Analyse and binary transformation

Guillaume Bouffard

Université de Limoges

technicolor

Analyse and binary transformation

Guillaume Bouffard

Outline

technicolor

Outline

technicolor

Technicolor Security and Content Protection Labs

Technicolor

- Creating, managing and delivering video
- For the Communication, Media and Entertainment industries.

Their works

- Cryptography
- Signal processing for security
- Content protection (DRM)
- Network security
- Tamper resistance

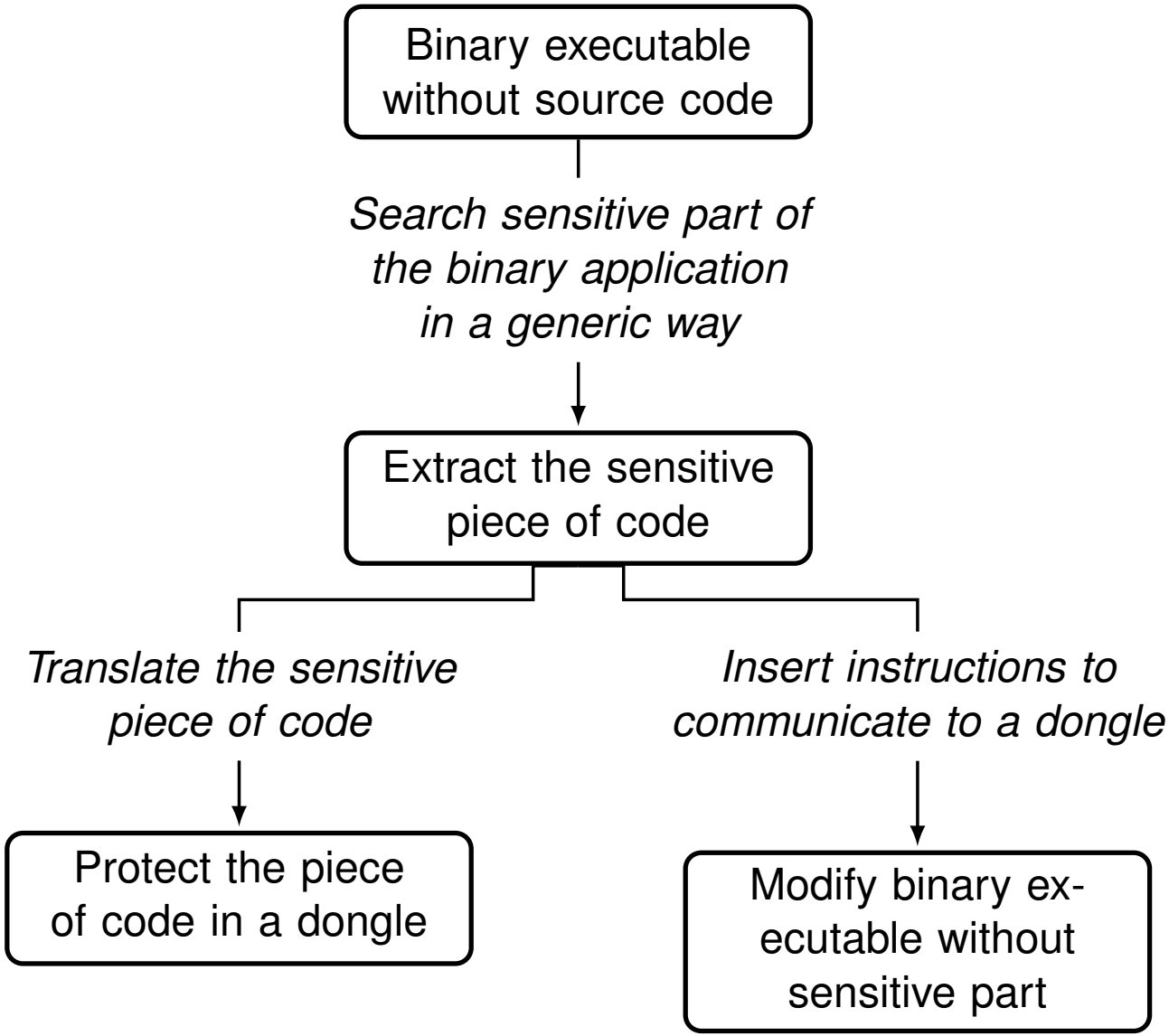
technicolor

Context

The Internship Context

- Illegal software duplication and intellectual property theft
- Software protection VS hardware protection
- Hardware protection?

Subject



technicolor

Motivation

What was my motivation?

- A blend of compilation and smart card problems
- Discover the computer science underground
- Think on a research subject

Outline

technicolor

Application Profiling

What do you want to find?

- Each executed binary piece of code
- Found the **sensitive** parts

What can tools do that?

- OProfile
- Valgrind

Outline

technicolor

Translation step

The Goal

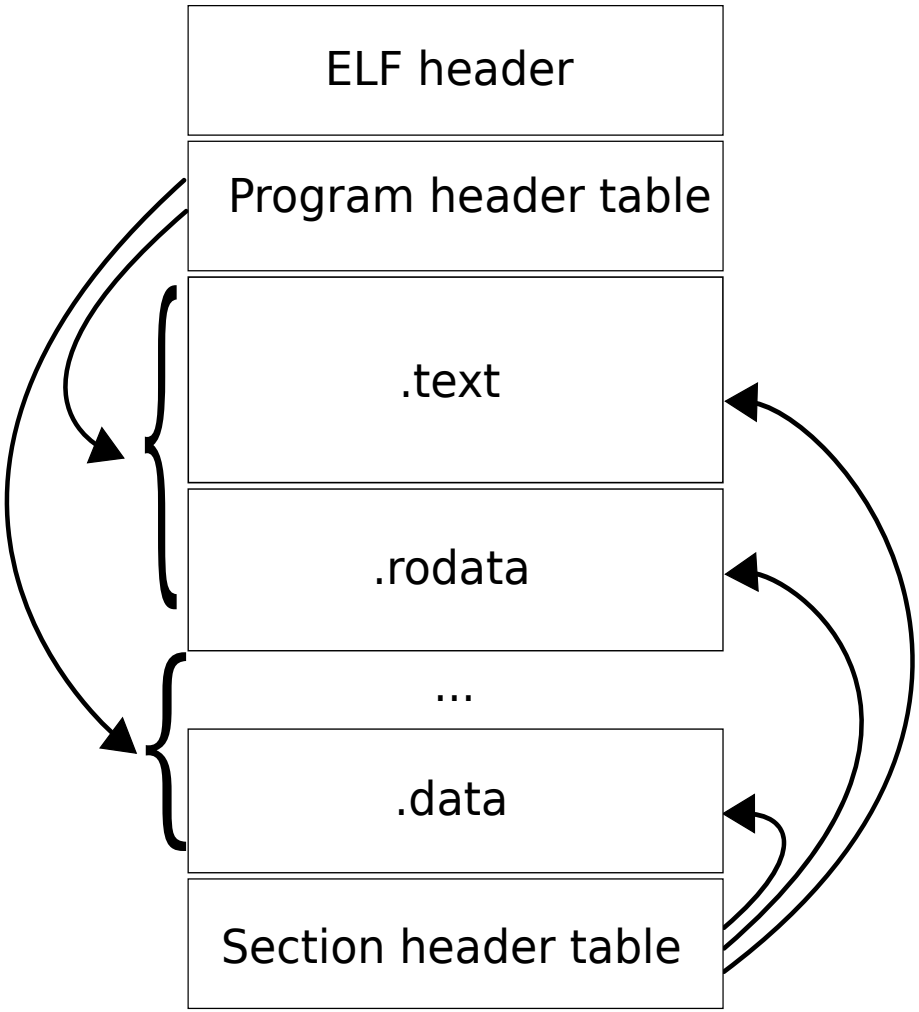
- Protect the sensitive pieces of code in a dongle
- These pieces of code are executed by the dongle

