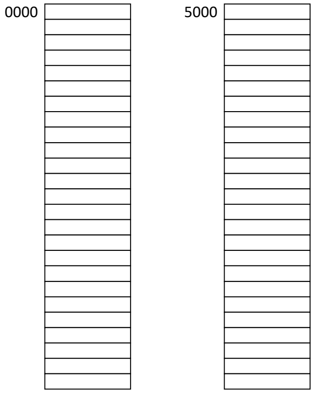
Functions, Outputs, Memory and Pointers

Supplementary for Lecture 4

Yakup Genc

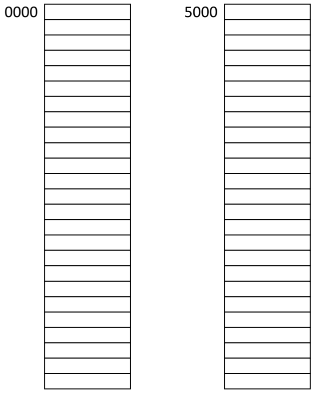
Abstract Memory Managed by Runtime (Compiler)

```c
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```



Abstract Memory Managed by Runtime (Compiler)

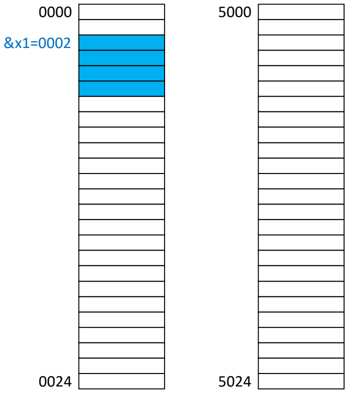
```c
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```



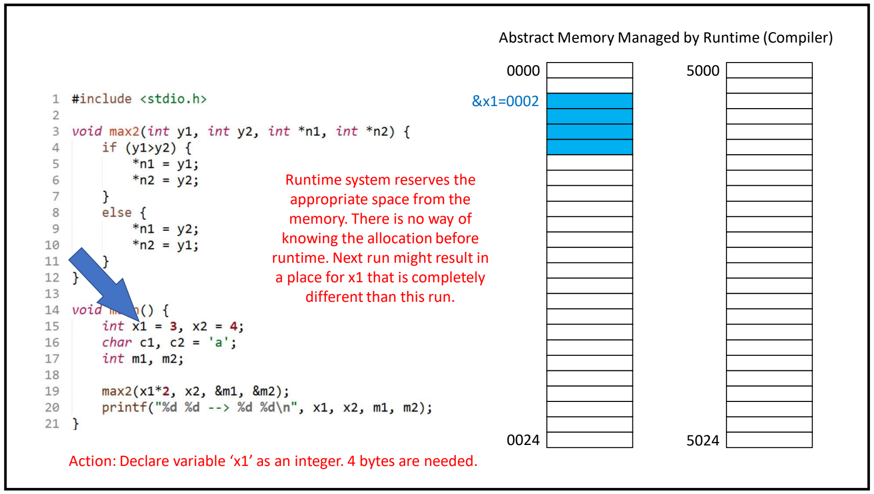
Action: Start the program at the main function.

