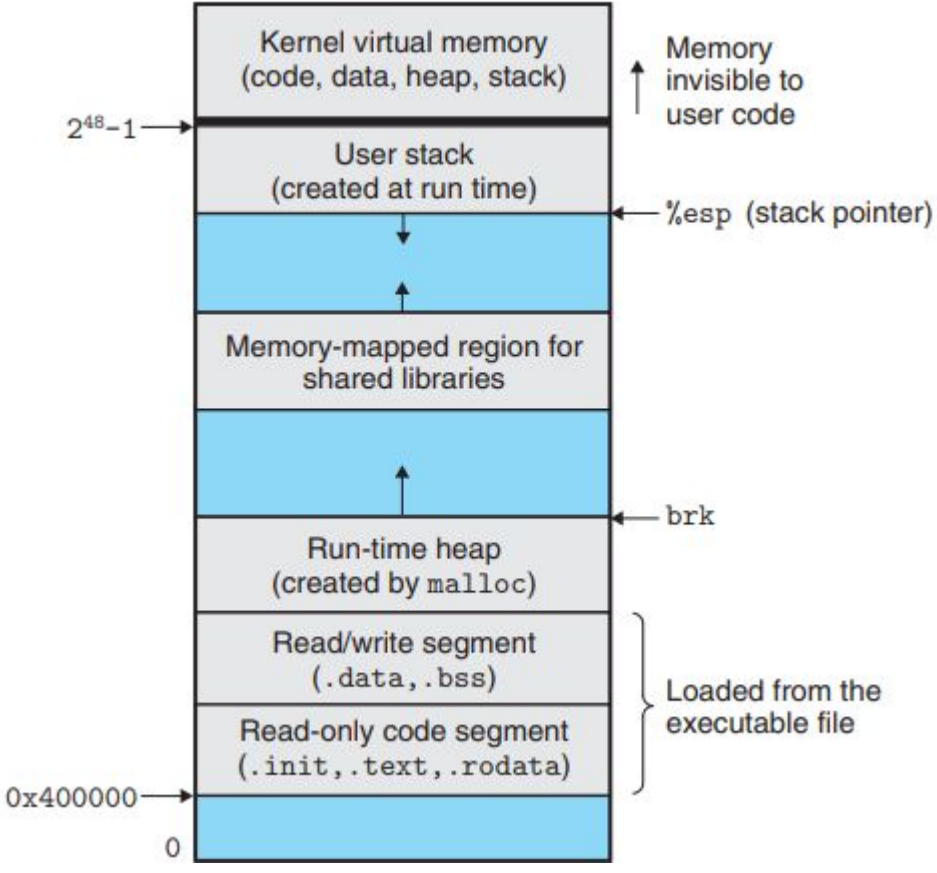
CLASE 4

Agenda

- Stack en funciones
- Registros x86-64
- Instrucciones x86-64 Intel
- Convenciones C x86
- Ejemplo CompletoStack Standards
- GDB - Stack, PC, little-big endian
- Mapa de Memoria
- Tipo de Instrucciones de Assembly
- Syntaxis de registros, immediates, etc

Stack

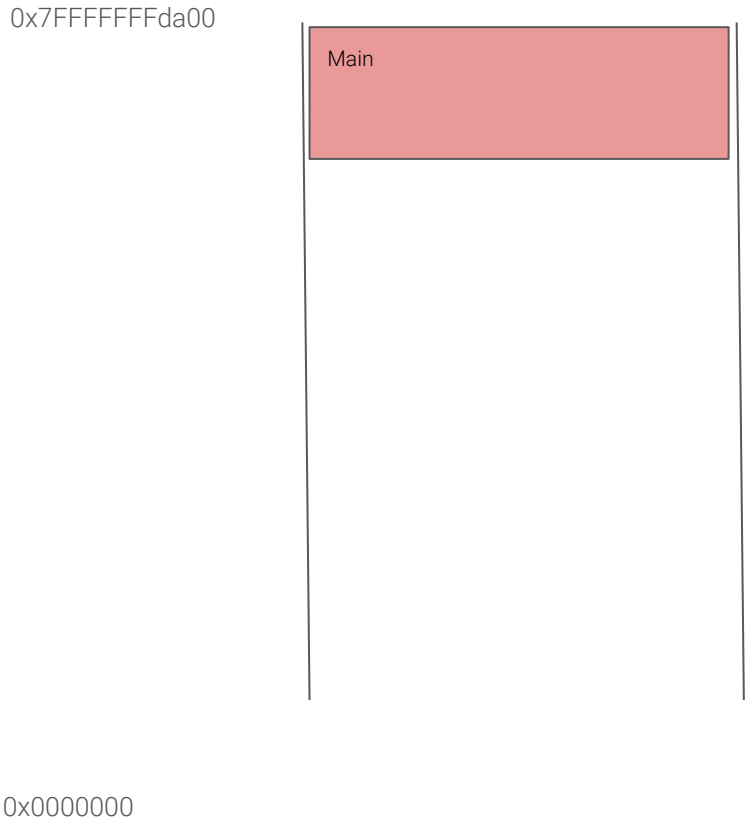

Kernel virtual memory
(code, data, heap, stack)
Memory
invisible to
user code
$2^{48}-1$
User stack
(created at run time)
%esp (stack pointer)
Memory-mapped region for
shared libraries
brk
Run-time heap
(created by malloc)
Read/write segment
(.data,.bss)
Read-only code segment
(.init,.text,.rodata)
Loaded from the
executable file
0x400000
0

Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```

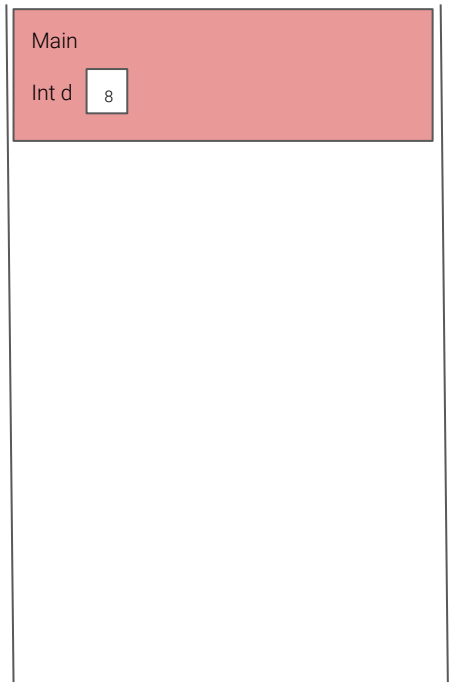


Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```

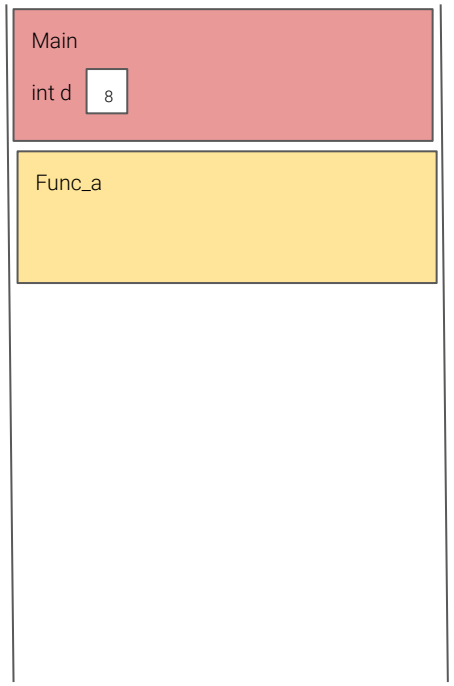


Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```

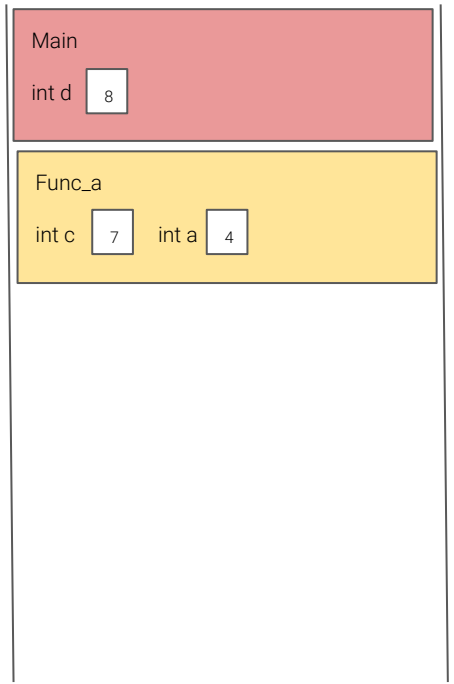


Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```

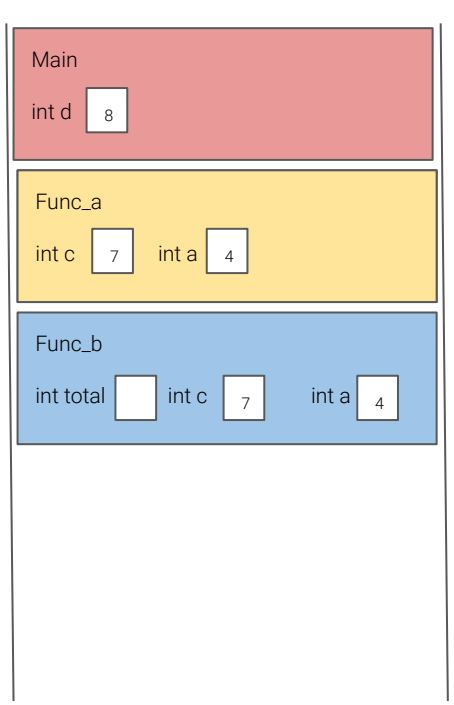


Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```

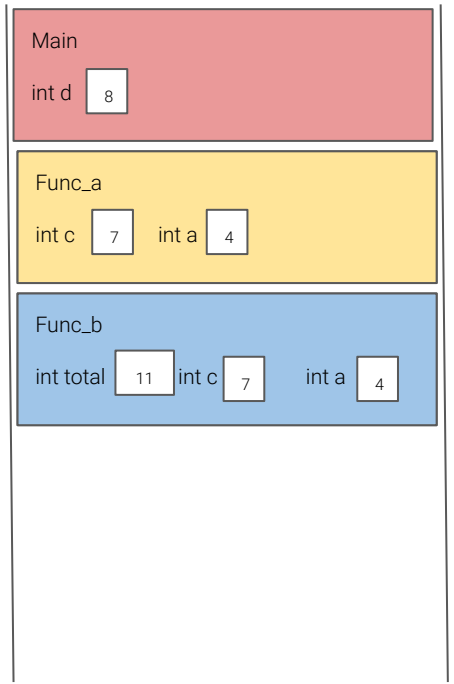


Stack

```
int func_b(int a, int c) {
   int total = a + c;
   return total;
}

int func_a(int a) {
   int c = 7;
   if ((a + c) % 2) {
       func_b(a, c);
   }
}

int main() {
   int d = 8;
   func_a(4);
}
```



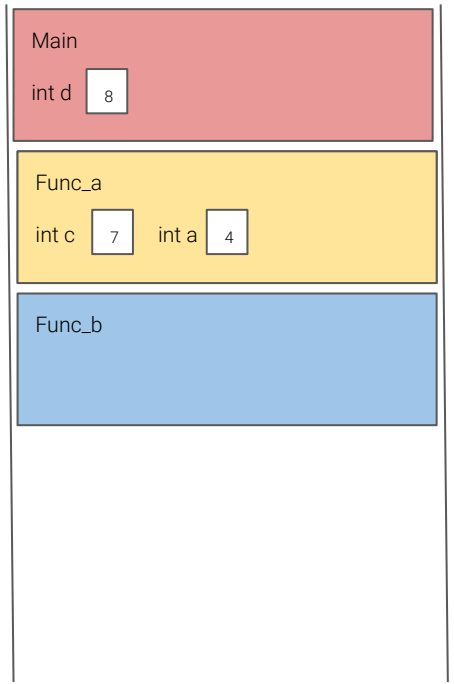
Stack - funciones subsequentes

```
void func_b(int a, int *c) {
    *c = a + *c;
}

int func_a(int a) {
    int c = 7;
    if ((a + c) % 2) {
        func_b(a, &c);
    }
}

int main() {
    int d = 8;
    func_a(4);
}
```

Se puede pasar la dir de una variable en el stack anterior



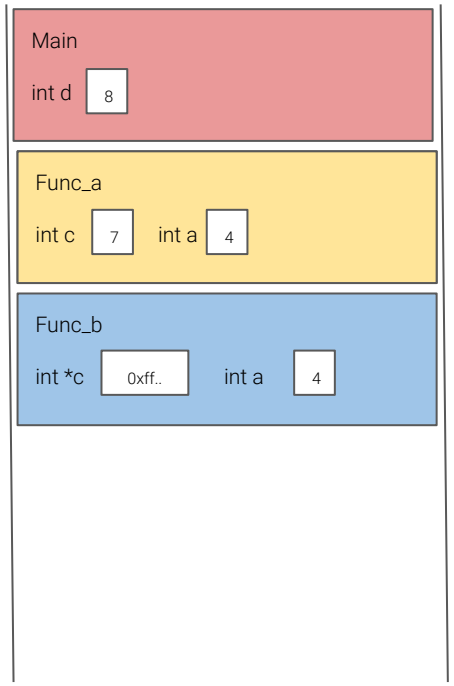
Stack - funciones subsequentes

```
void func_b(int a, int *c) {
    *c = a + *c;
}

int func_a(int a) {
    int c = 7;
    if ((a + c) % 2) {
        func_b(a, &c);
    }
}

int main() {
    int d = 8;
    func_a(4);
}
```

Se puede pasar la dir de una variable en el stack anterior



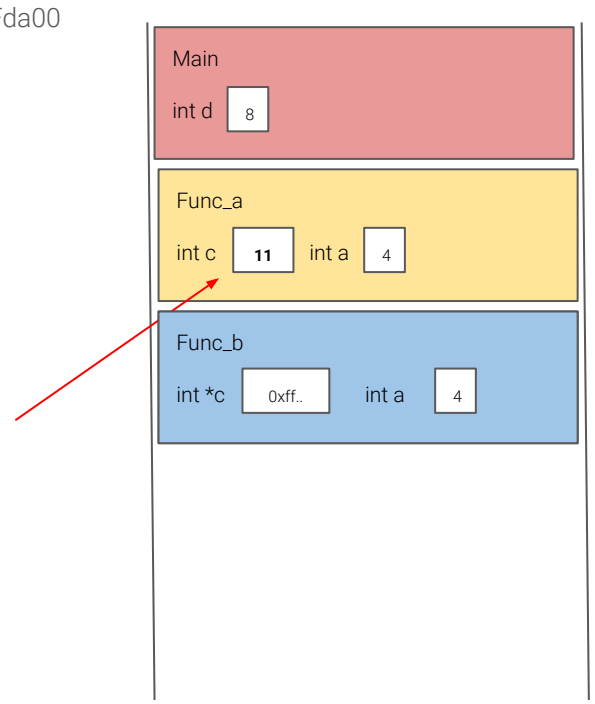
Stack - funciones subsequentes

```
void func_b(int a, int *c) {
    *c = a + *c;
}

int func_a(int a) {
    int c = 7;
    if ((a + c) % 2) {
        func_b(a, &c);
    }
}

int main() {
    int d = 8;
    func_a(4);
}
```

Se puede pasar la dir de una variable en el stack anterior



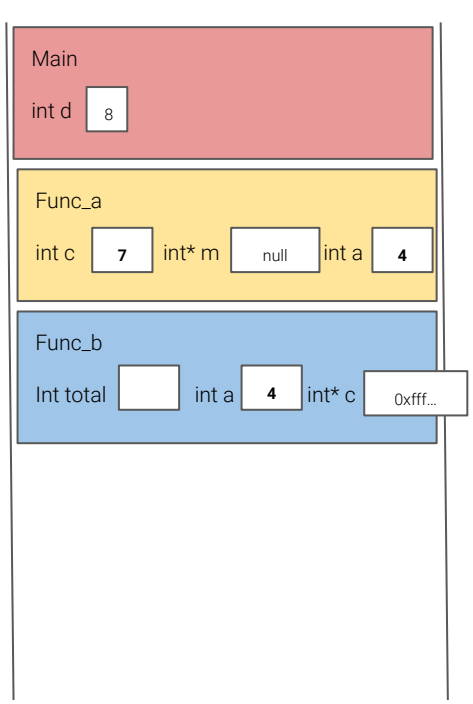
Stack - stack incorrecto

```
int* func_b(int a, int *c) {
   int total = *c;
   total = a + total;
   return &total;
}

int func_a(int a) {
   int c = 7;
   int* m = NULL;
   if ((a + c) % 2) {
       m = func_b(a, &c);
   }
   printf("m: %d\n", *m);
}


int main() {
   int d = 8;
   func_a(4);
}
```