$=>$ **A solution: UQBT**

Outline

technicolor

Executable and Linkable Format



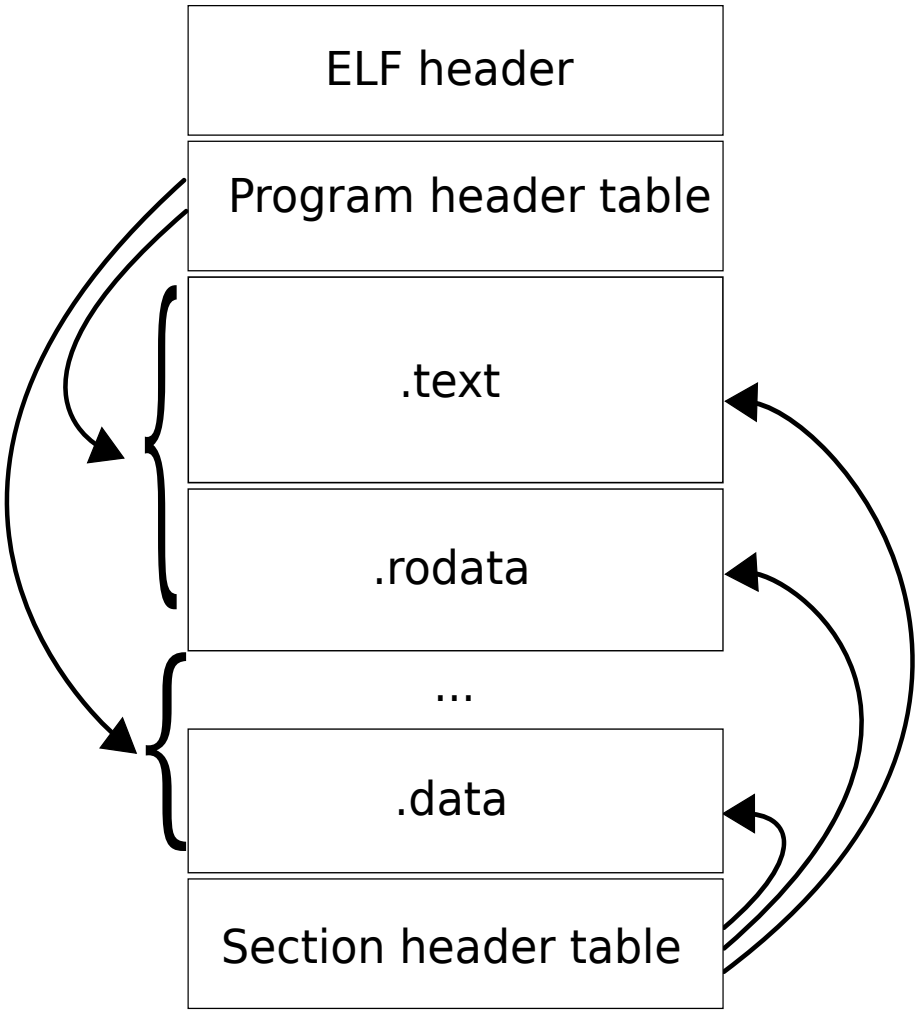
Executable and Linkable Format

- Used by Unices & GNU/Linux
- Each section are linked

How can I modify this file format?

technicolor

Executable and Linkable Format

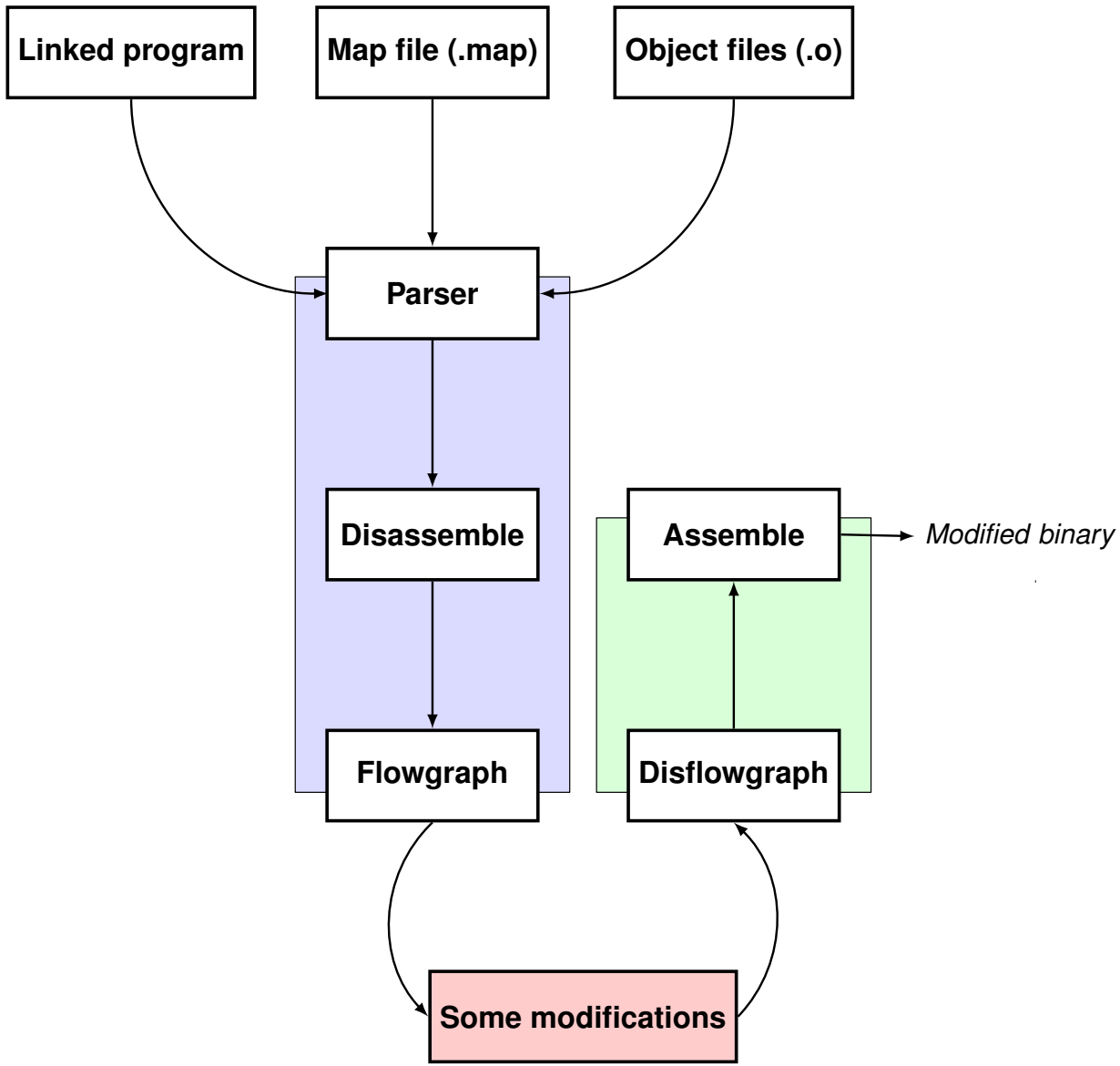


Executable and Linkable Format

- Used by Unices & GNU/Linux
- Each section are linked

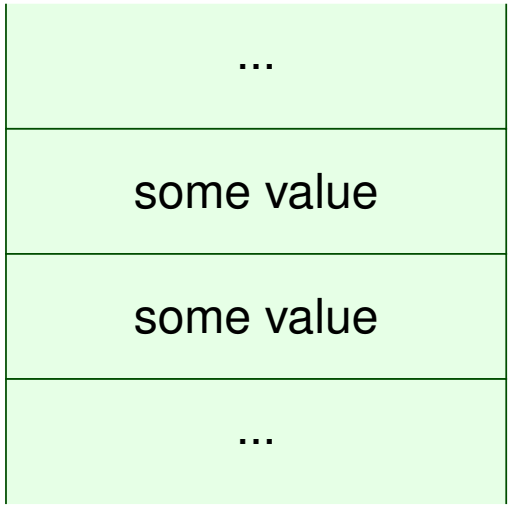
How can I modify this file format?