Abstract Memory Managed by Runtime (Compiler)

```c
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```



Action: Declare variable ‘x1’ as an integer. 4 bytes are needed.



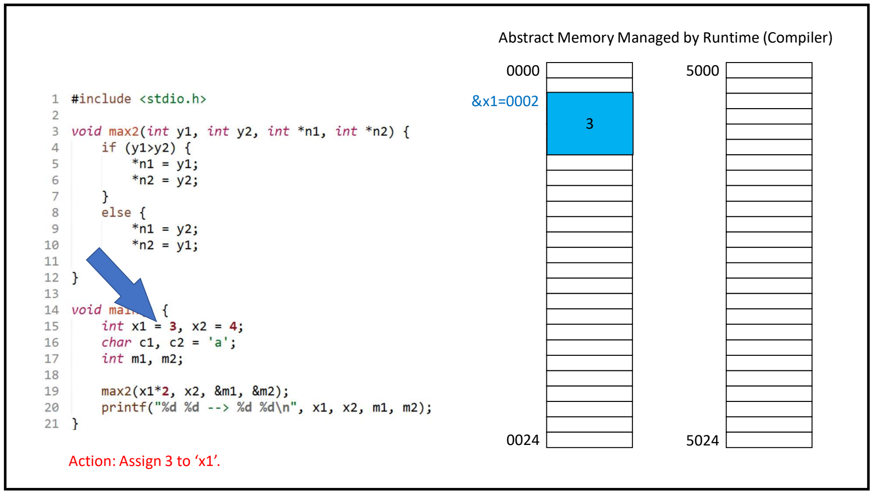
Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Runtime system reserves the appropriate space from the memory. There is no way of knowing the allocation before runtime. Next run might result in a place for x1 that is completely different than this run.

Action: Declare variable ‘x1’ as an integer. 4 bytes are needed.

5

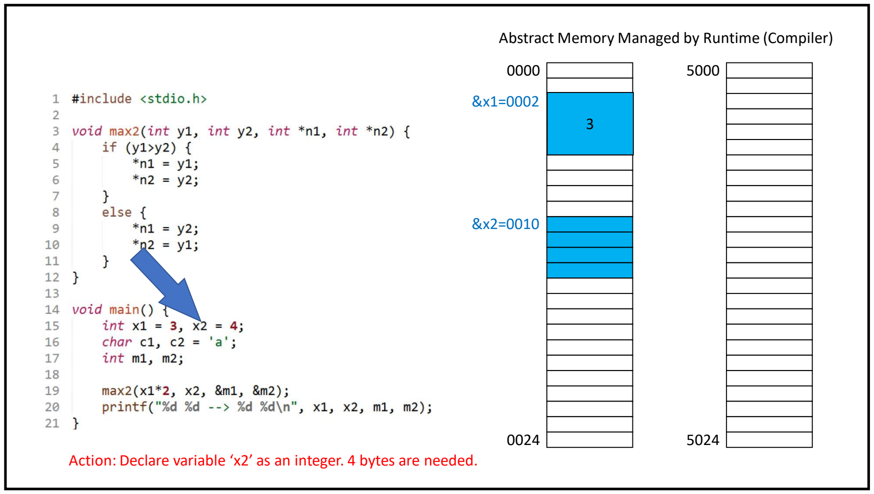


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Assign 3 to ‘x1’.

6

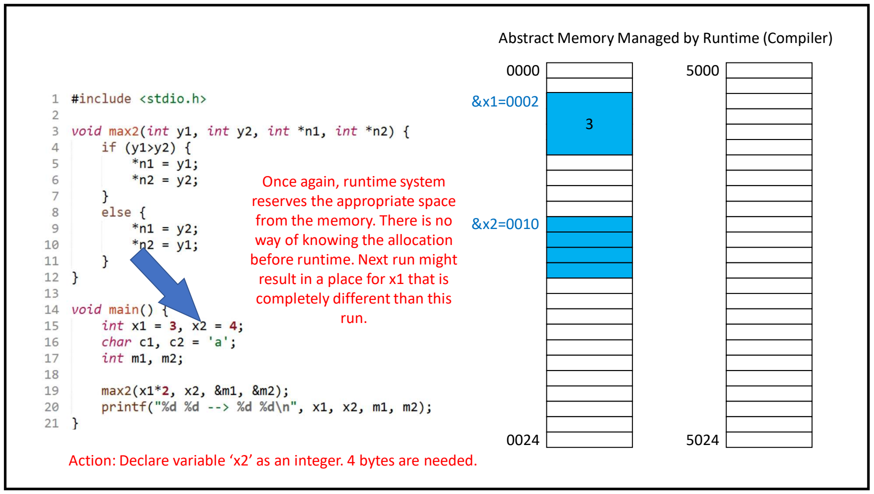


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Declare variable ‘x2’ as an integer. 4 bytes are needed.

7



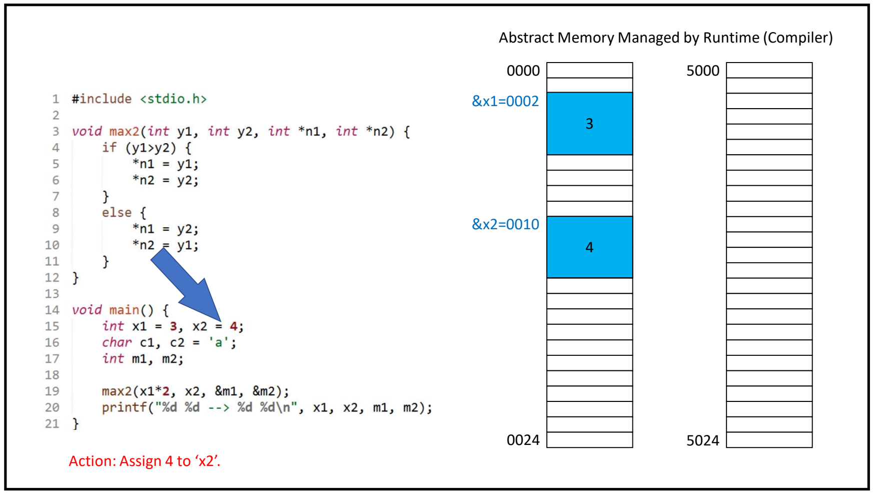
Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Once again, runtime system reserves the appropriate space from the memory. There is no way of knowing the allocation before runtime. Next run might result in a place for x1 that is completely different than this run.

Action: Declare variable ‘x2’ as an integer. 4 bytes are needed.

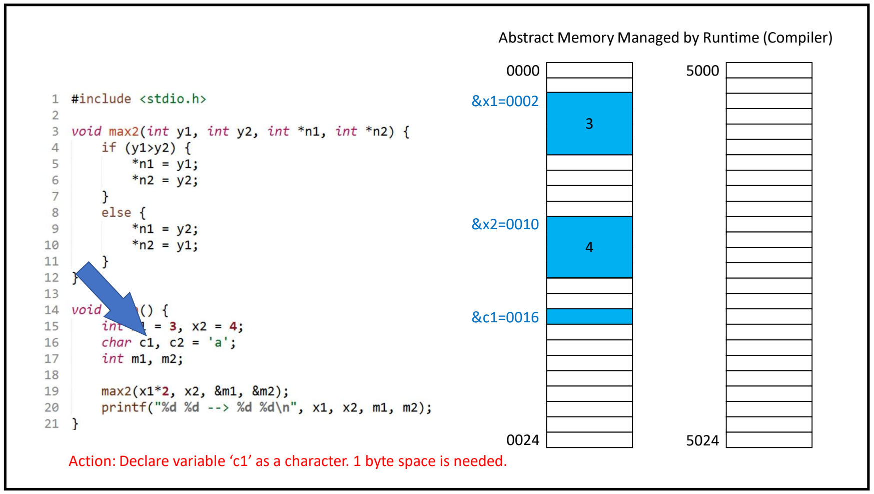
8



Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

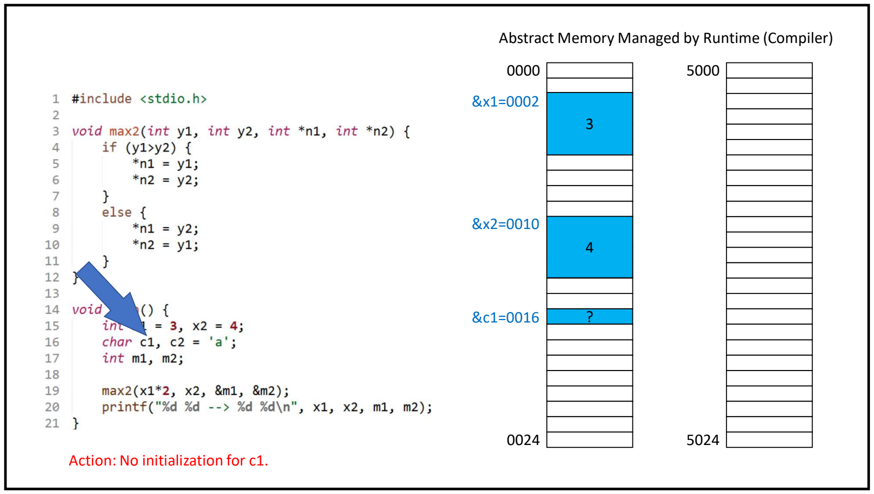
Action: Assign 4 to ‘x2’.



Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

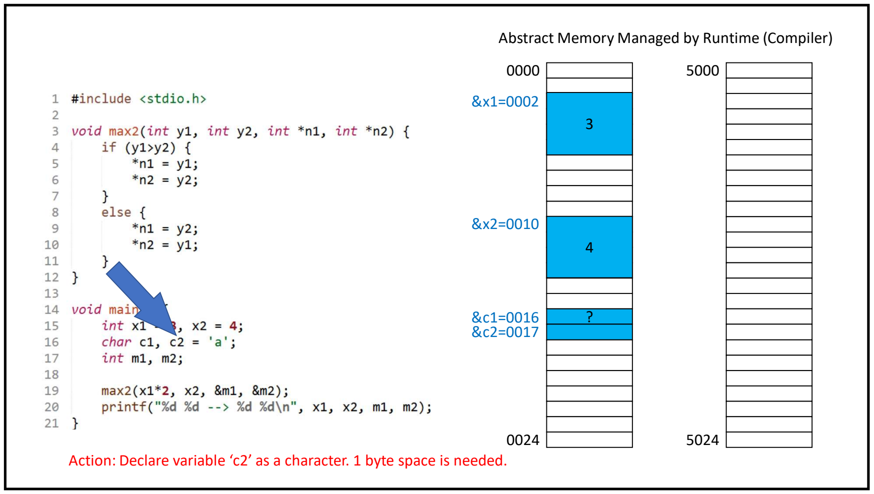
Action: Declare variable ‘c1’ as a character. 1 byte space is needed.



Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

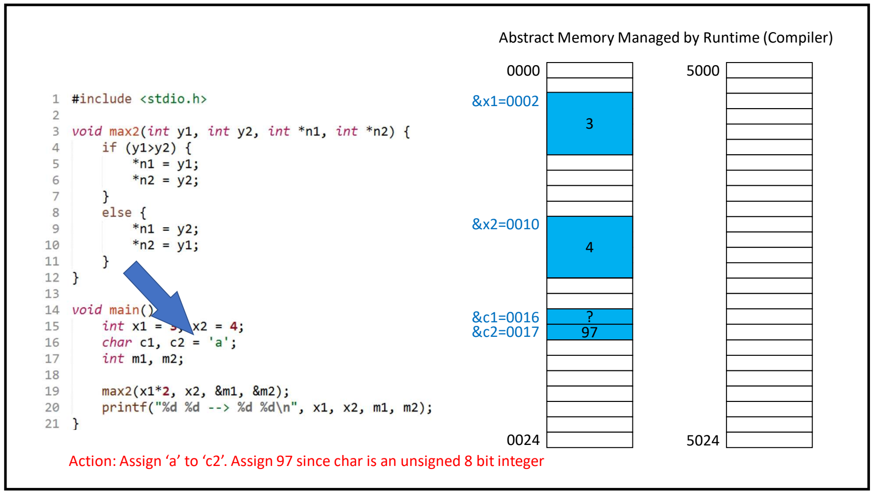
Action: No initialization for c1.



Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

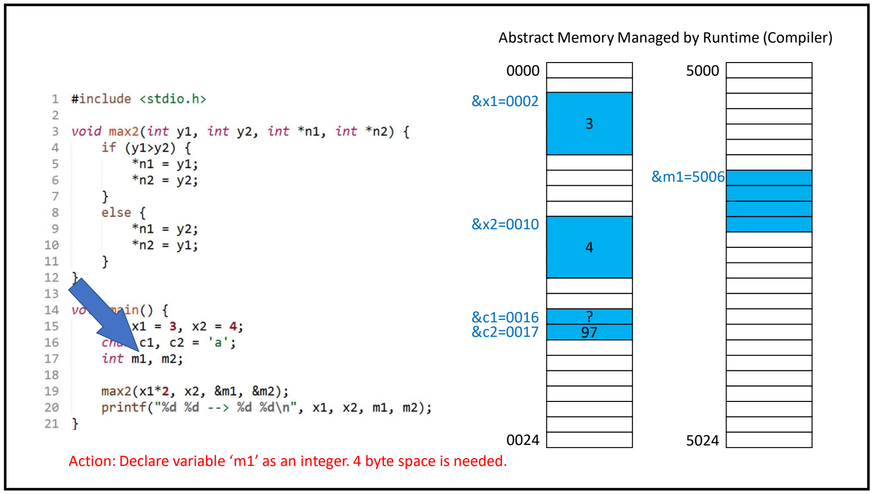
Action: Declare variable ‘c2’ as a character. 1 byte space is needed.



