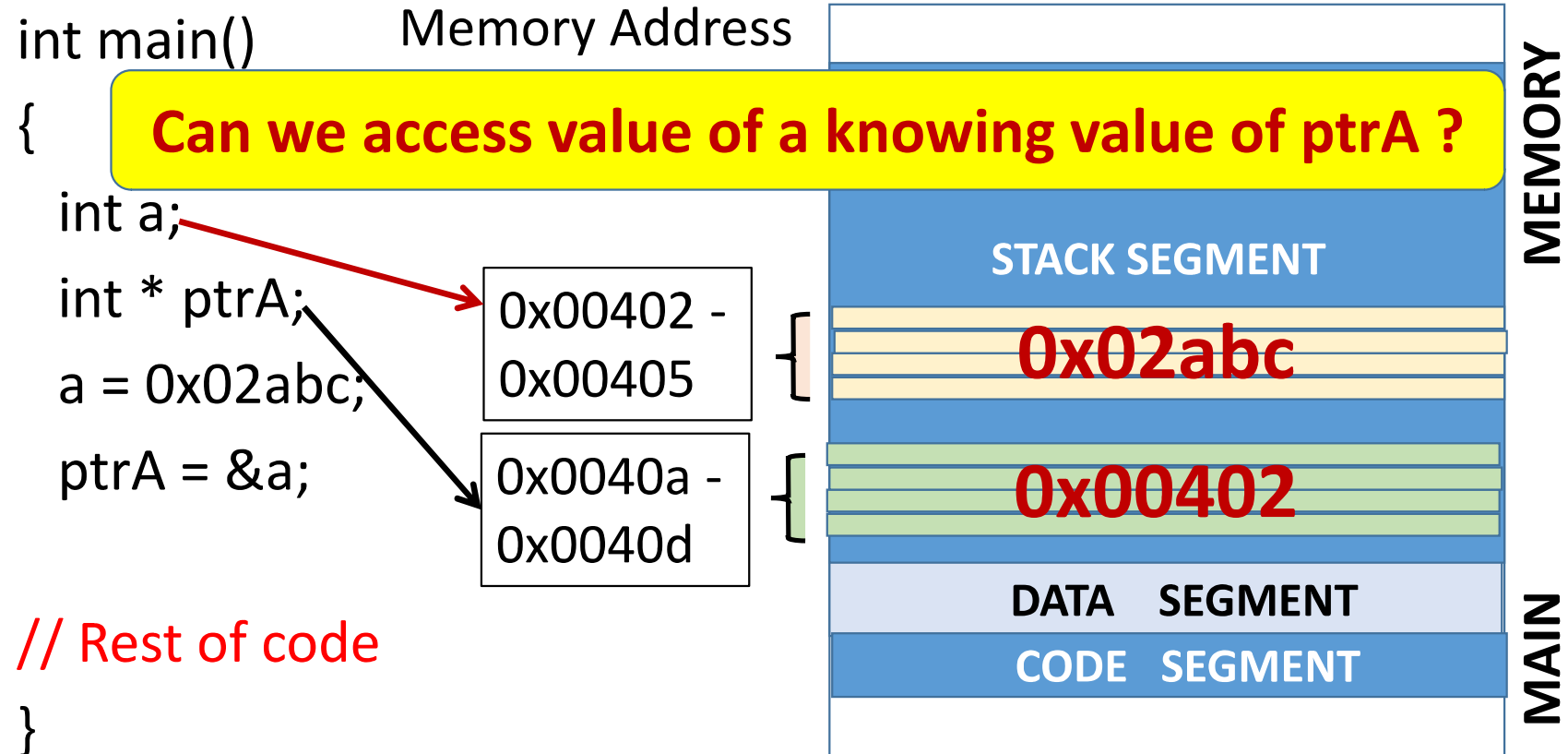
0x0040a -
0x0040d

{

{



Accessing Content At Given Address



Accessing Content At Given Address

- C++ provides a “content of” operator: unary `*`
 - If “ptrA” is a program variable of type “int *”,
“`* ptrA`” gives the integer stored at address given by
“ptrA”