No se puede pasar variables del stack a stacks anteriores



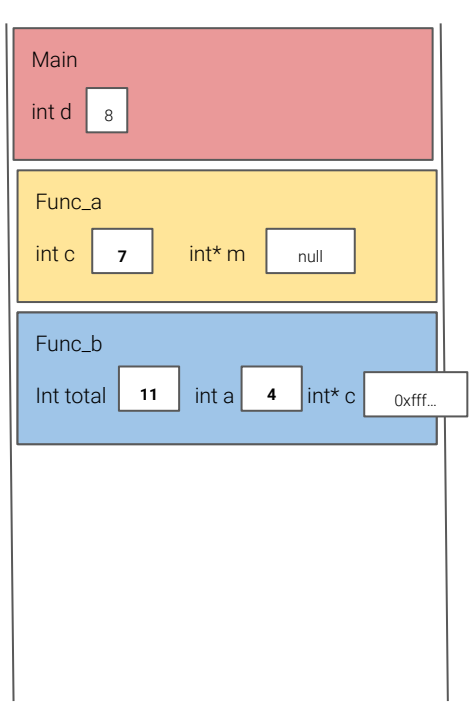
Stack

```
int* func_b(int a, int *c) {
   int total = *c;
   total = a + total;
   return &total;
}

int func_a(int a) {
   int c = 7;
   int* m = NULL;
   if ((a + c) % 2) {
       m = func_b(a, &c);
   }
   printf("m: %d\n", *m);
}


int main() {
   int d = 8;
   func_a(4);
}
```

No se puede pasar variables del stack a stacks anteriores



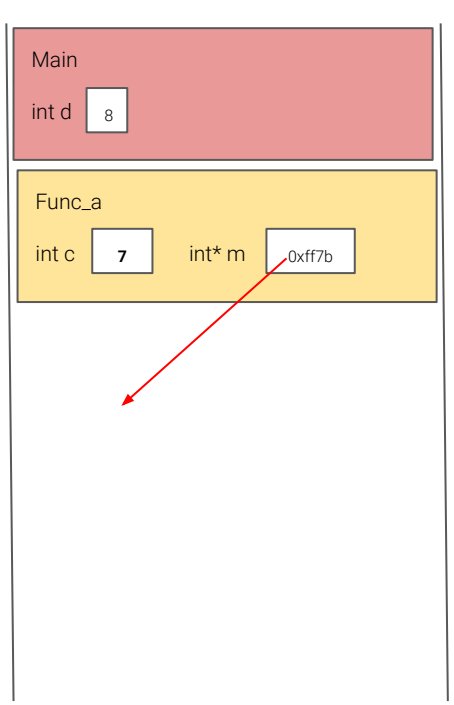
Stack - stack incorrecto

```c
int* func_b(int a, int *c) {
   int total = *c;
   total = a + total;
   return &total;
}

int func_a(int a) {
   int c = 7;
   int* m = NULL;
   if ((a + c) % 2) {
       m = func_b(a, &c);
   }
   printf("m: %d\n", *m);
}


int main() {
   int d = 8;
   func_a(4);
}
```

No se puede pasar variables del stack a stacks anteriores



Stack - stack incorrecto

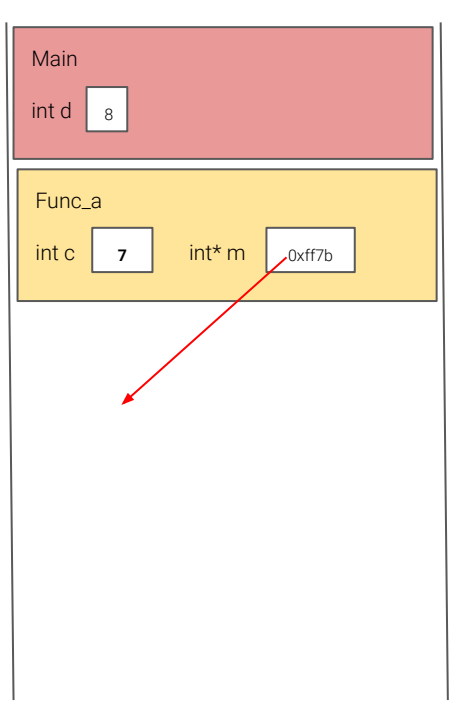
```
int* func_b(int a, int *c) {
   int total = *c;
   total = a + total;
   return &total;
}

int func_a(int a) {
   int c = 7;
   int* m = NULL;
   if ((a + c) % 2) {
       m = func_b(a, &c);
   }
   printf("m: %d\n", *m);
}

int main() {
   int d = 8;
   func_a(4);
}
```