```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Assign ‘a’ to ‘c2’. Assign 97 since char is an unsigned 8 bit integer

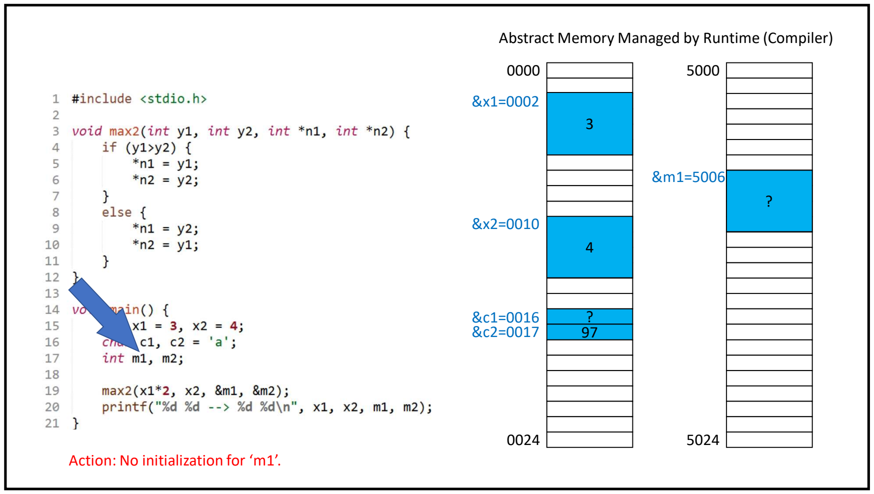
13



```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Declare variable ‘m1’ as an integer. 4 byte space is needed.

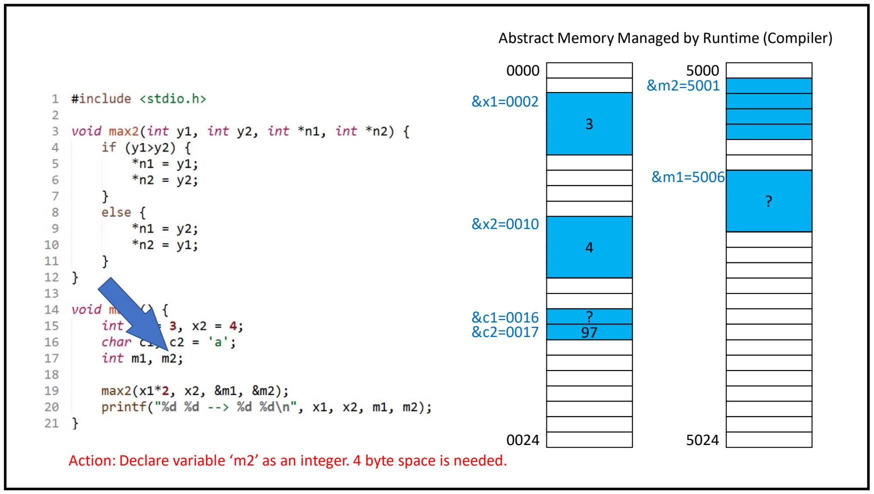
14



```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: No initialization for ‘m1’.