Diablo



technicolor

Brief overview of assembler



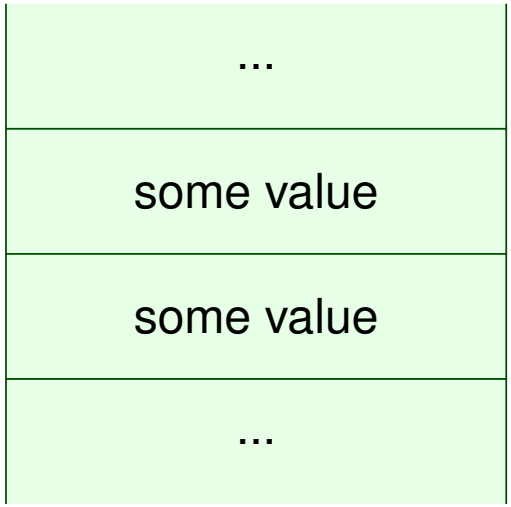

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



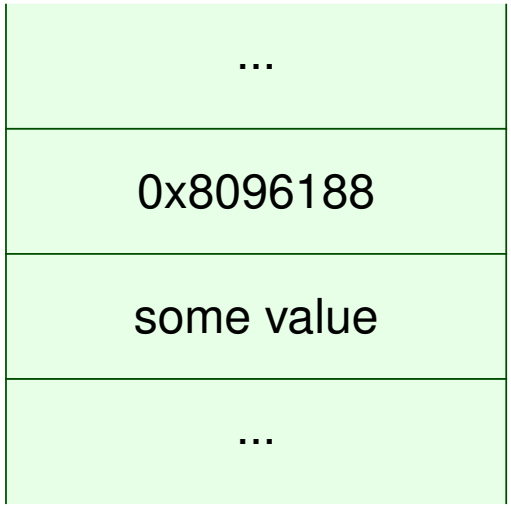

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



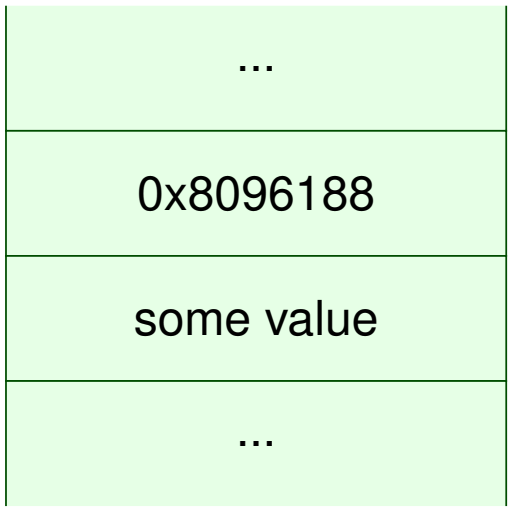

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



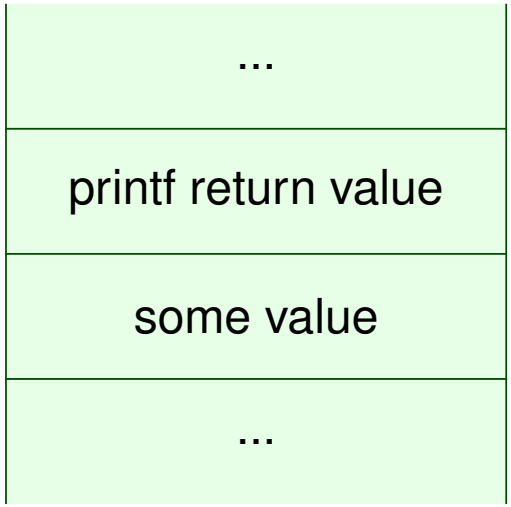

```
$ ./hello_world
hello world
```

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



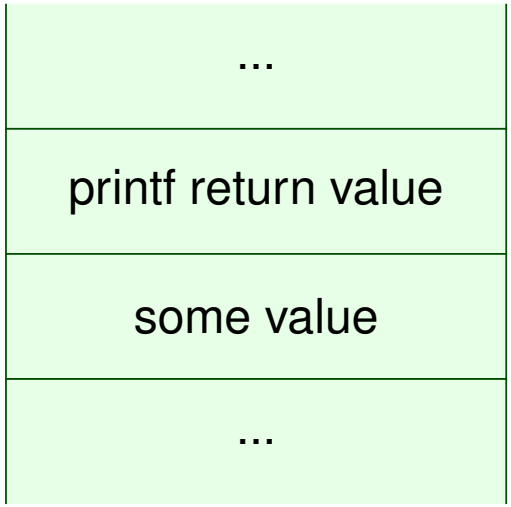

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



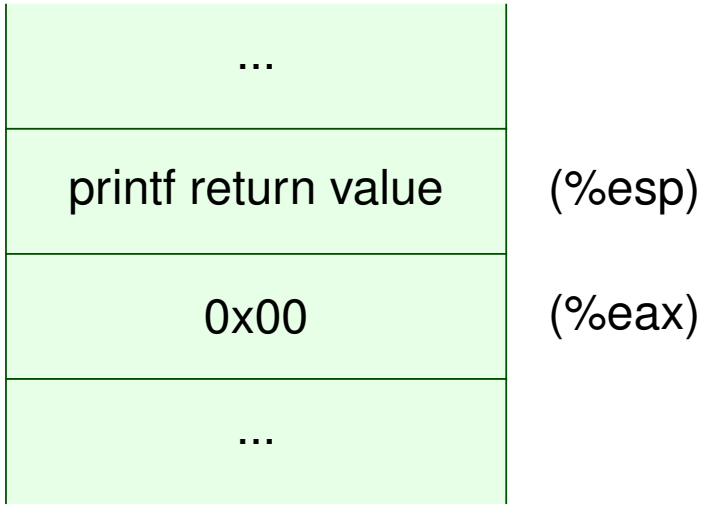

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



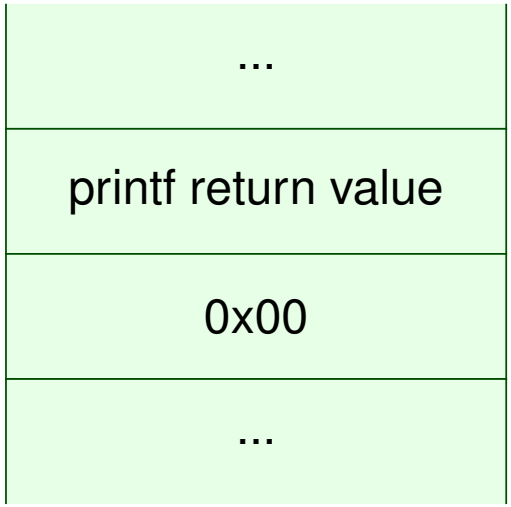

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



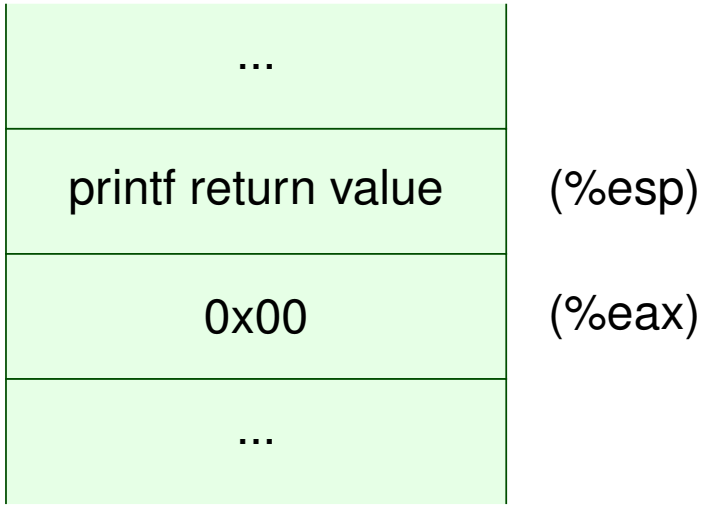

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
$ ./hello_world
hello world
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Brief overview of assembler