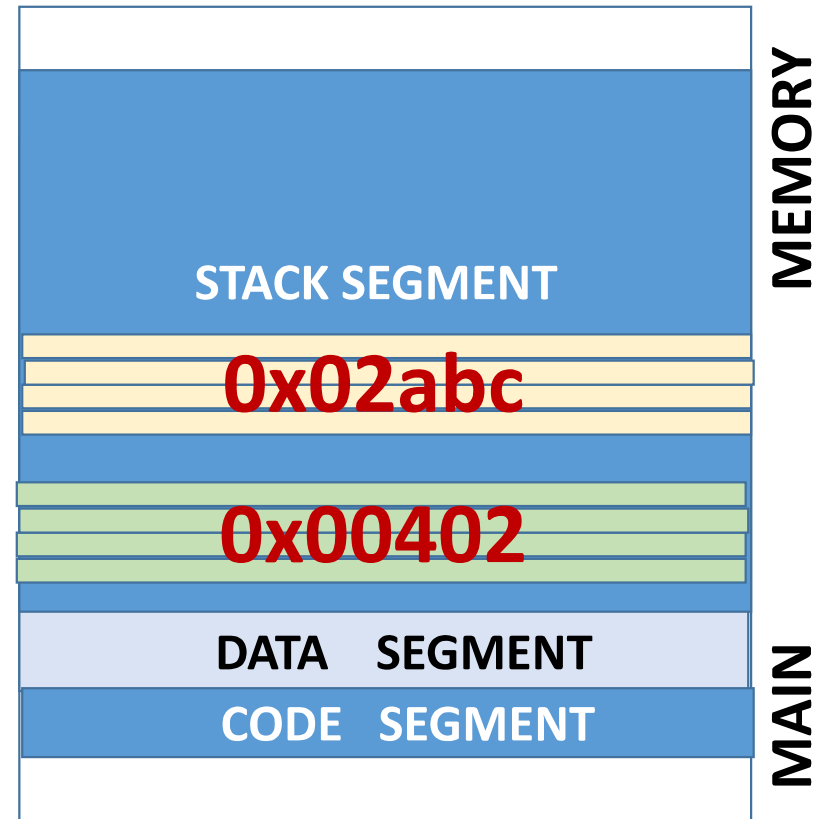
Accessing Content At Given Address

```

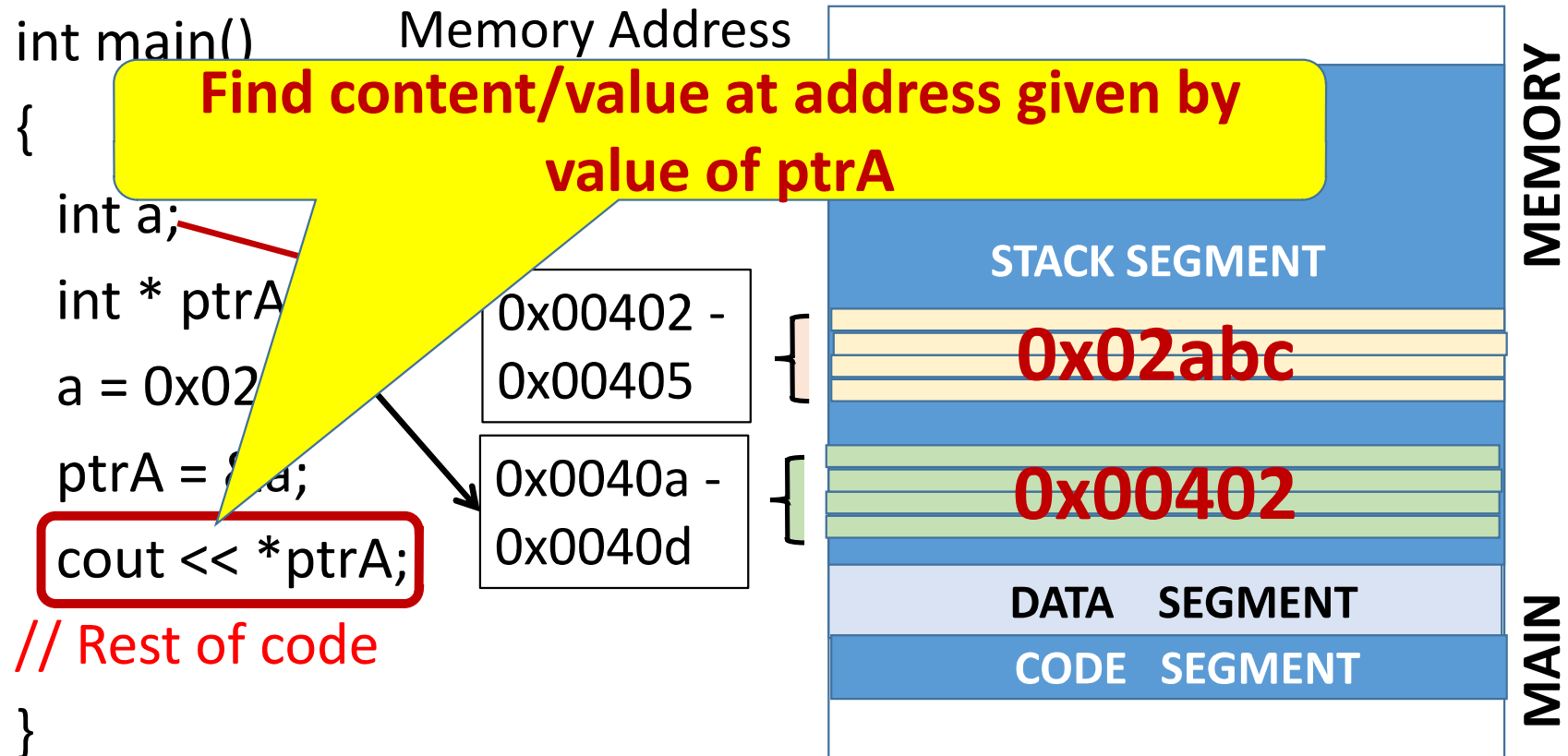
int main()      Memory Address
{
    int a;
    int * ptrA;
    a = 0x02abc;
    ptrA = &a;
    cout << *ptrA;
    // Rest of code
}
  