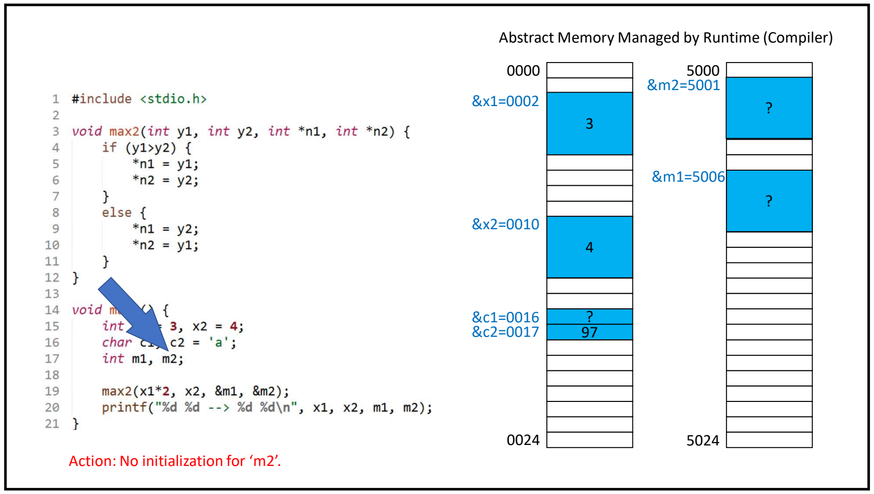
15



```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Declare variable ‘m2’ as an integer. 4 byte space is needed.

16

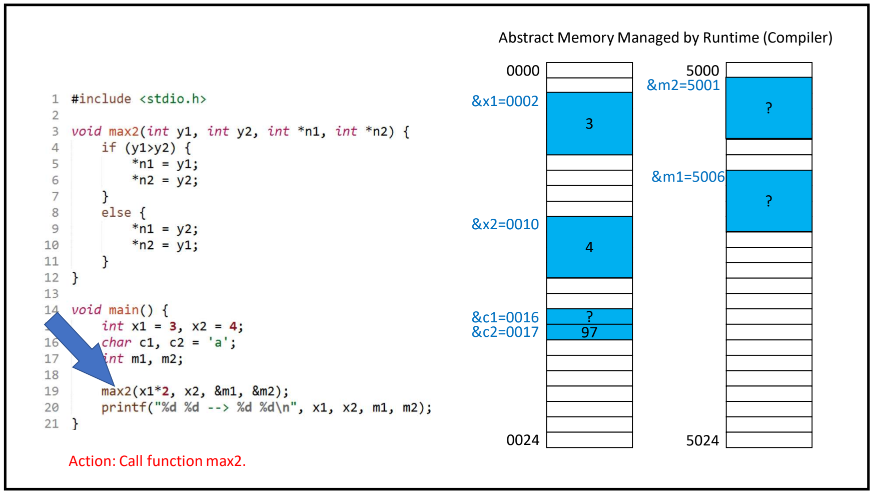


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: No initialization for ‘m2’.

17

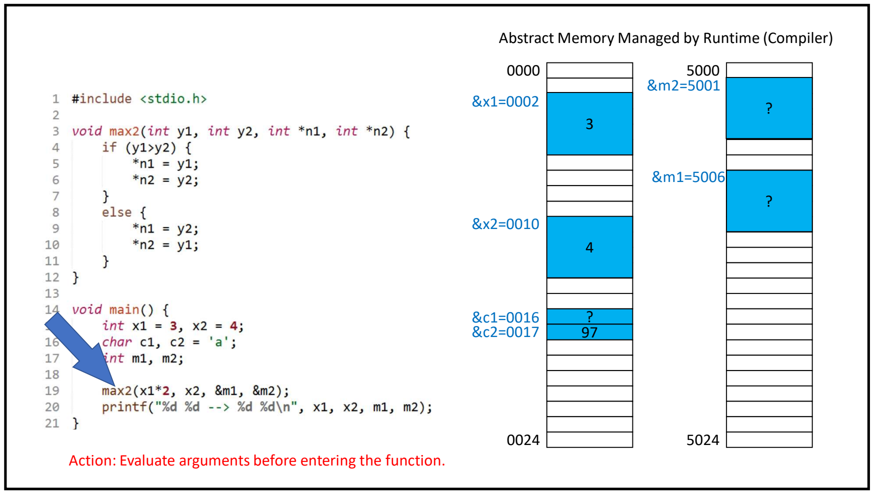


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Call function max2.

18

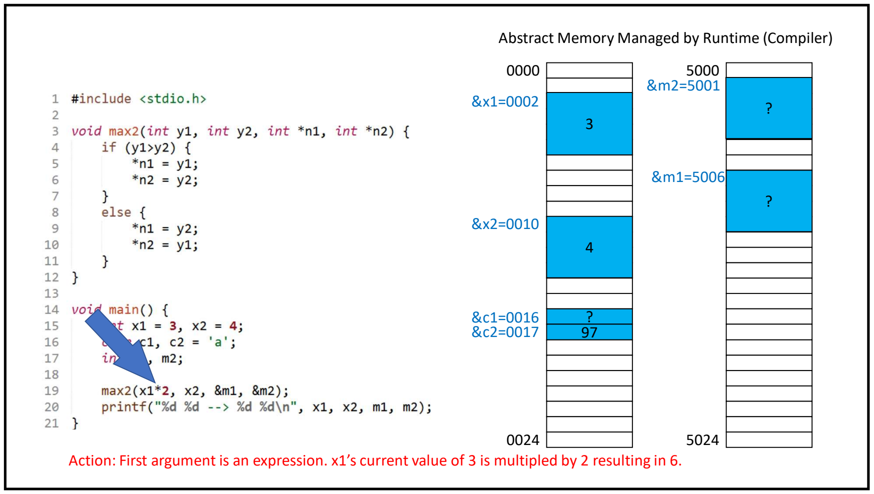


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Evaluate arguments before entering the function.

19

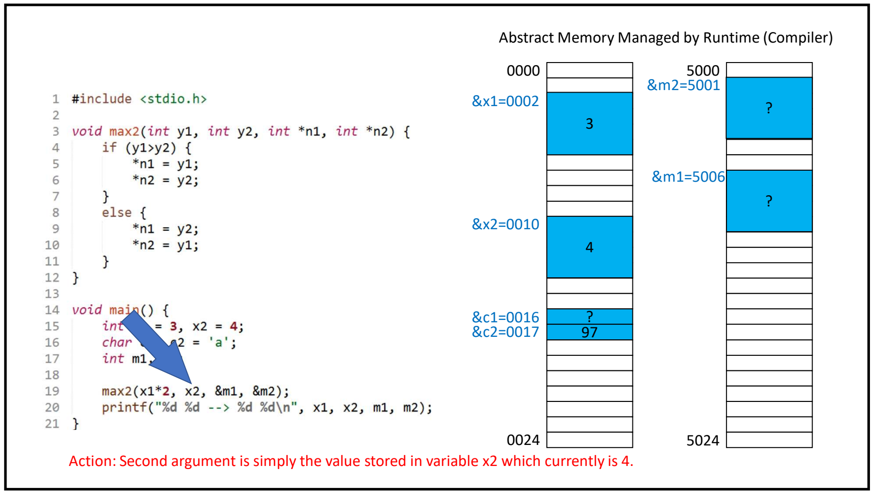


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: First argument is an expression. x1’s current value of 3 is multipled by 2 resulting in 6.

20

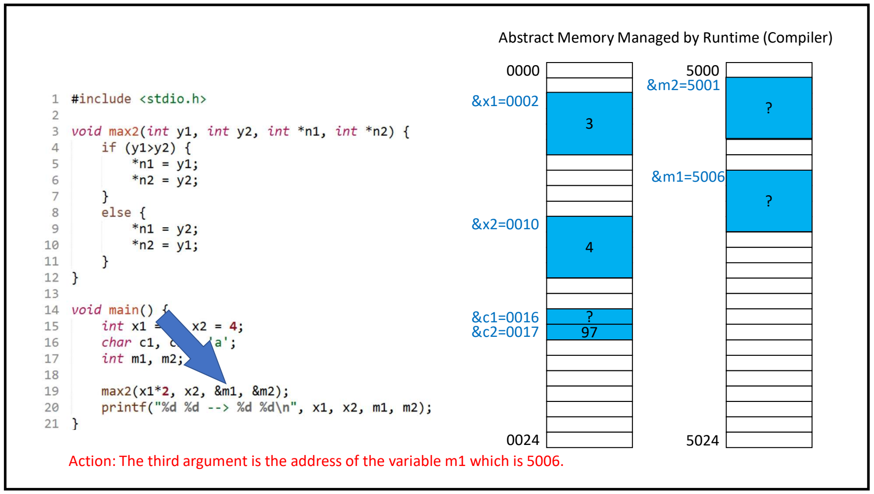


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int     = 3, x2 = 4;
    char c    2 = 'a';
    int m1,

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Second argument is simply the value stored in variable x2 which currently is 4.

21

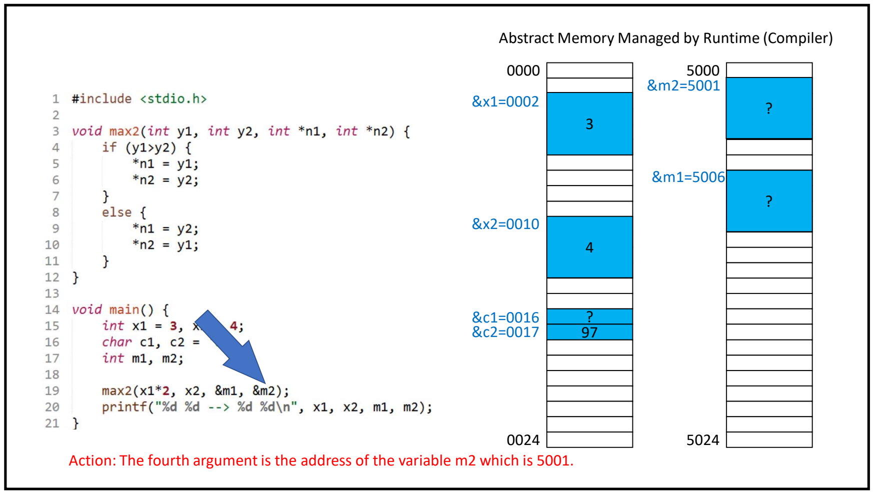


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 =    x2 = 4;
    char c1, c    'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: The third argument is the address of the variable m1 which is 5006.

22

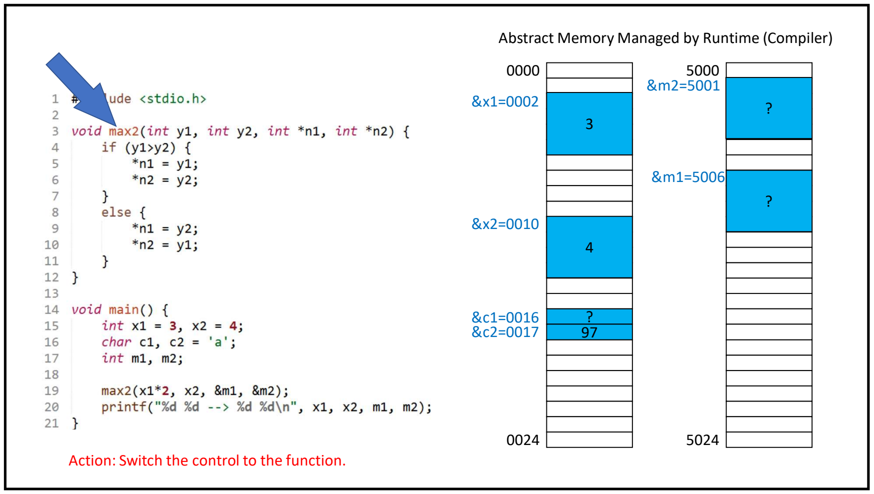


Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x     4;
    char c1, c2 =
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: The fourth argument is the address of the variable m2 which is 5001.

23



Abstract Memory Managed by Runtime (Compiler)

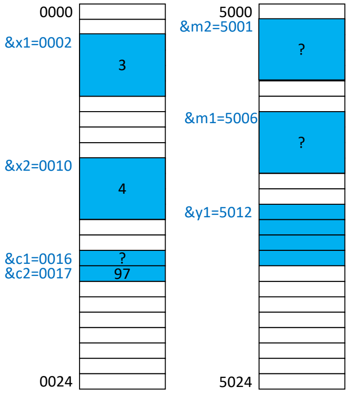
```
#    ude <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Action: Switch the control to the function.