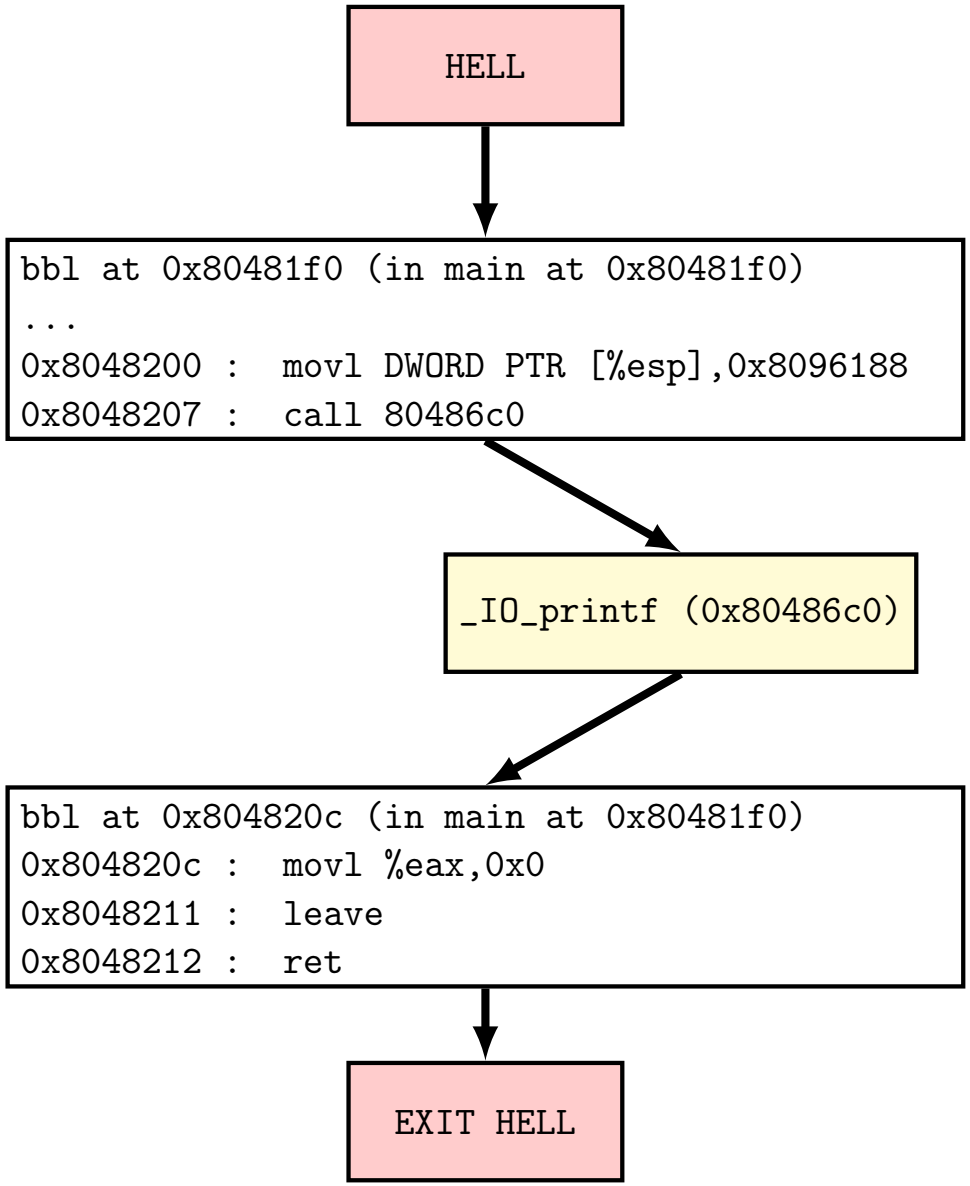

```
$ ./hello_world
hello world
```

```
#include <stdio.h>

int main ( void ) {
⇒printf("hello world\n");
⇒return EXIT_SUCCESS ;
⇒}
```

```
<main>:
⇒mov DWORD PTR [esp],0x8096188
⇒call 80486c0 <_IO_printf>
⇒mov eax,0x0
⇒leave
⇒ret
```


technicolor

Hello World

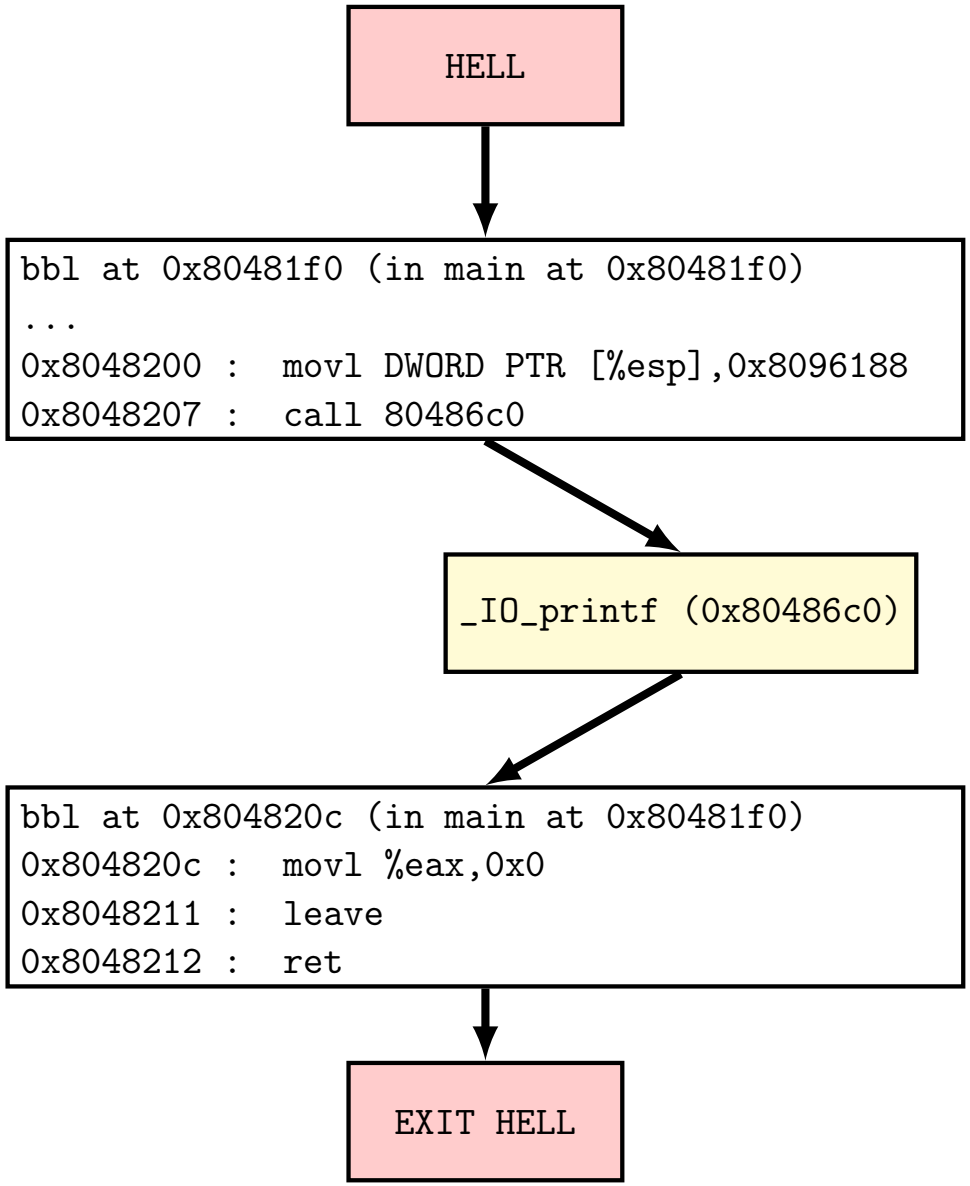



```c
int MyFunction (char *msg)
{
FILE * file = fopen
( "output" , "w" );
fprintf(file,msg);
fclose(file);
return EXIT_SUCCESS;
}
```