```

0x00402 -
0x00405

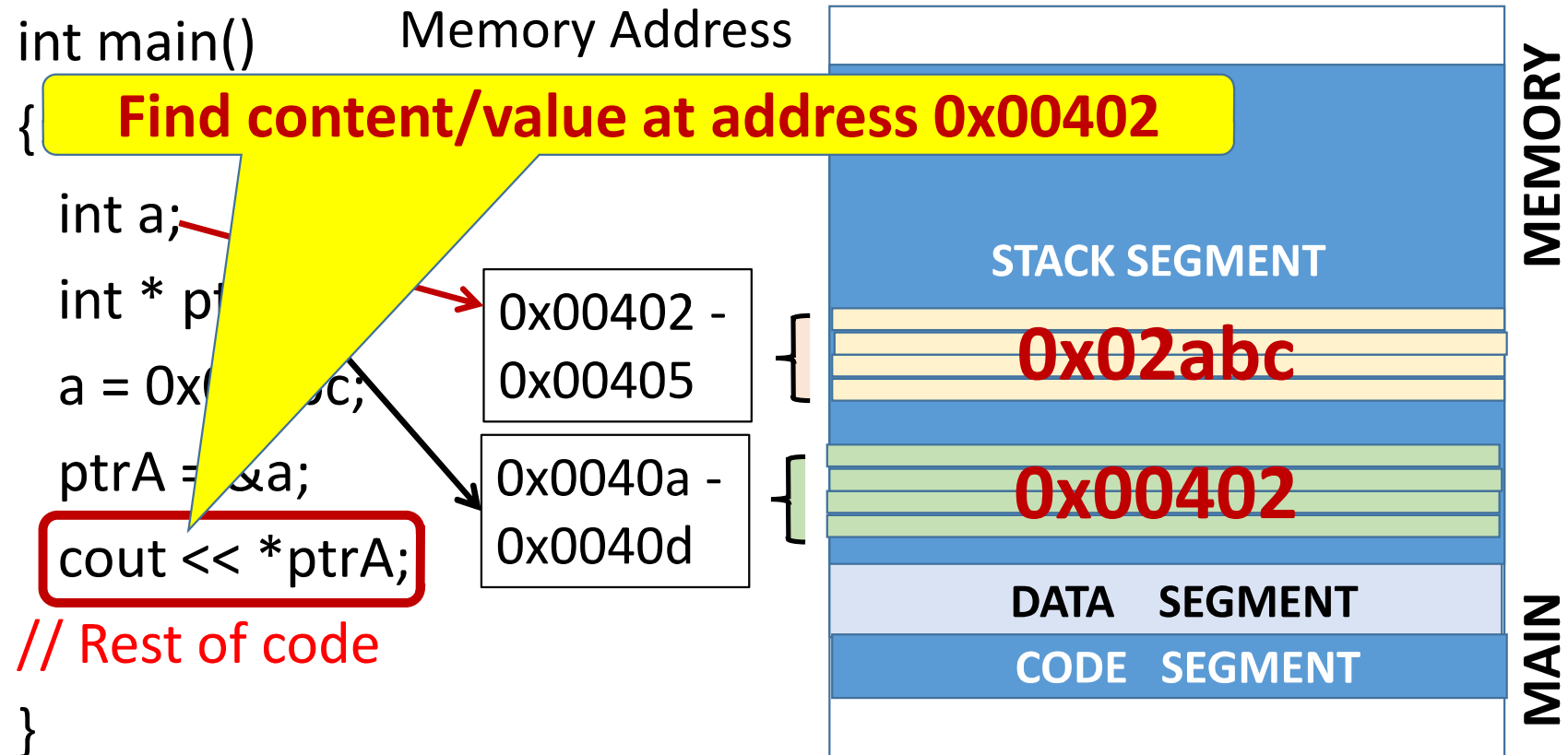
0x0040a -
0x0040d



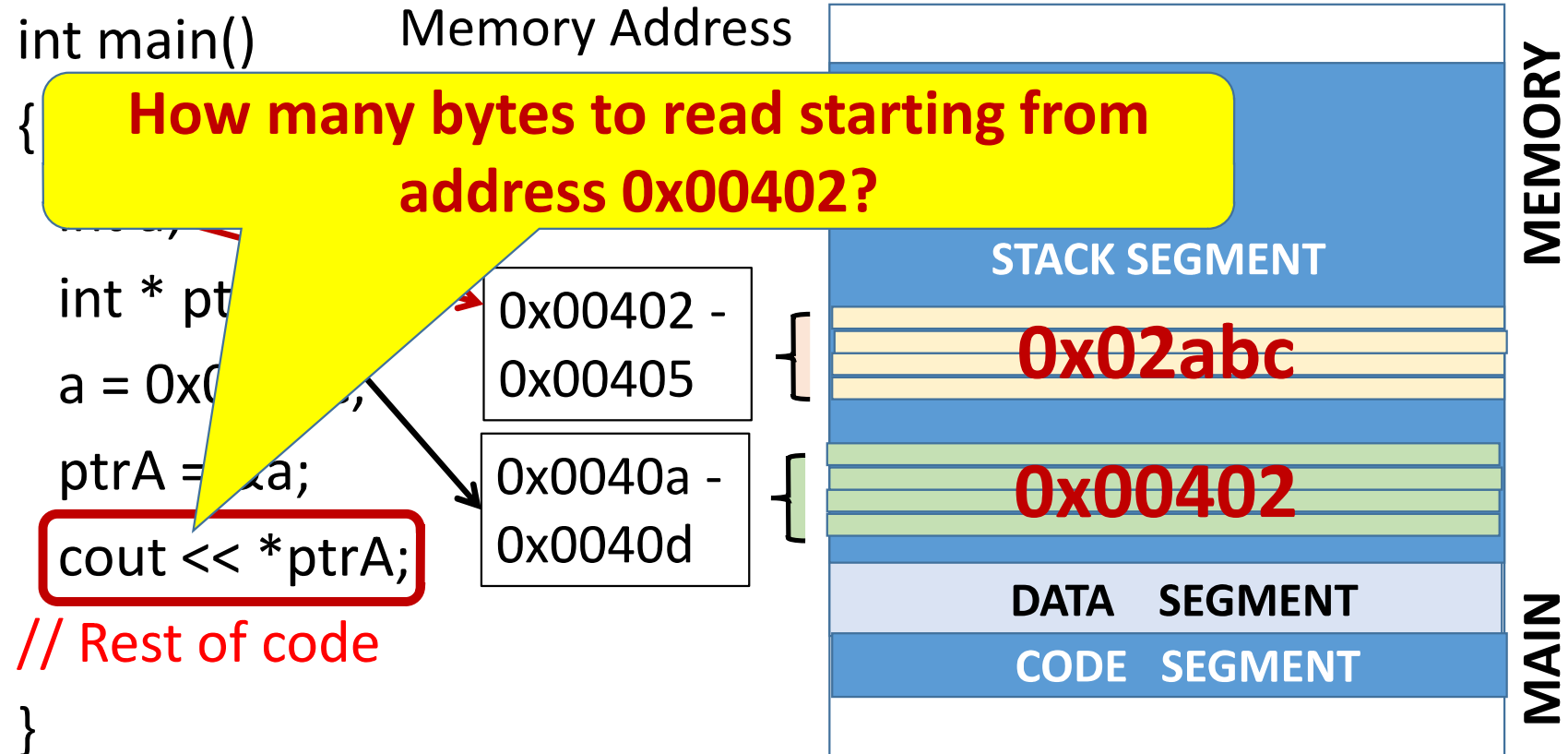
Accessing Content At Given Address



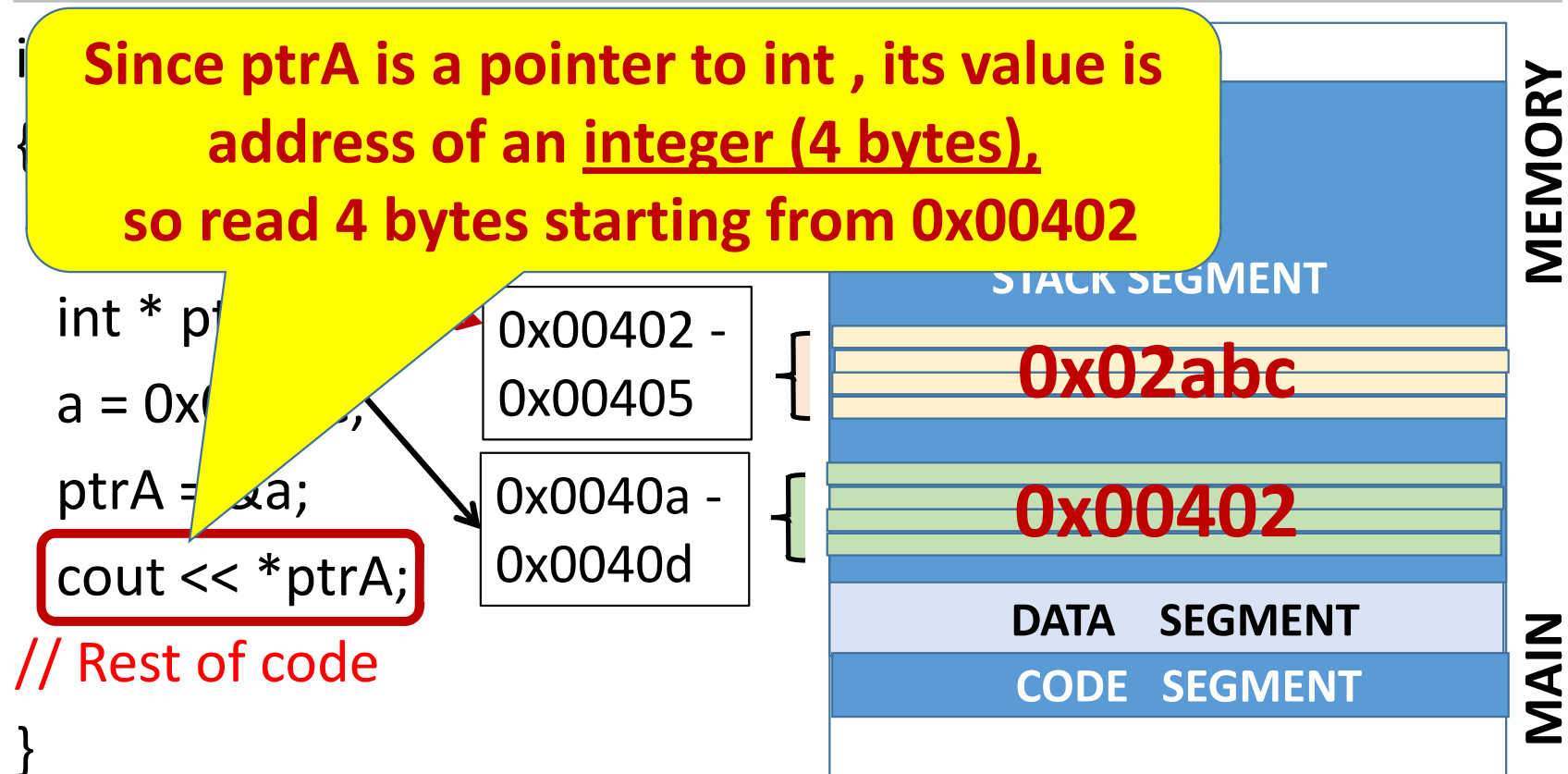
Accessing Content At Given Address



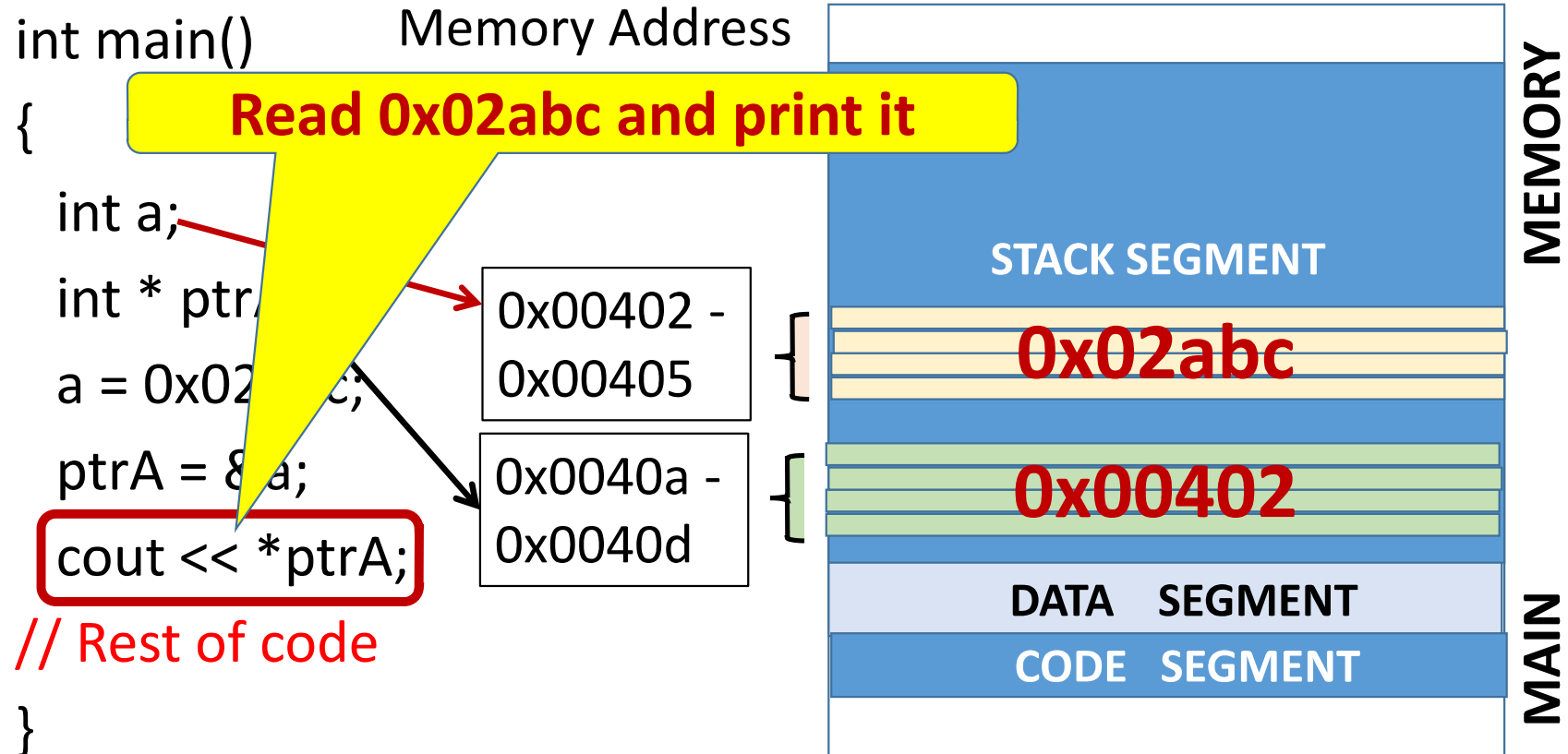
Accessing Content At Given Address



Accessing Content At Given Address



Accessing Content At Given Address



Accessing Content At Given Address

- C++ provides a “content of” operator: unary `*`
 - `*ptrA` gives the content at address given by `ptrA`
 - Unary operator: Takes a single argument
 - `* ptrA` is a C++ expression
- Worry about operator precedence, associativity???
- Simplify life: use parentheses

“Dereferencing an address”: Accessing content at that address

Can have spaces after “&” and “*”: use carefully
Which is more understandable? `&a` or `& a`, `*ptrA` or `* ptrA`

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00302

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00302