24

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

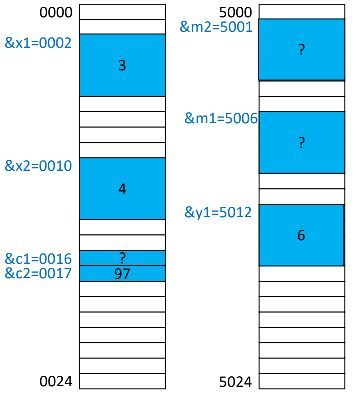
Abstract Memory Managed by Runtime (Compiler)



Action: Declare the local variable y1.

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

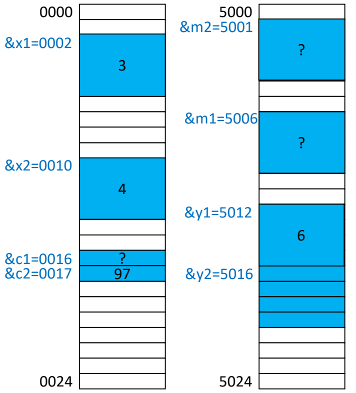
Abstract Memory Managed by Runtime (Compiler)



Action: Assign the value for the first argument to y1.

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

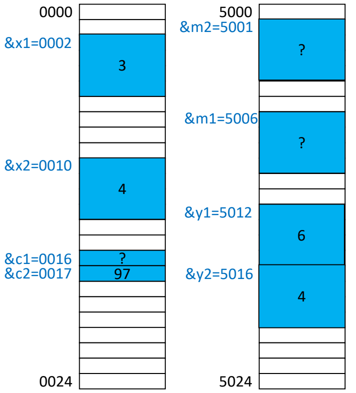
Abstract Memory Managed by Runtime (Compiler)



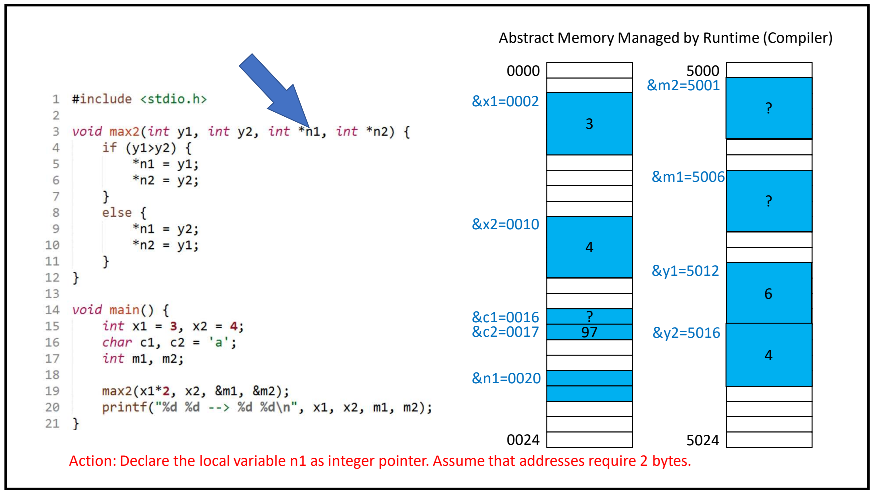
Action: Declare the local variable y2.

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

Abstract Memory Managed by Runtime (Compiler)



Action: Assign the second input value to y2.