No se puede pasar variables del stack a stacks anteriores

Acceso a memoria no valida

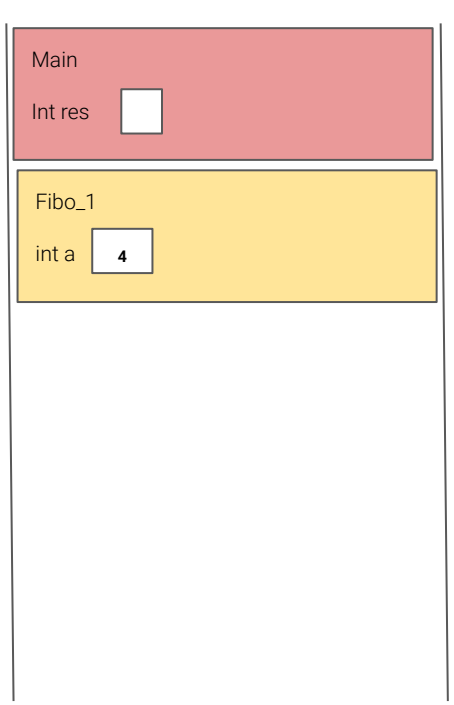


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack



Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

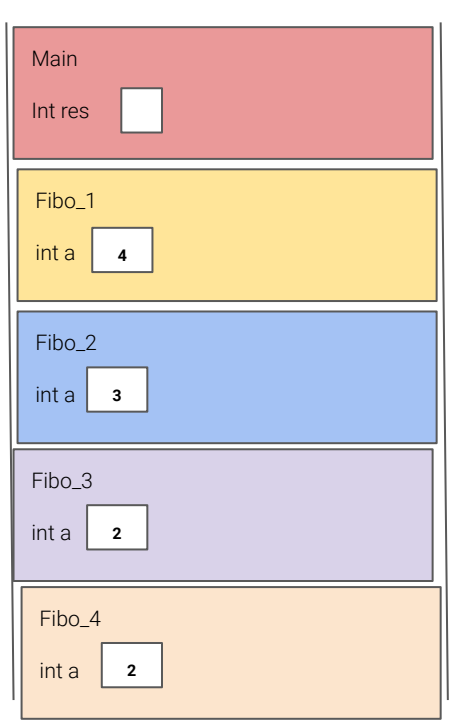
Cada llamando a función tiene su stack

Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

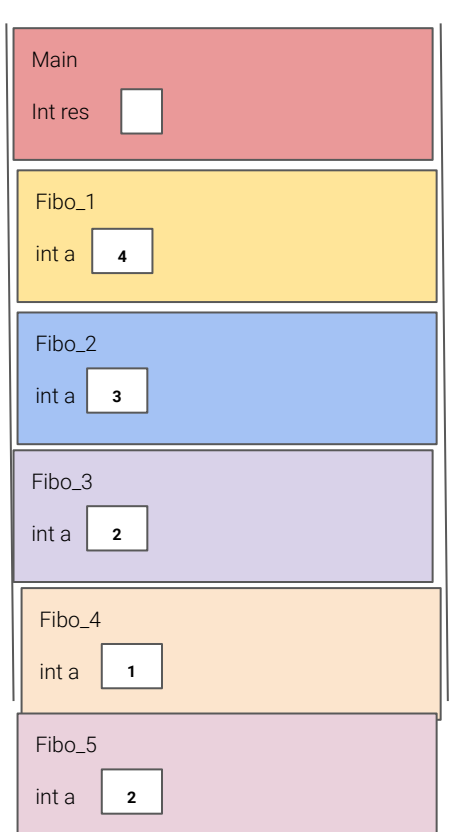


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

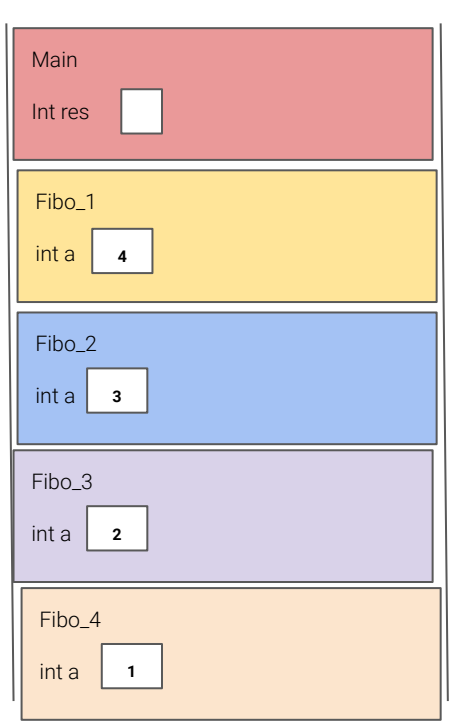


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

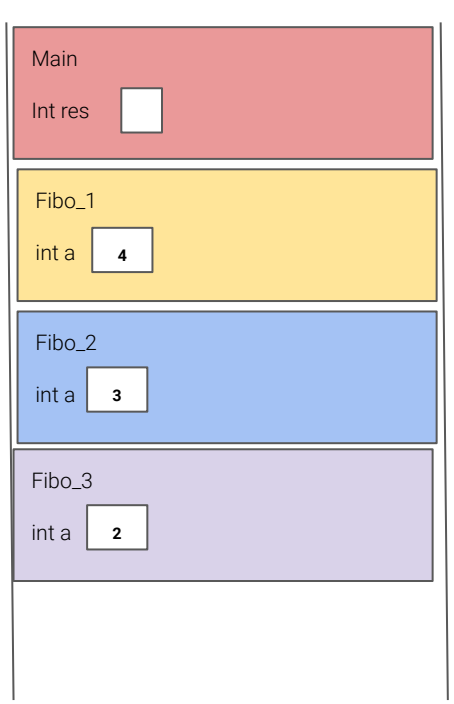


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

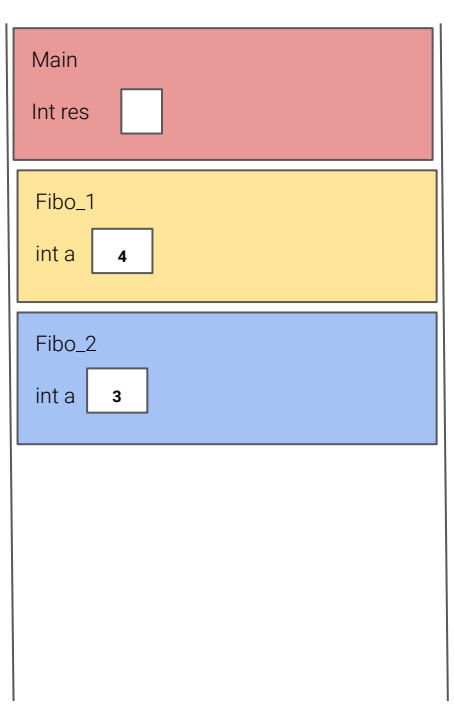


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

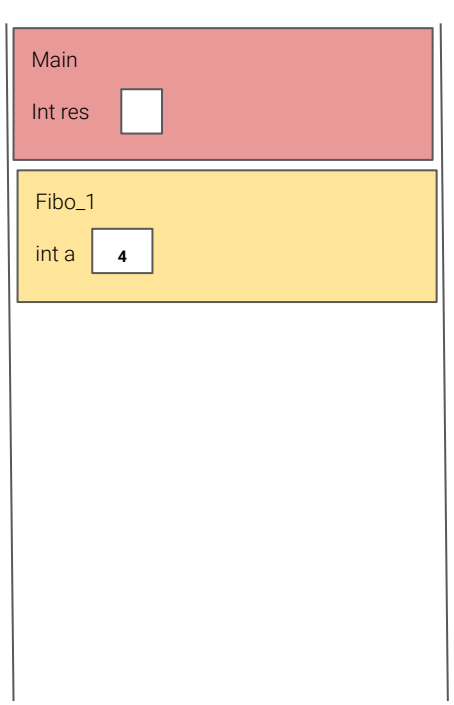


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack

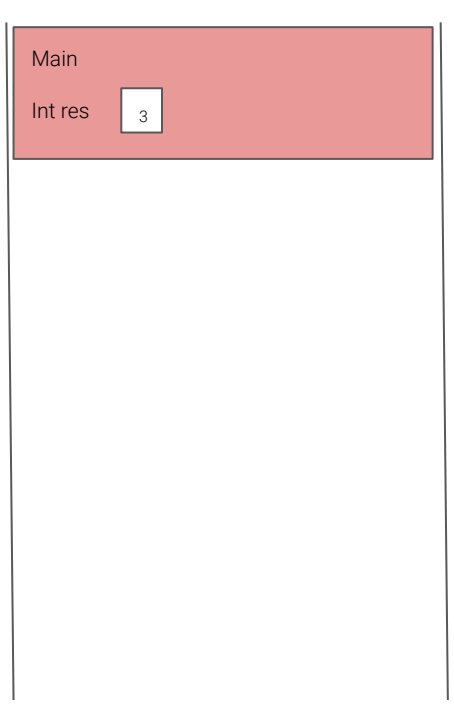


Stack - Fibonacci

```
int fibo(int a){
   if (a <= 2) {
       return 1;
   }
   return fibo(a-1) + fibo(a-2);
}

int main() {
   int res = fibo(4);
}
```

Cada llamando a función tiene su stack



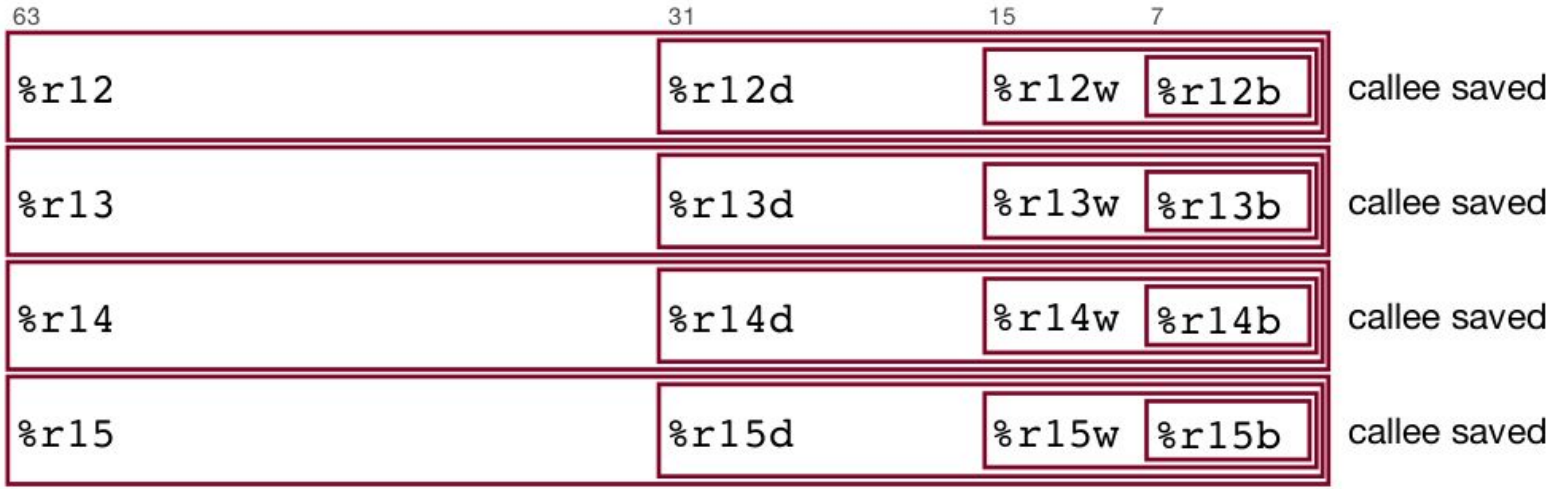
Registros - Registers

63	31	15	7	
%rax	%eax	%ax	%al	return value
%rbx	%ebx	%bx	%bl	callee saved
%rcx	%ecx	%cx	%cl	4th argument
%rdx	%edx	%dx	%dl	3rd argument
%rsi	%esi	%si	%sil	2nd argument
%rdi	%edi	%di	%dil	1st argument