MyFunction.o

technicolor

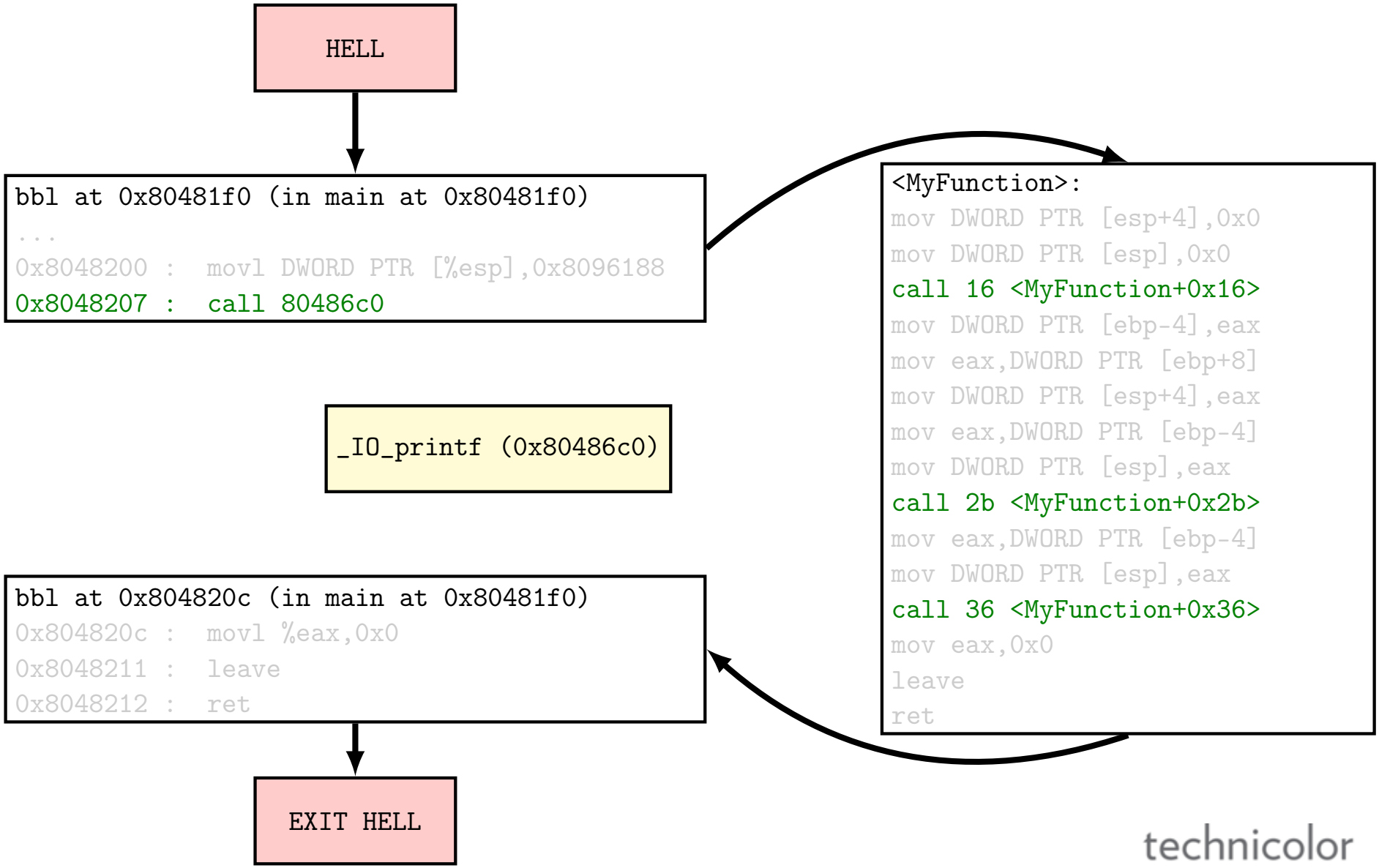
Hello World




```
<MyFunction>:
mov DWORD PTR [esp+4],0x0
mov DWORD PTR [esp],0x0
call 16 <MyFunction+0x16>
mov DWORD PTR [ebp-4],eax
mov eax,DWORD PTR [ebp+8]
mov DWORD PTR [esp+4],eax
mov eax,DWORD PTR [ebp-4]
mov DWORD PTR [esp],eax
call 2b <MyFunction+0x2b>
mov eax,DWORD PTR [ebp-4]
mov DWORD PTR [esp],eax
call 36 <MyFunction+0x36>
mov eax,0x0
leave
ret
```