Abstract Memory Managed by Runtime (Compiler)

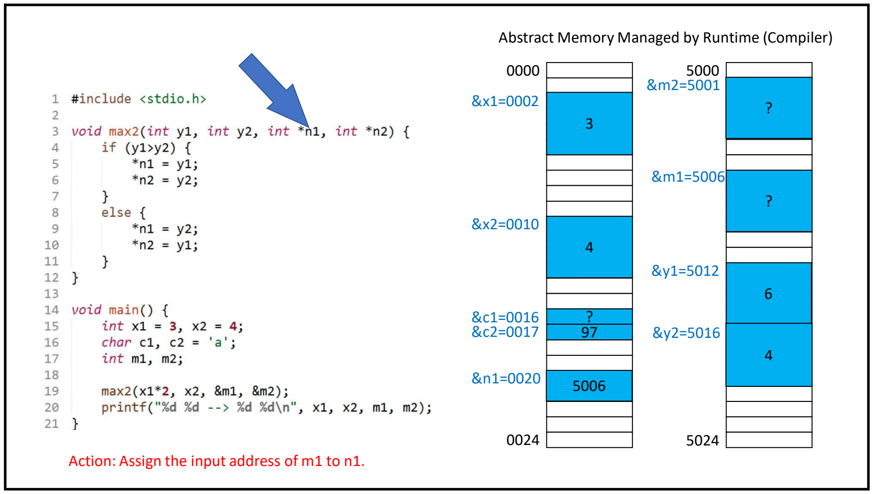
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Declare the local variable n1 as integer pointer. Assume that addresses require 2 bytes.

29



Abstract Memory Managed by Runtime (Compiler)

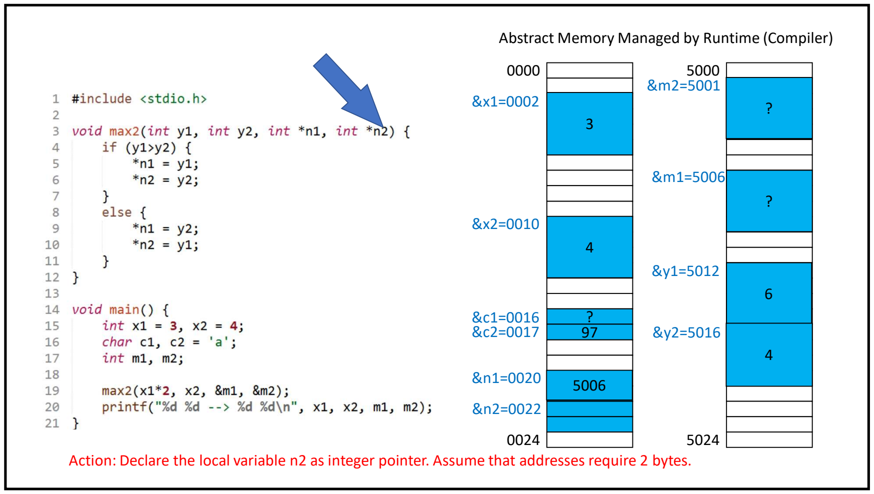
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Assign the input address of m1 to n1.

30



Abstract Memory Managed by Runtime (Compiler)

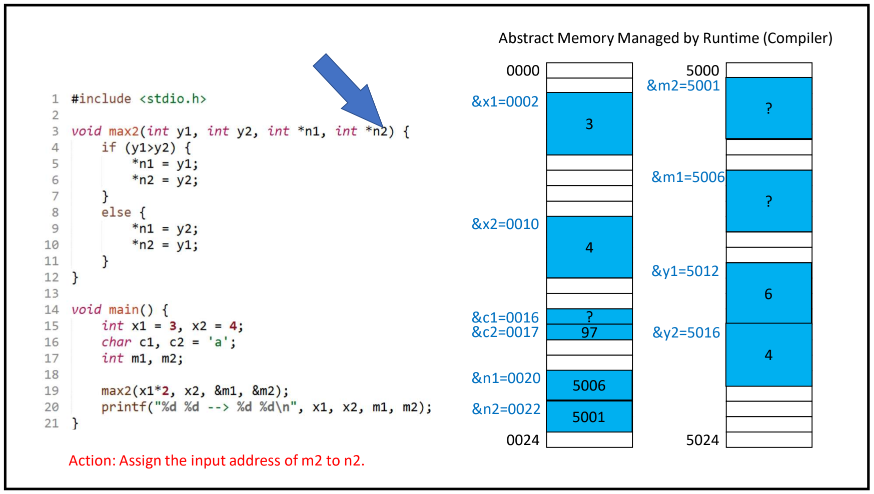
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Declare the local variable n2 as integer pointer. Assume that addresses require 2 bytes.

31



Abstract Memory Managed by Runtime (Compiler)

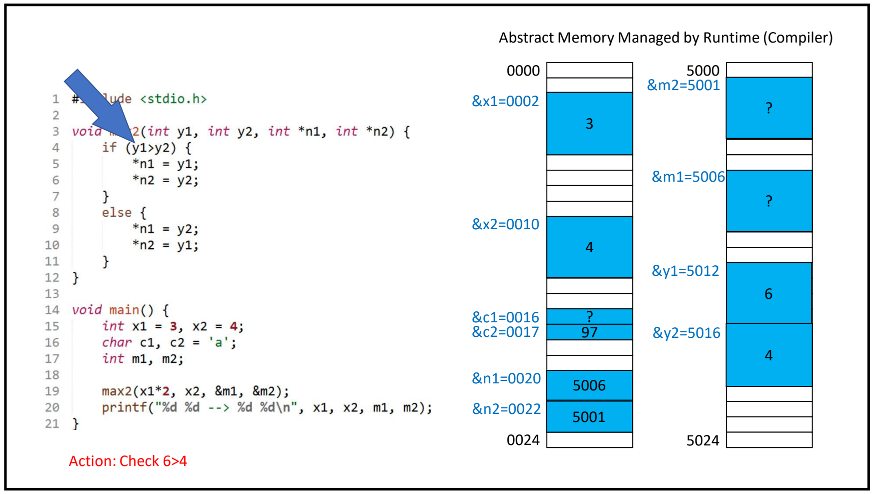
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022 5001
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Assign the input address of m2 to n2.