Registros - Registers

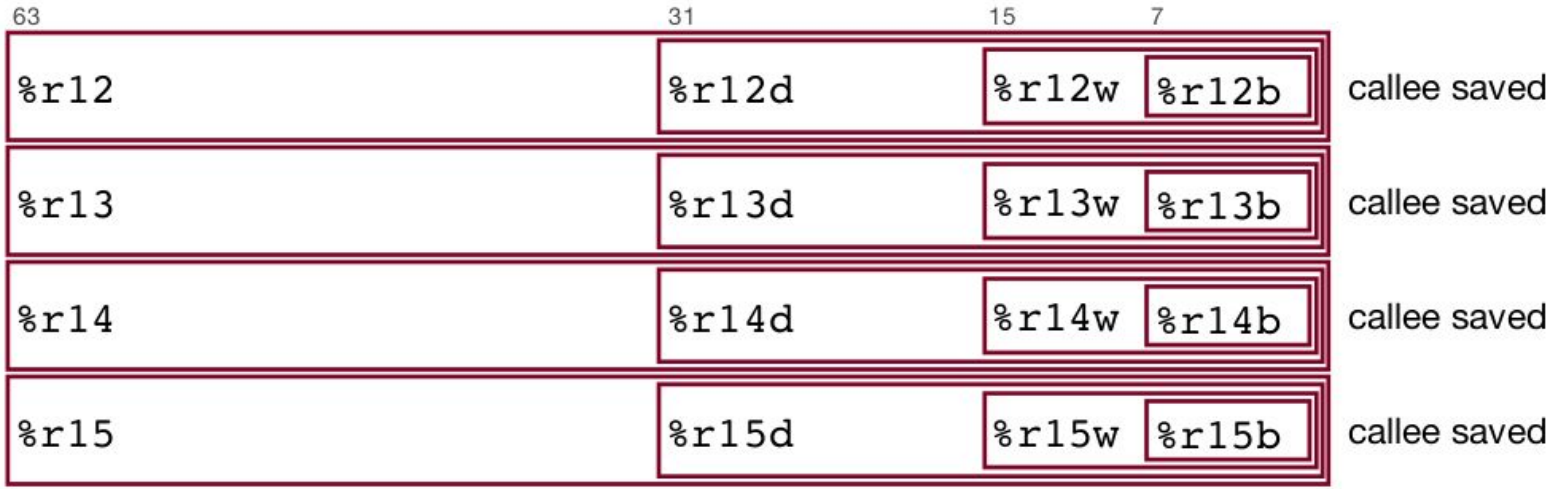
63	31	15	7	
%rbp	%ebp	%bp	%bpl	callee saved
%rsp	%esp	%sp	%spl	stack pointer
%r8	%r8d	%r8w	%r8b	5th argument
%r9	%r9d	%r9w	%r9b	6th argument
%r10	%r10d	%r10w	%r10b	caller saved
%r11	%r11d	%r11w	%r11b	caller saved

Registros - Registers



Caller Saved: Son registros volatiles, si una funcion quiere preservar su valor, debe guardarlo en el Stack. El que llama debe guardar antes de llamar

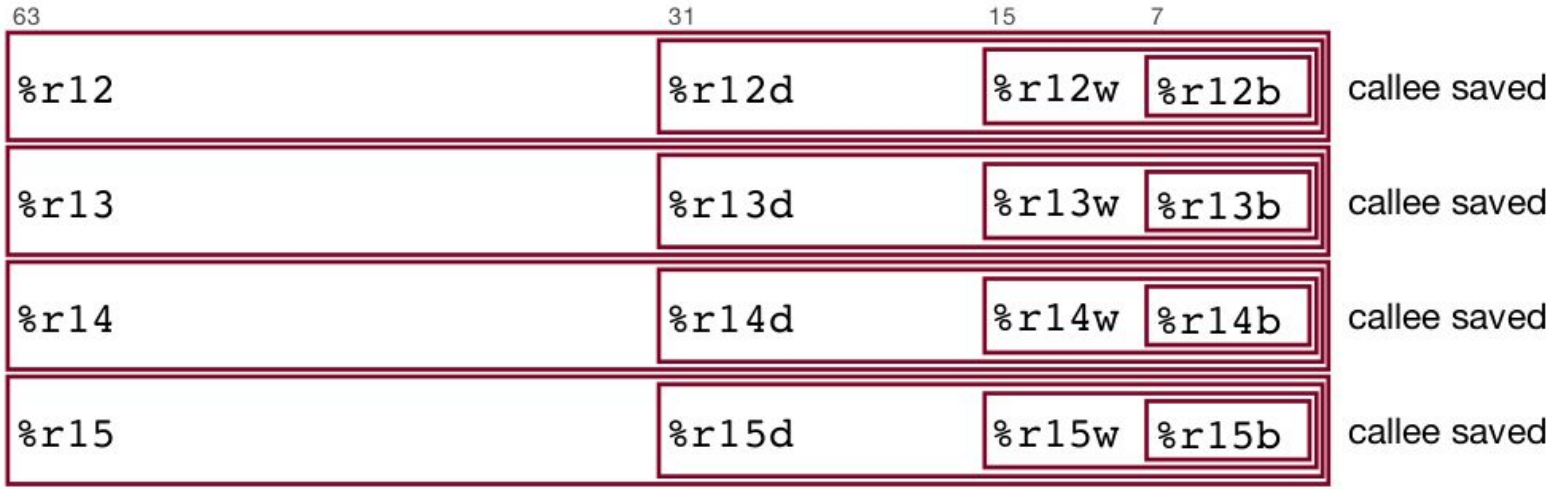
Registros - Registers



Caller Saved: Son registros volatiles, si una función quiere preservar su valor, debe guardarlo en el Stack. El que llama debe guardar antes de llamar

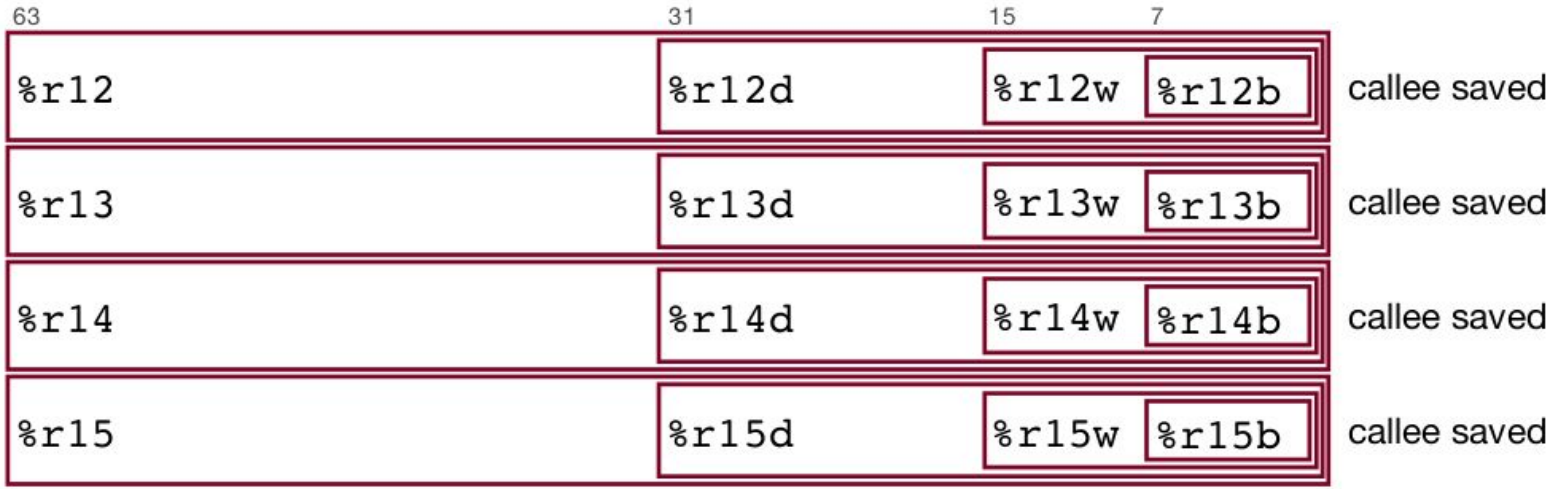
Callee Saved: Son registros que deben preservar el valor después de un llamado (call), por lo tanto si la función llamada quiere usarlos, debe guardar y restaurar su valor después de usarlo

Registros - Registers



Mas adelante lo vamos a ver en detalle

Registros - Registers



Mas adelante lo vamos a ver en detalle

Registros - Registers


```
(gdb) p/x $rax
$3 = 0x55555555517e
(gdb) p/x $eax
$4 = 0x5555517e
(gdb) p/x $ax
$5 = 0x517e
(gdb) p/x $al
$6 = 0x7e
(gdb)
```

Los nombres son partes del mismo registro.

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

Callee Saved: Guardar rbp en el Stack

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

Estamos usando syntax de intel, destino está a la izquierda. Syntax de AT&T es al reves y distinto

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

Tamaño de instrucciones varia

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

Mover a lo que apunta el rsp al registro rbp.
Restarle el tamaño de lo movido al rsp

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

Restarle 0x18 rsp, y guardar en rsp. por que?

```
0000000000001147 <func_a>:
   1147:       f3 0f 1e fa             endbr64
   114b:       55                      push   rbp
   114c:       48 89 e5                mov    rbp,rsp
   114f:       48 83 ec 18             sub    rsp,0x18
   1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
   1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
   115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
   1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
   1163:       01 d0                   add    eax,edx
   1165:       83 e0 01                and    eax,0x1
   1168:       85 c0                   test   eax,eax
   116a:       74 0f                   je     117b <func_a+0x34>
   116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
   116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
   1172:       89 d6                   mov    esi,edx
   1174:       89 c7                   mov    edi,eax
   1176:       e8 ae ff ff ff          call   1129 <func_b>
   117b:       90                      nop
   117c:       c9                      leave
   117d:       c3                      ret
```

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Mover 4 bytes del registro edi, al lugar de memorias de [rbp-0x13]. Que tiene rbp?