0x00402

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

What is the value of *ptrPtrC ?

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

ptrPtrC has value 0x00402

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

***ptrPtrC has value
“content of memory at address 0x00402”**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

**Content at address 0x00402:
How many bytes to read?**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

How many bytes to read?
ptrPtrC is pointer to what data type?
pointer (4 bytes)

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

**To get `*ptrPtrC`, read 4 bytes
starting from address 0x00402**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

***ptrPtrC has value 0x00302**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

What is * (*ptrPtrC) ?

“content of memory at address 0x00302”

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

**Content at address 0x00302:
How many bytes to read?**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

How many bytes to read?
***ptrPtrC is pointer to what data type?**
char (1 byte)

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

**To get *(*ptrPtrC), read 1 byte
starting from address 0x00302**

Can We Have Dereferences of Dereferences?

```
int main()
```

```
{ char c;
```

```
  char * ptrC;
```

```
  char ** ptrPtrC;
```

```
  ptrC = &c;
```

```
  ptrPtrC = &ptrC;
```

```
  cin >> c;
```

```
  cout << *(*ptrPtrC);
```

```
  return 0;
```

```
}
```

Memory Address

0x00302

0x00402 -
0x00405

0x0040a -
0x0040d

STACK SEGMENT

0x00030

0x00302

0x00402

***(*ptrPtrC) has value 0x00030**

How Far Can We Nest Dereferences?

- No pre-specified limit
- If **x** is a variable of type **int ******, we can use upto four levels of dereferencing of **x**
 - Declaration: **int **** x;**
 - ***x**: expression of type **int ***** [Think **int *** *x;**]
 - **__(*x)**: expression of type **int **** [Think **int ** **x;**]
 - **__(*(*x))**: expression of type **int *** [Think **int * ***x;**]
 - **__(*(*(*x)))**: expression of type **int** [Think **int ****x;**]
 - **__(*(*(*(*x))))**: **Compilation error, since *(*(*(*x))) is not of pointer type, and cannot be dereferences**

Caveats When Dereferencing



- Certain memory addresses outside the part of memory allocated to process by operating system
- Dereferencing such an address leads to runtime error
 - Segmentation violation
 - Program aborts/crashes
- Memory location with address 0x0 is never within any user process' memory space
 - Dereferencing 0x0 will certainly cause program to crash
- Need to be careful that we are dereferencing addresses of memory locations allocated to process

Summary



- Dereferencing a memory address
 - Finding content at given address
- “Content of” operator in C++
- Caveats when using “content of” operator
 - Dereferencing “bad” addresses
- We can now access memory locations through their addresses