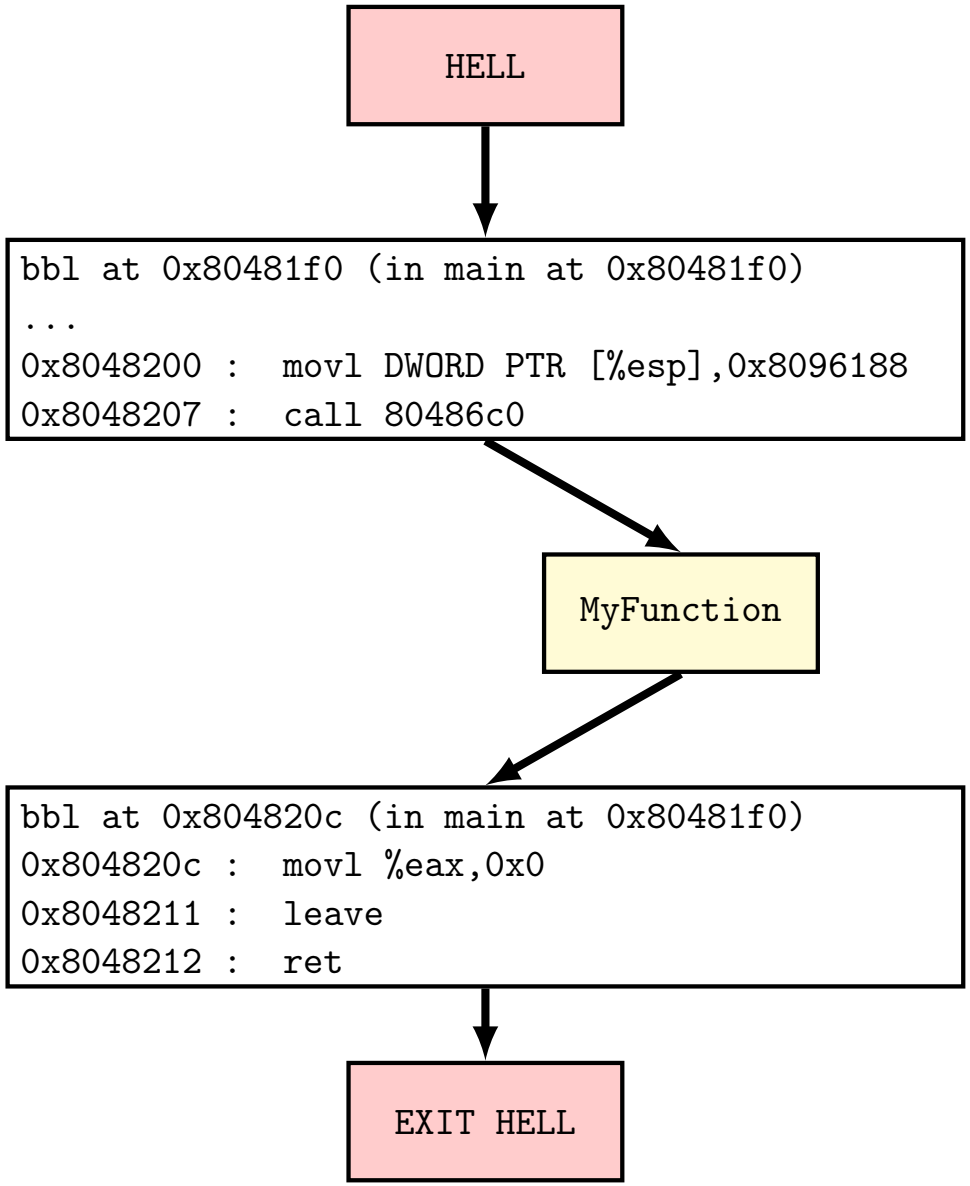

technicolor

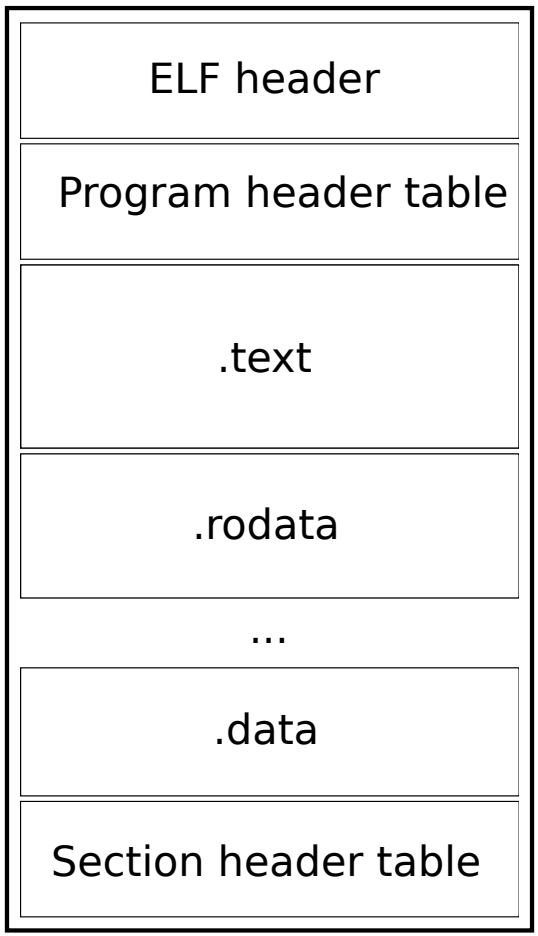
Hello World



technicolor

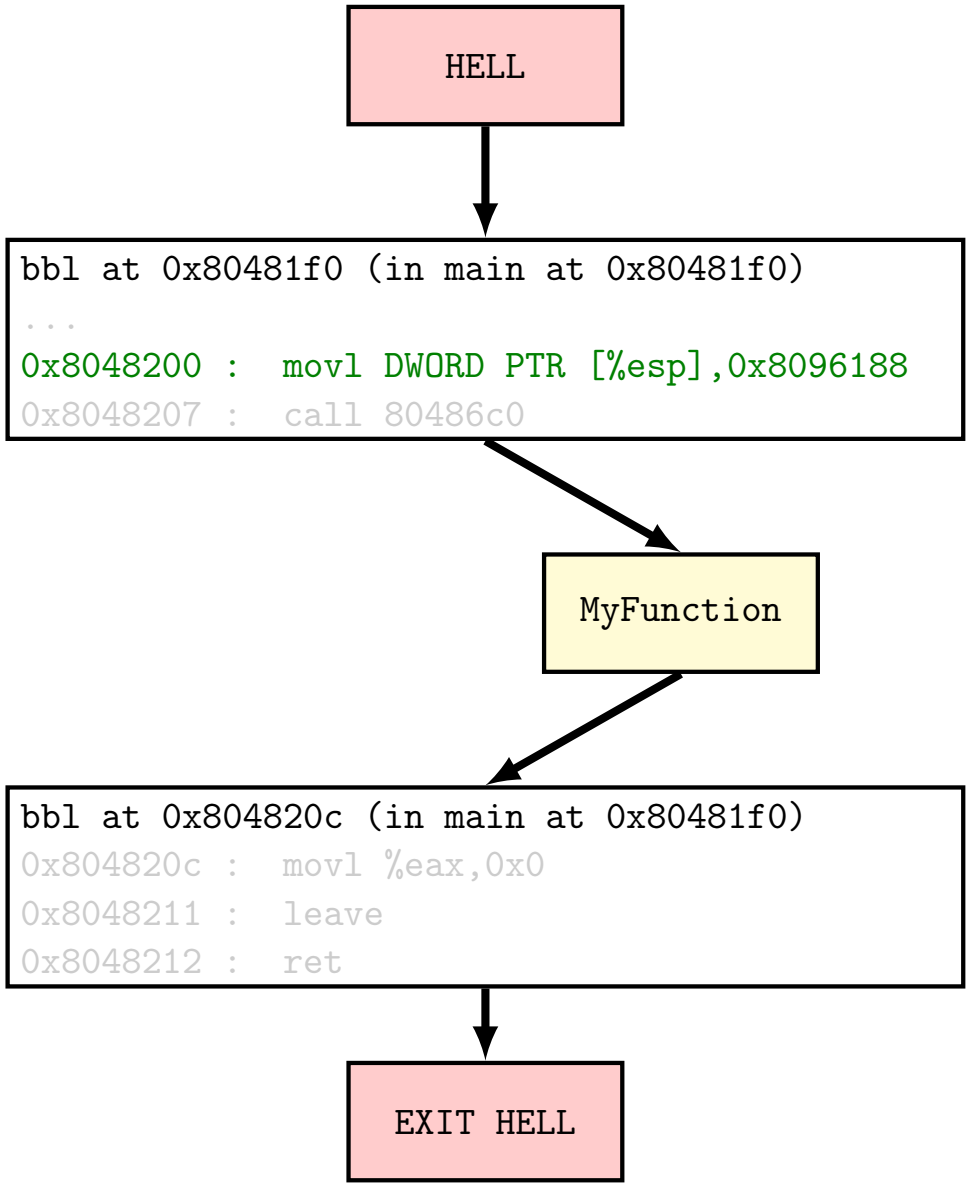
CouCou World

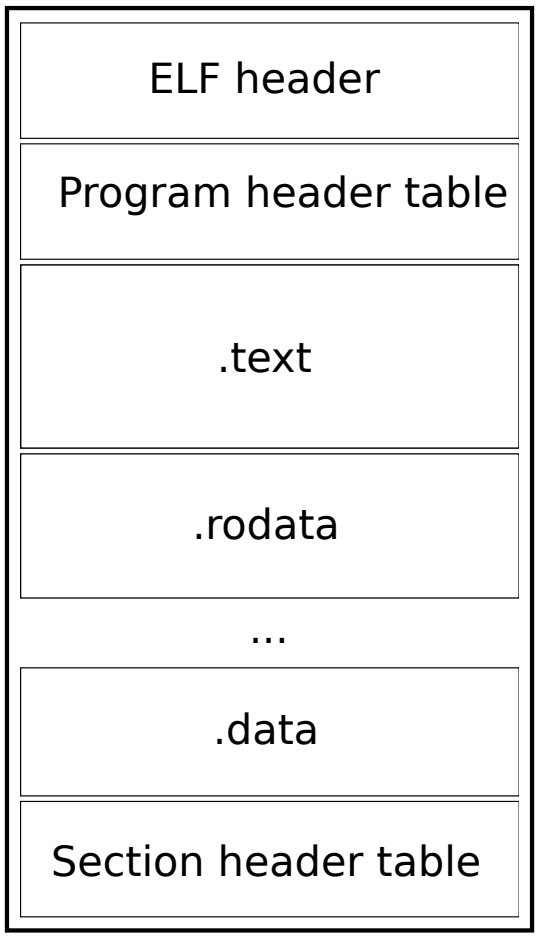




technicolor

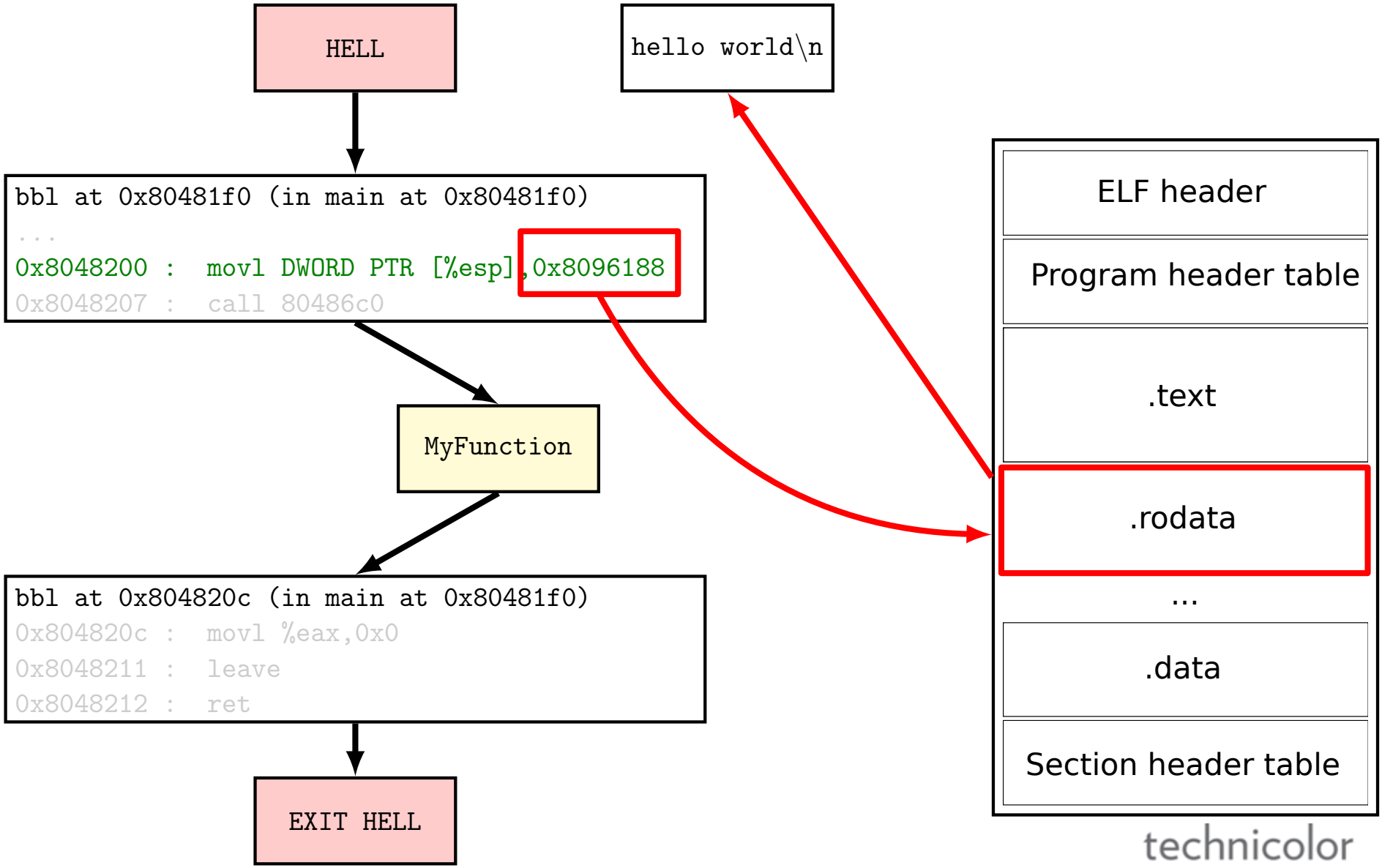
CouCou World





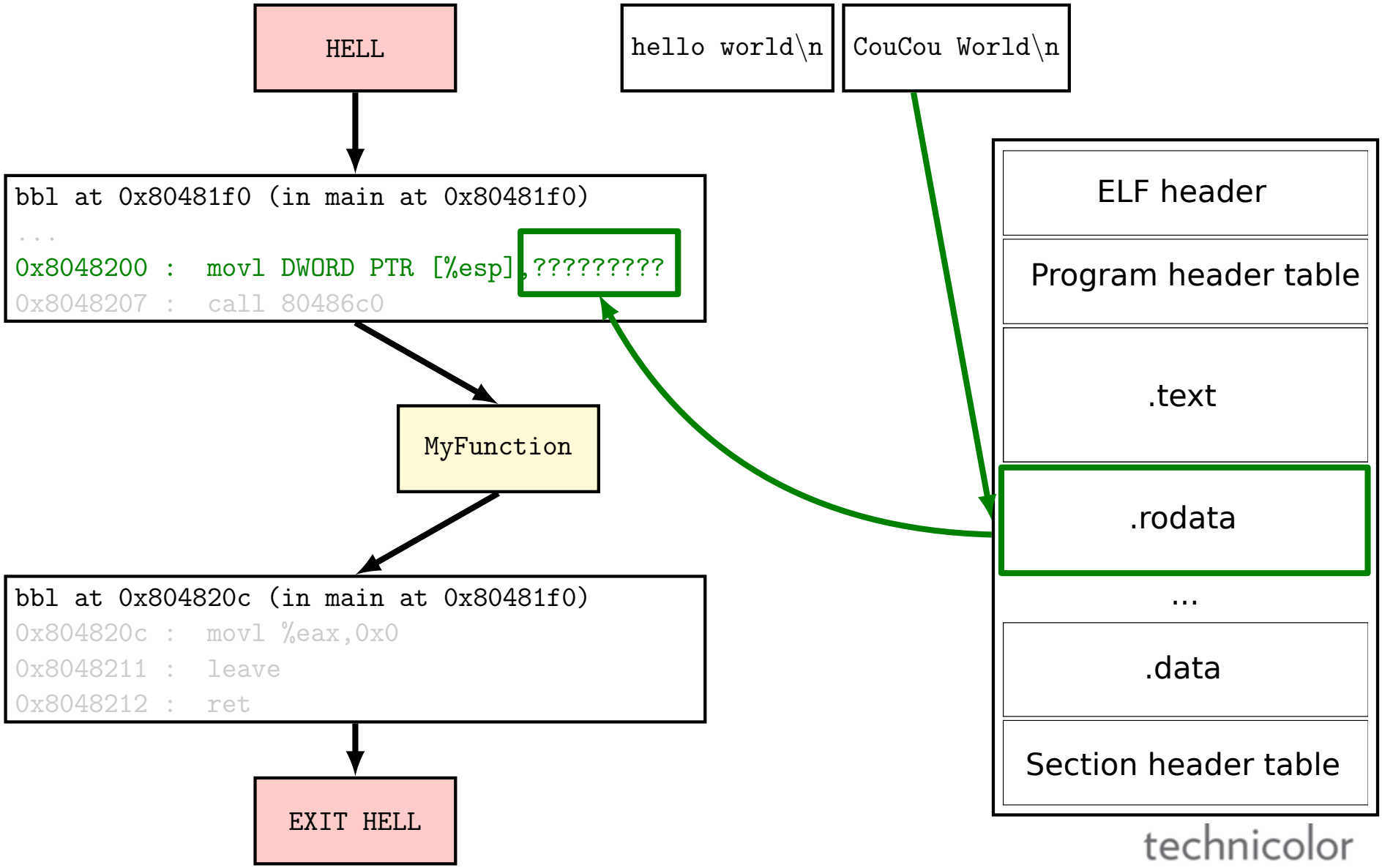
