INTRODUCTION TO POINTERS

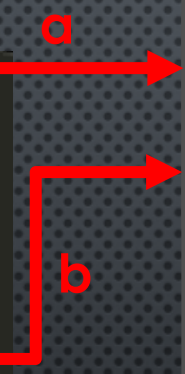
HOW MEMORY WORKS ?

Code :

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int main(){
5      int a = 0;
6      int b = 3;
7
8      printf("%x", &b);
9  }
```

Output :

```
$sh  ./a.out
ff00fa00
```



Process Memory

0xff00fa03	Start of stack frame
0xff00fa02	0
0xff00fa00	3
0xff000000	End of stack frame

What is a pointer ?

- A pointer is a variable : has an adress
- A pointer contains the adress of another variable
- The adress contained comes from the context where the pointer is initialized

What for ?

- When you call a function : new stackframe, cannot access variables from the caller context
- You pass variables as arguments of the function : same problem, because you create a copy of the original variable in the new stackframe
- Solution : pointer, contains adress of the variable in the caller's stackframe

HOW POINTERS WORKS ?

Code :

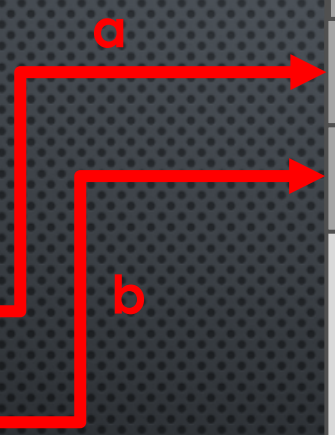
```c
#include <stdlib.h>
#include <stdio.h>

int main(){

    // variable
    int a = 0;

    // pointer to the variable
    int* b = &a;

    printf("%d", *b);
}
```



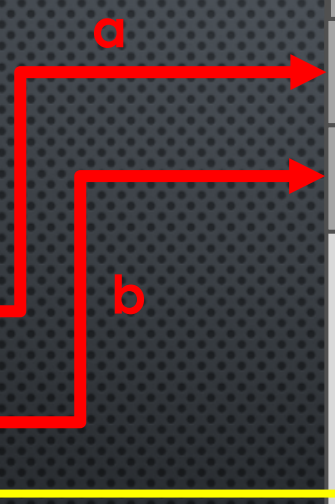
Process Memory

0xff00fa03	Start of stack frame
0xff00fa02	0
0xff00fa00	ff00fa02
0xff000000	End of stack frame

HOW POINTERS WORKS ?

Code :

```
1   #include <stdlib.h>
2   #include <stdio.h>
3
4   int main(){
5
6       // variable
7       int a = 0;
8
9       // pointer to the variable
10      int* b = &a;
11
12      printf("%d", *b);
13  }
```



Process Memory

0xff00fa03	Start of stack frame
0xff00fa02	0
0xff00fa00	ff00fa02
0xff000000	End of stack frame

b

HOW POINTERS WORKS ?

Code :

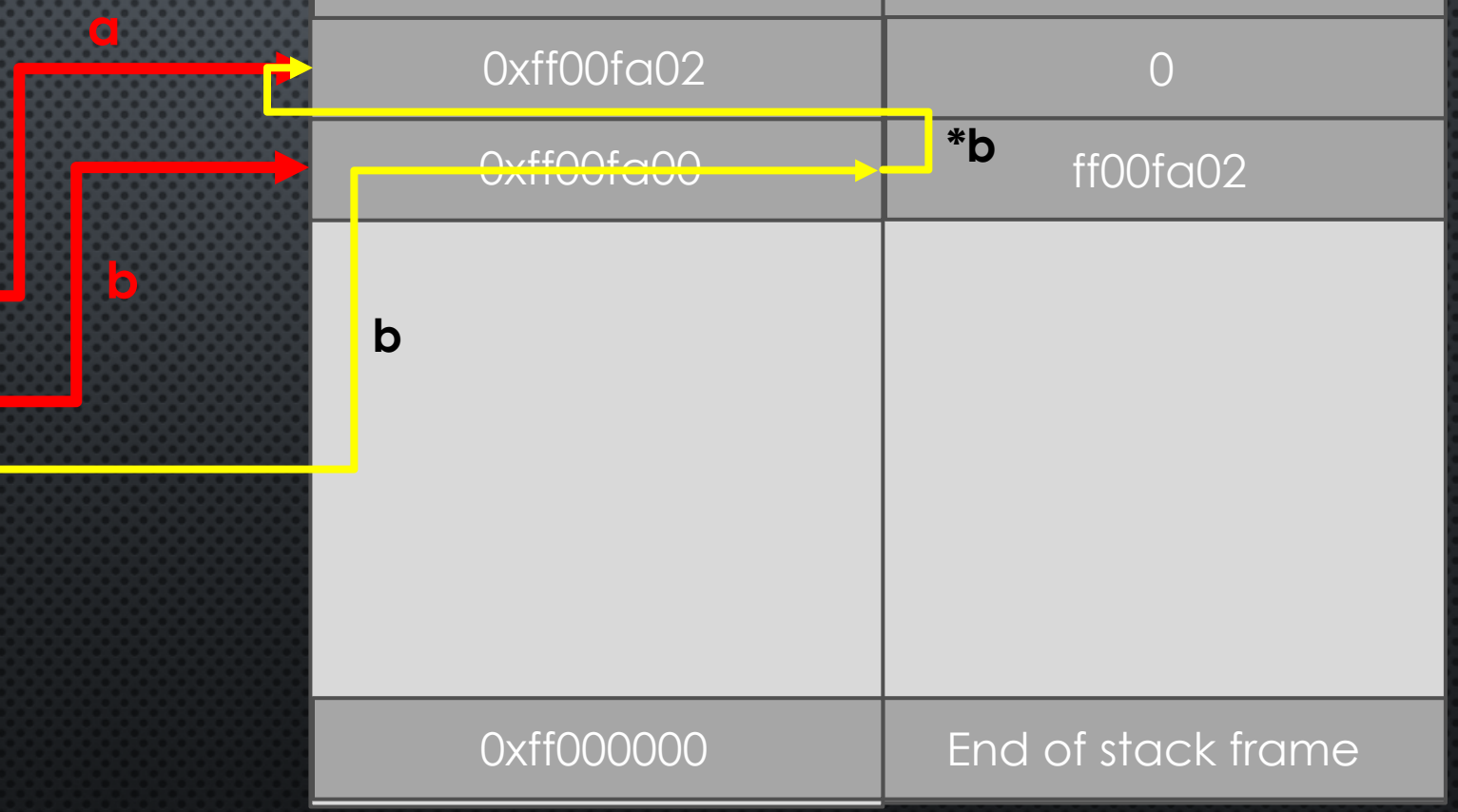
```c
#include <stdlib.h>
#include <stdio.h>

int main(){

    // variable
    int a = 0;

    // pointer to the variable
    int* b = &a;

    printf("%d", *b);
}
```



HOW POINTERS WORKS ?

Code :

```c
#include <stdlib.h>
#include <stdio.h>

int main(){

    // variable
    int a = 0;

    // pointer to the variable
    int* b = &a;

    printf("%d", *b);
}
```

Output :

```
$sh   ./a.out
0
```