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Hacer bit operation and entre 4 bytes de rax y 0x1

Assembly instructions Intel

```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Es un and, que setea el Signed Flag (SF) si eax es negativo

Assembly instructions Intel

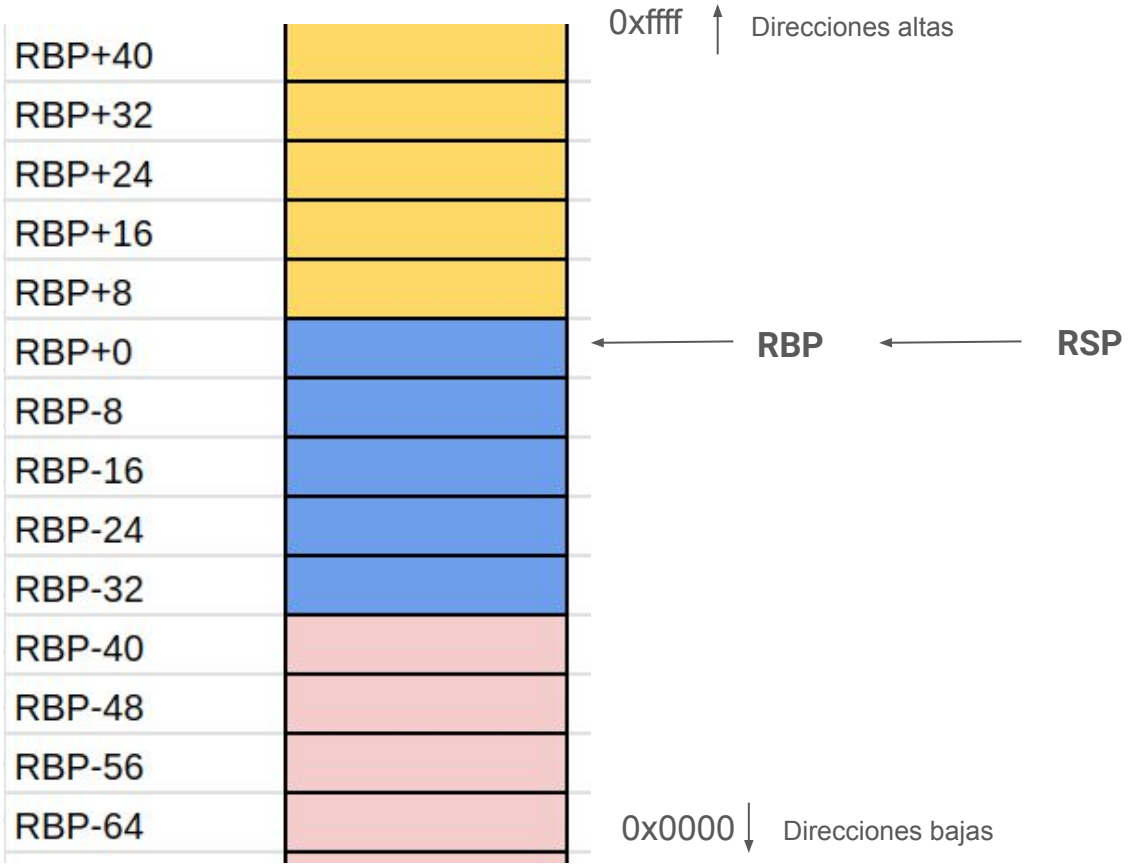
```
000000000000117e <main>:
    117e:       f3 0f 1e fa             endbr64
    1182:       55                      push   rbp
    1183:       48 89 e5                mov    rbp,rsp
    1186:       bf 04 00 00 00          mov    edi,0x4
    118b:       e8 b7 ff ff ff          call   1147 <func_a>
    1190:       b8 00 00 00 00          mov    eax,0x0
    1195:       5d                      pop    rbp
    1196:       c3                      ret
```

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Es lo opuesto a ENTER, que es que saca copia rbp a rsp, y empuja lo que esta en el stack a rbp (pop rbp)

PUSH

RAX = 0x77cc

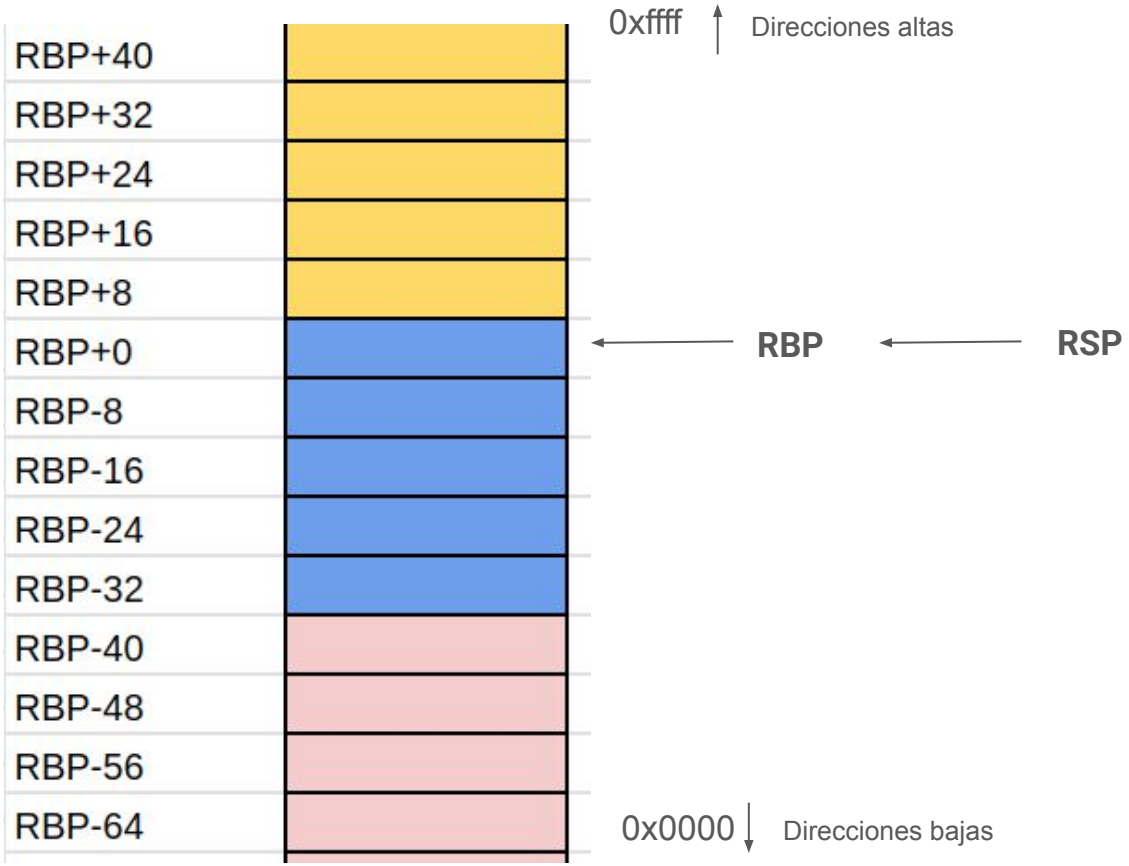


PUSH

RAX = 0x77cc

push rax

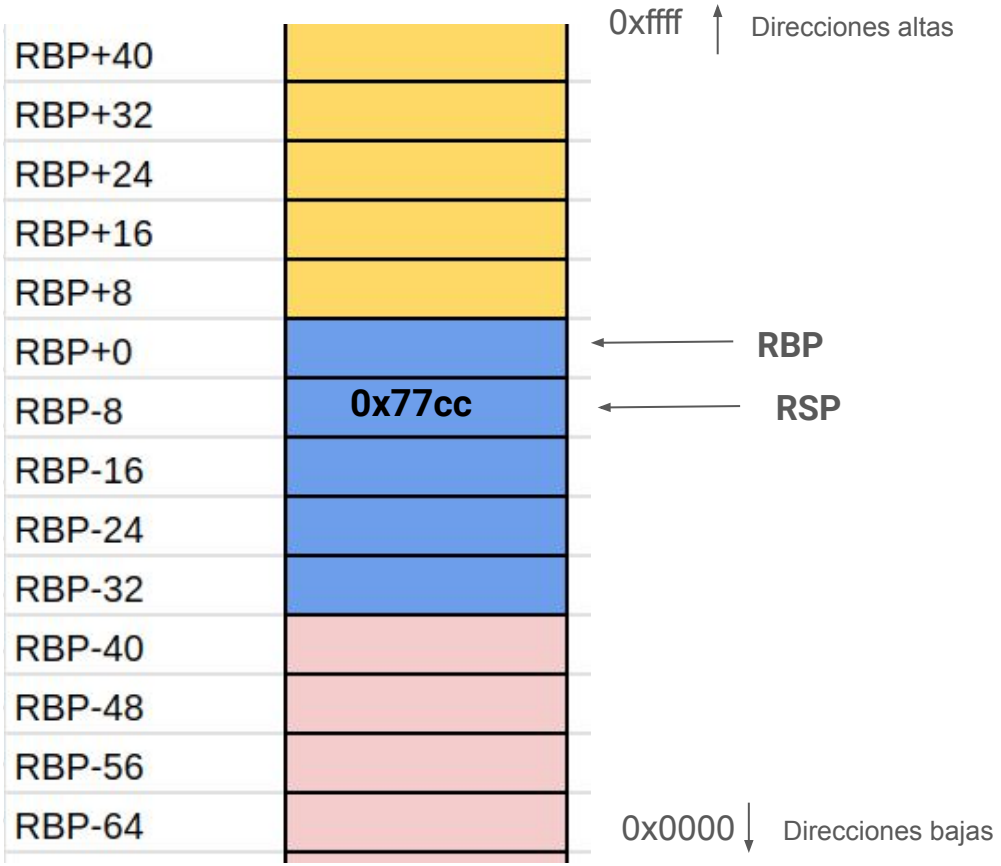
Hacer lugar para el contenido de rax en el stack, y mover el contenido de rax al stack



Push

RAX = 0x77cc

push rax

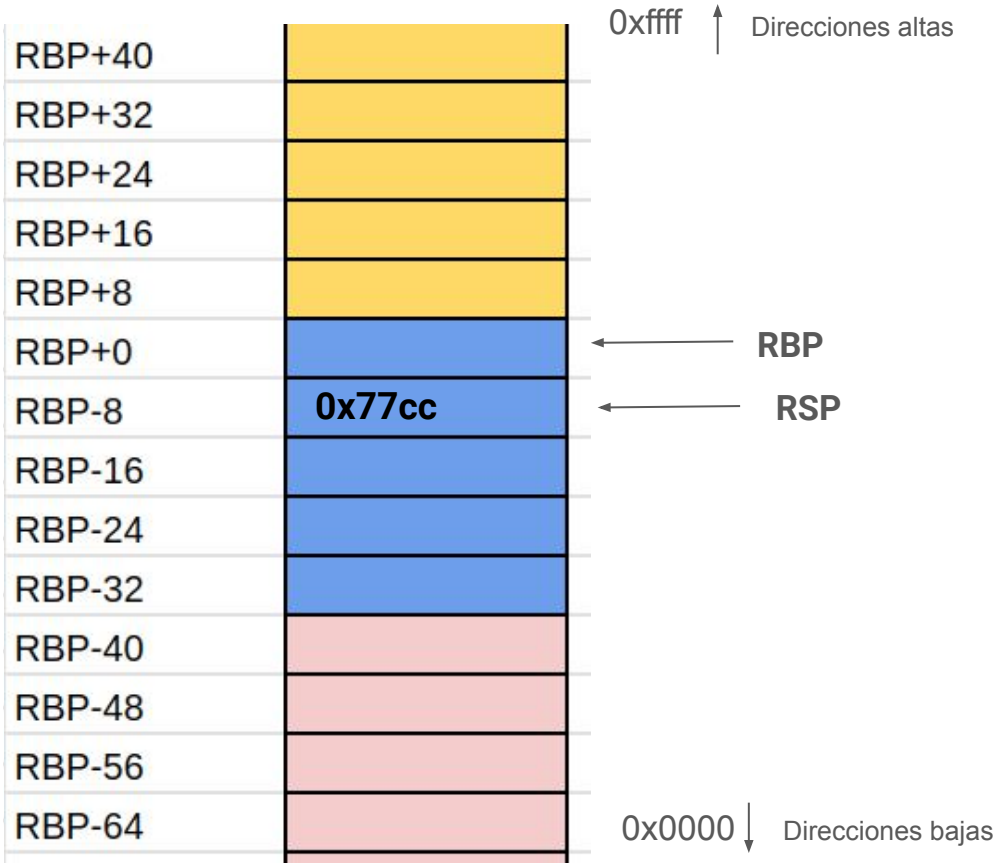


Pop

RAX = 0x77cc

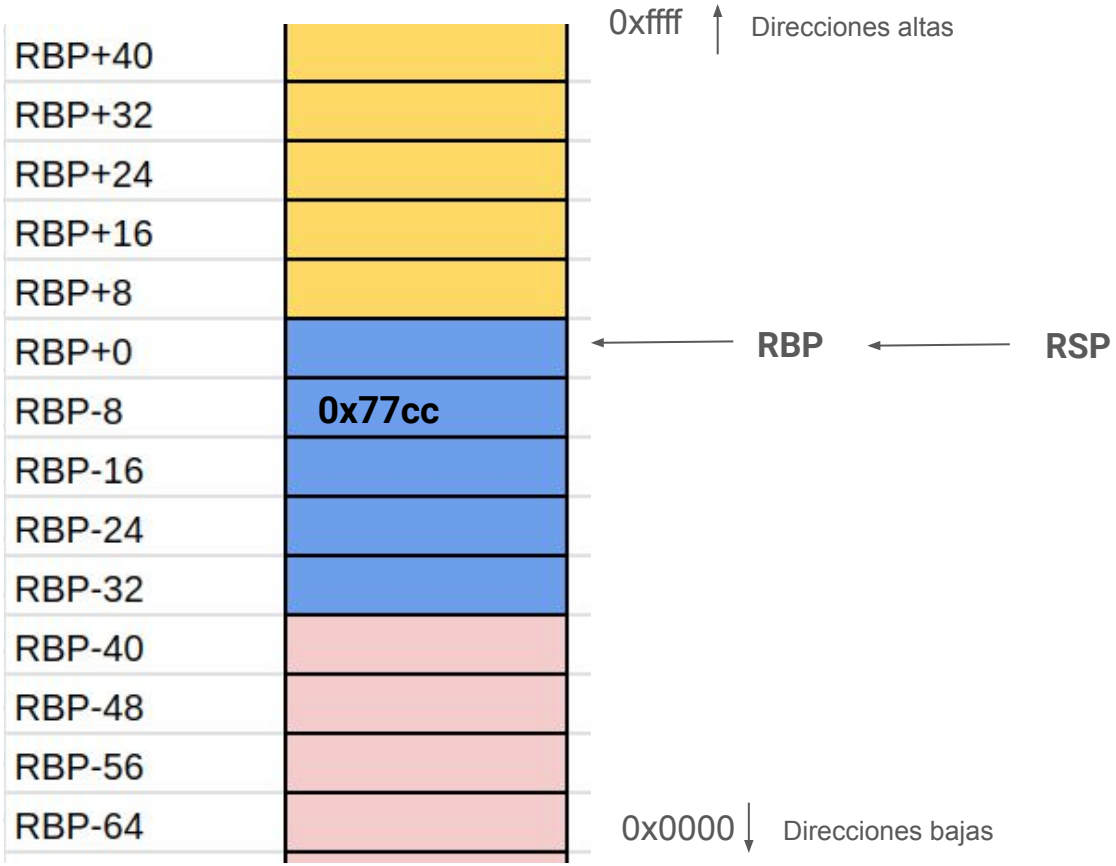
pop rdi

Mover a lo que está apuntando el stack, al lugar denominado por pop. Reducir el stack.



Pop

RAX = 0x77cc
RDI = 0x77cc



Convencion x86_64

Universidad de SanAndrés

Como se pasan los parámetros en el llamado de una función?

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Convencion x86_64

Universidad de SanAndrés