technicolor

CouCou World



technicolor

CouCou World



technicolor

Outline

technicolor

Integers Multiplication

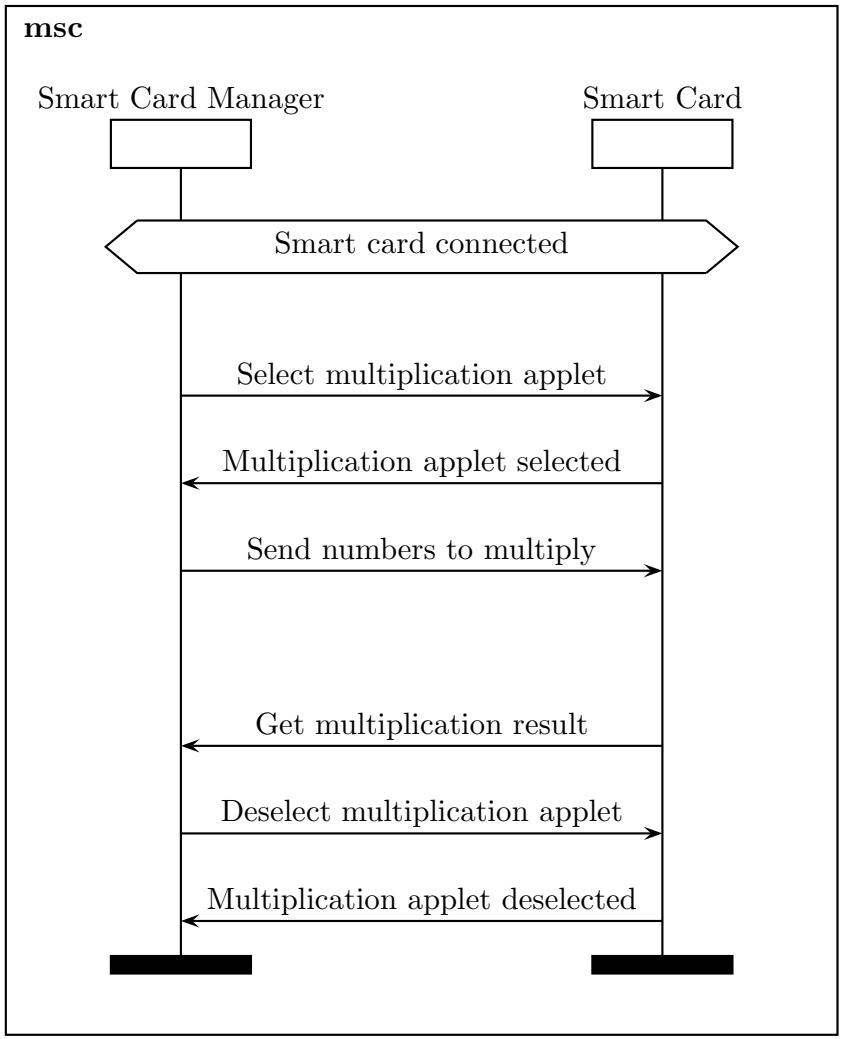
Main Idea

- Use a simple product matrix
- Make each multiplication operation on a smart card
- Search & replace each multiplication instruction