32



Abstract Memory Managed by Runtime (Compiler)

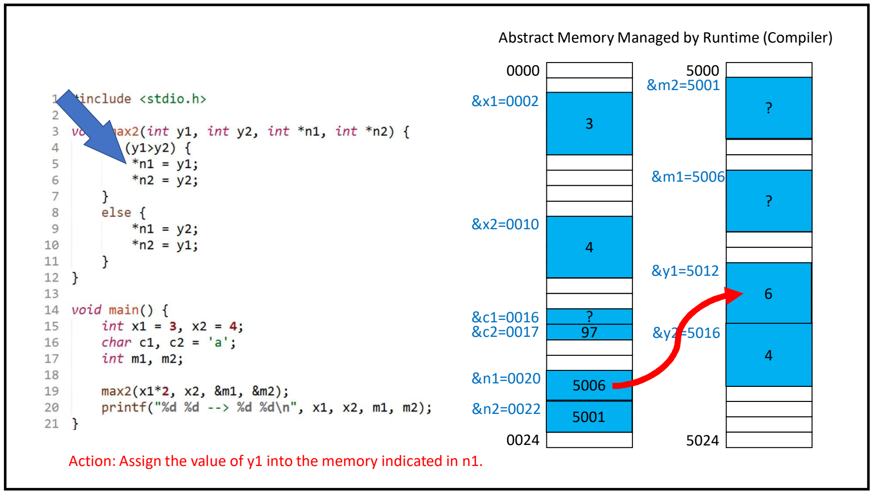
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022 5001
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Check 6>4

33



Abstract Memory Managed by Runtime (Compiler)

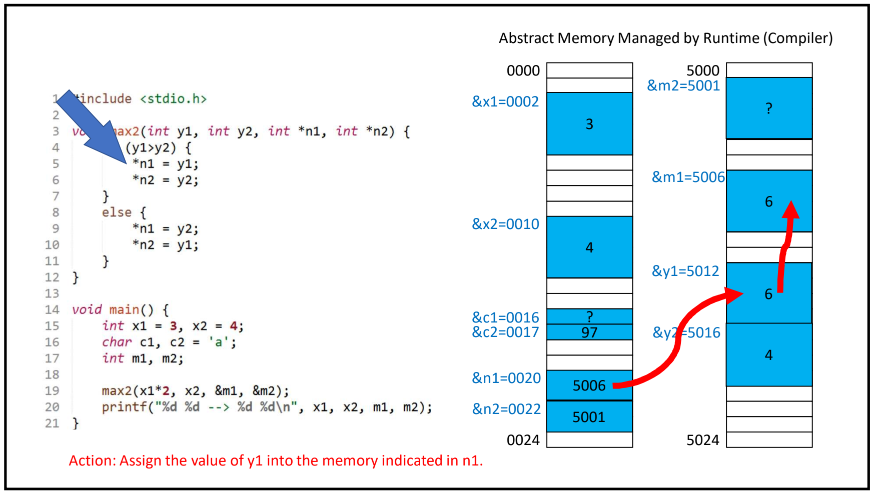
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022 5001
0024

5000
&m2=5001 ?
&m1=5006 ?
&y1=5012 6
&y2=5016 4
5024

Action: Assign the value of y1 into the memory indicated in n1.

34



Abstract Memory Managed by Runtime (Compiler)

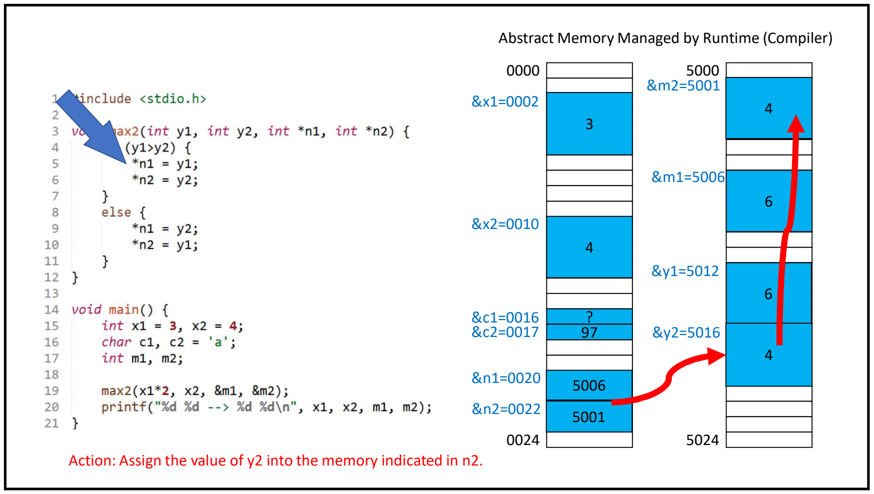
```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022 5001
0024

5000
&m2=5001 ?
&m1=5006 6
&y1=5012 6
&y2=5016 4
5024

Action: Assign the value of y1 into the memory indicated in n1.

35



Abstract Memory Managed by Runtime (Compiler)

```
#include <stdio.h>

void max2(int y1, int y2, int *n1, int *n2) {
    if (y1>y2) {
        *n1 = y1;
        *n2 = y2;
    }
    else {
        *n1 = y2;
        *n2 = y1;
    }
}

void main() {
    int x1 = 3, x2 = 4;
    char c1, c2 = 'a';
    int m1, m2;

    max2(x1*2, x2, &m1, &m2);
    printf("%d %d --> %d %d\n", x1, x2, m1, m2);
}
```

0000
&x1=0002 3
&x2=0010 4
&c1=0016 ?
&c2=0017 97
&n1=0020 5006
&n2=0022 5001
0024

5000
&m2=5001 4
&m1=5006 6
&y1=5012 6
&y2=5016 4
5024

Action: Assign the value of y2 into the memory indicated in n2.

36