Como se pasan los parámetros en el llamado de una función?

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Cómo se retorna un valor?

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Parametros de integers:

- **%rdi** primer parametro
- **%rsi** segundo parametro
- **%rdx** tercer parametro
- **%rcx** cuarto parametro
- **%r8** quinto parametro
- **%r9** sexto parametro
- **%xmm0-7** 8 parametros de float

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Parametros de integers:

- **%rdi** primer parametro
- **%rsi** segundo parametro
- **%rdx** tercer parametro
- **%rcx** cuarto parametro
- **%r8** quinto parametro
- **%r9** sexto parametro
- **%xmm0-7** 8 parametros de float

Y si hay mas?

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Parametros de integers:

- **%rdi** primer parametro
- **%rsi** segundo parametro
- **%rdx** tercer parametro
- **%rcx** cuarto parametro
- **%r8** quinto parametro
- **%r9** sexto parametro
- **%xmm0-7** 8 parametros de float

Por que %esi, %edi?

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Parametros de integers:

- **%rdi** primer parametro
- **%rsi** segundo parametro
- **%rdx** tercer parametro
- **%rcx** cuarto parametro
- **%r8** quinto parametro
- **%r9** sexto parametro
- **%xmm0-7** 8 parametros de float

Son ints de 4 bytes

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

Como sabe donde volver después del call?

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

CALL <label>: hace un push de **%rip** al stack y hace un jump incondicional a la direccion del label. **%rip** contiene la dirección de la instrucción justo después del call.

Convencion x86_64