An Integers Multiplication on a Java Card

- Java Card cannot make a 32-bit number multiplication

Communication Binary Application ⇔ the Smart Card

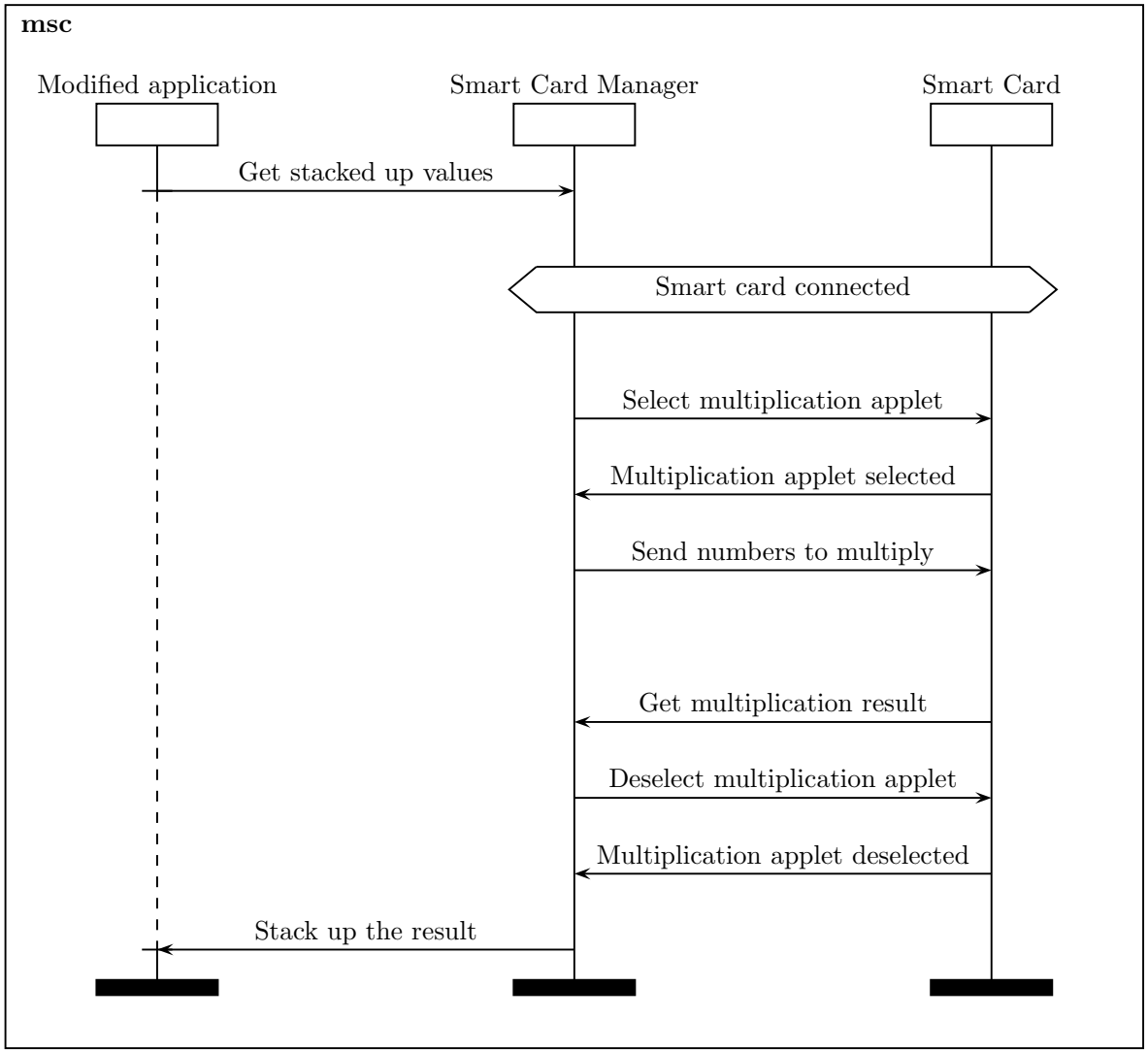


Implementation

- Using a framework made by laboratory members
- Override `libpcsc-lite` to add some features
- Just a little bit complex...

technicolor

The Last Binary Modification with Diablo



Problems

- Diablo cannot parse the C++ framework...
- ...and it cannot parse `libpcsc-lite`

technicolor

Outline

technicolor

Internship conclusion

Objectives accomplished

- Can found each executed instruction without source code
- Modify binary executable with Diablo

To Do list

- Realize the translation step
- Make a complete proof of concept
- Don’t use Java Card!
- Obfuscate the APDU request
- Upgrade Diablo toolchain

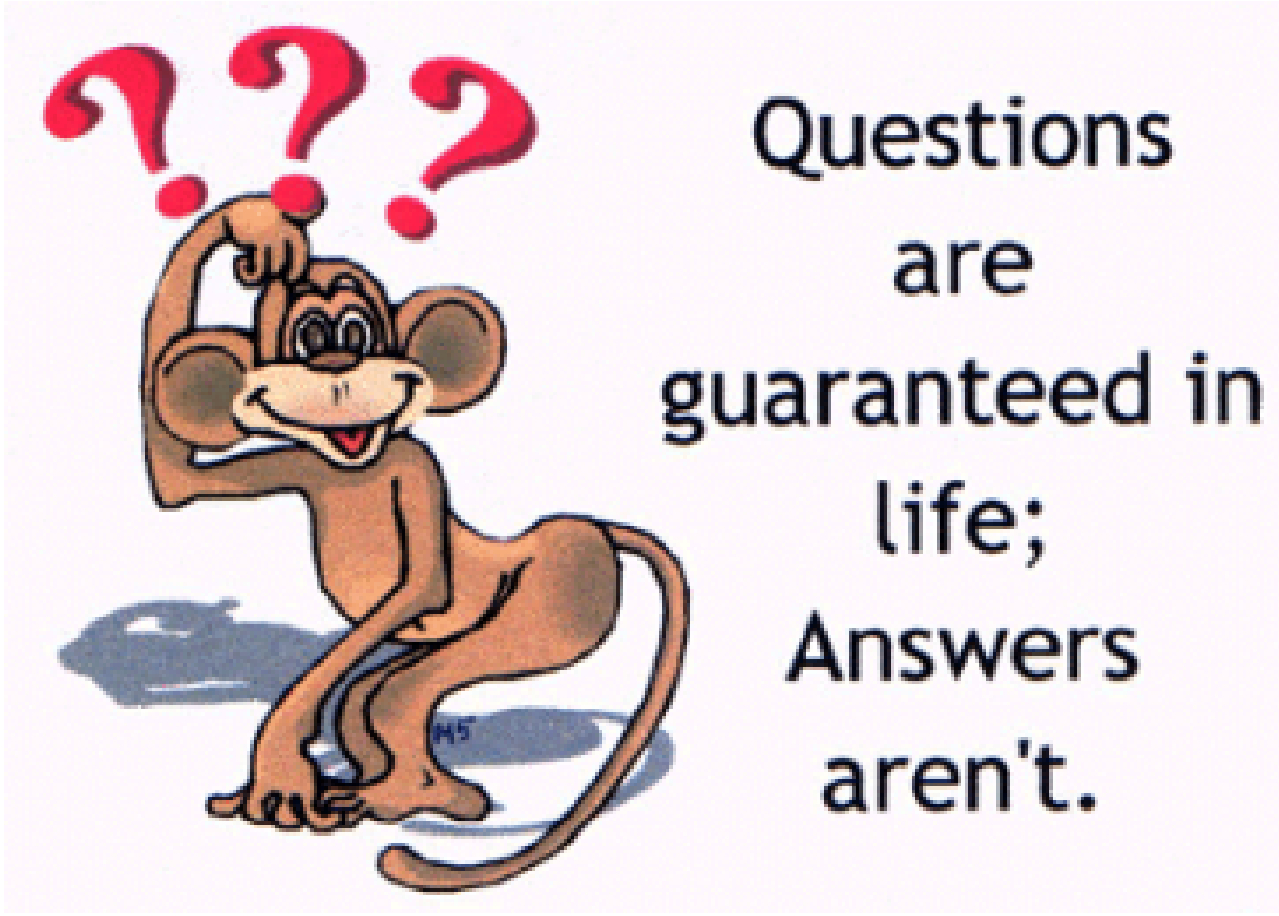
Personal Conclusion

Personal impact

- Discover a private laboratory
- With a research project

The End. . .

Thank you for your attention!
Any questions?



technicolor