```
0000000000001147 <func_a>:
    1147:       f3 0f 1e fa             endbr64
    114b:       55                      push   rbp
    114c:       48 89 e5                mov    rbp,rsp
    114f:       48 83 ec 18             sub    rsp,0x18
    1153:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    1156:       c7 45 fc 07 00 00 00    mov    DWORD PTR [rbp-0x4],0x7
    115d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
    1160:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1163:       01 d0                   add    eax,edx
    1165:       83 e0 01                and    eax,0x1
    1168:       85 c0                   test   eax,eax
    116a:       74 0f                   je     117b <func_a+0x34>
    116c:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    116f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    1172:       89 d6                   mov    esi,edx
    1174:       89 c7                   mov    edi,eax
    1176:       e8 ae ff ff ff          call   1129 <func_b>
    117b:       90                      nop
    117c:       c9                      leave
    117d:       c3                      ret
```

CALL <label>: hace un push de **%rip** al stack y hace un jump incondicional a la direccion del label. **%rip** contiene la dirección de la instrucción justo después del call.

ret: hace un pop de **%rsp** a **%rip.**

Convencion x86_64

La convencion que estamos mostrando es:
System V AMD 64 ABI (linux)
Hay otras
(https://en.wikipedia.org/wiki/X86_calling_conventions) :

- Microsoft x86 calling conventions
- X32 ABI o cdecl (Linux)

Convencion x86_64

La convencion que estamos mostrando es:
System V AMD 64 ABI (linux)
Hay otras
(https://en.wikipedia.org/wiki/X86_calling_conventions) :

- Microsoft x86 calling conventions
- X32 ABI o cdecl (Linux)

Por qué es importante seguir la convención?

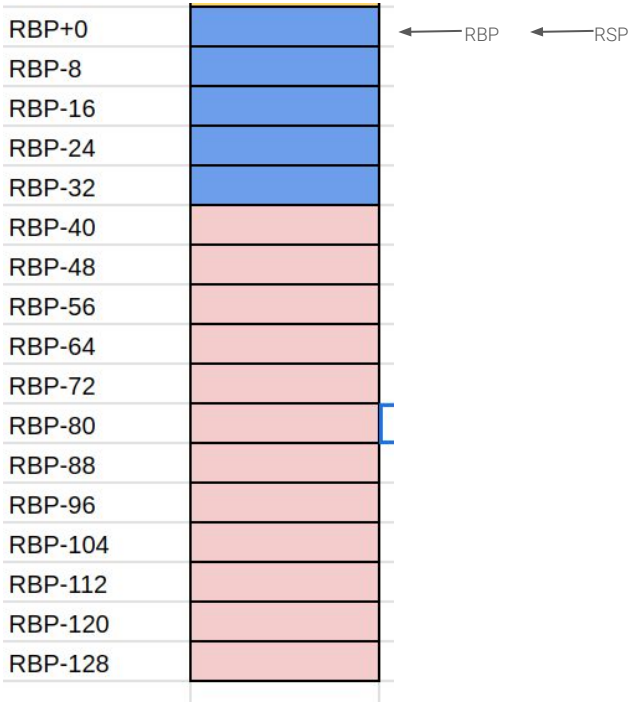
Convencion x86_64

Ejemplo completo:

Objdump -M intel -d simple

Ejemplo

STACK



0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51ab	inst pointer						
rsp	fff..1000	stack pointer	**rax**		return	**rsi**		2ndo arg
rbp	fff..1000	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

STACK

RBP+0		← RBP
RBP-8	fff...1000	← RSP
RBP-16		
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		
RBP-128		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51ac	inst pointer						
rsp	fff..0ff8	stack pointer	**rax**		return	**rsi**		2ndo arg
rbp	fff..0000	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

STACK

RBP+0	
RBP+0	fff...1000
RBP-8	
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

RBP ← RSP

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51af	inst pointer						
rsp	fff..0ff8	stack pointer	**rax**		return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+0	
RBP+0	fff...1000
RBP-8	
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

← RBP

← RSP

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51b3	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**		return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+0	
RBP+0	fff...1000
RBP-8	Ptr fs:0x28
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

← RBP

← RSP

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51c0	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	Ptr fs:0x28	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+0	
RBP+0	fff...1000
RBP-8	Ptr fs:0x28
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

← RBP

← RSP

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51c0	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	Ptr fs:0x28	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16		← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51c2	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51c9	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51cd	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0xff…0fe8	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**		3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51d0	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0xff…0fe8	return	**rsi**		2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51d5	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**		1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	← RSP
RBP-24		
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51da	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+0	
RBP+0	fff...1000 ← RBP
RBP-8	Ptr fs:0x28
RBP-16	0x0000 ← RSP
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

Push %RIP
Jump a Div

rip	555…51da	inst pointer						
rsp	fff..0fe8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+0	
RBP+0	fff...1000
RBP-8	Ptr fs:0x28
RBP-16	0x0000
RBP-24	0x555…51df
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5178	inst pointer						
rsp	fff..0fe0	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+0		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	
RBP-24	0x555…51df	
RBP-32	fff….0ff8	← RSP
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5179	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	
RBP-8	
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RSP ← RBP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…517c	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	
RBP-8	
RBP-16	
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5180	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	
RBP-8	
RBP-16	0x4 |
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5183	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5186	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>        endbr64
0x555555555178 <div+4>      push   rbp
0x555555555179 <div+5>      mov    rbp,rsp
0x55555555517c <div+8>      sub    rsp,0x20
0x555555555180 <div+12>     mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>     mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>     mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>     mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>     mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>     cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>     jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>     mov    eax,0xffffffff
0x55555555519e <div+42>     jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>     mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>     cdq
0x5555555551a4 <div+48>     idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>     mov    edx,eax
0x5555555551a9 <div+53>     mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>     mov    DWORD PTR [rax],edx
0x5555555551af <div+59>     mov    eax,0x0
0x5555555551b4 <div+64>     call   0x555555555169 <another>
0x5555555551b9 <div+69>     mov    eax,0x0
0x5555555551be <div+74>     leave
0x5555555551bf <div+75>     ret
```

rip	555…518a	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5191	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5194	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

Flags stay 0,

rip	555…5194	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…5197	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51a0	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51a3	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x4	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0xff…0fe8	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>        endbr64
0x555555555178 <div+4>      push   rbp
0x555555555179 <div+5>      mov    rbp,rsp
0x55555555517c <div+8>      sub    rsp,0x20
0x555555555180 <div+12>     mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>     mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>     mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>     mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>     mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>     cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>     jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>     mov    eax,0xffffffff
0x55555555519e <div+42>     jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>     mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>     cdq
0x5555555551a4 <div+48>     idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>     mov    edx,eax
0x5555555551a9 <div+53>     mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>     mov    DWORD PTR [rax],edx
0x5555555551af <div+59>     mov    eax,0x0
0x5555555551b4 <div+64>     call   0x555555555169 <another>
0x5555555551b9 <div+69>     mov    eax,0x0
0x5555555551be <div+74>     leave
0x5555555551bf <div+75>     ret
```

https://www.felixcloutier.com/x86/idiv

rip	555…51a4	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x4	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51a7	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0	3er arg

Ejemplo

STACK

RBP+44		
RBP+40	fff...1000	
RBP+32	Ptr fs:0x28	
RBP+24	0x0000	
RBP+16	0x555…51df	
RBP+8	fff….0ff8	← RBP
RBP+0	0x0 |	
RBP-8		
RBP-16	0x4 | 0x8	
RBP-24	0xff…0fe8	← RSP
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51a9	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	| 0x0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51ad	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	|0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51af	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	|0xff…0fe8	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

Considerar no op (sin operacion)

rip	555…51b4	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44		
RBP+40	fff...1000	
RBP+32	Ptr fs:0x28	
RBP+24	0x0000	
RBP+16	0x555…51df	
RBP+8	fff….0ff8	← RBP
RBP+0	0x0 |	
RBP-8		
RBP-16	0x4 | 0x8	
RBP-24	0xff…0fe8	← RSP
RBP-32		
RBP-40		
RBP-48		
RBP-56		
RBP-64		
RBP-72		
RBP-80		

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

rip	555…51b9	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

Leave:
mov rsp, rbp
pop rbp

rip	555…51be	inst pointer						
rsp	fff..0fb8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+44	
RBP+40	fff...1000
RBP+32	Ptr fs:0x28
RBP+24	0x0000
RBP+16	0x555…51df
RBP+8	fff….0ff8
RBP+0	0x0 |
RBP-8	
RBP-16	0x4 | 0x8
RBP-24	0xff…0fe8
RBP-32	
RBP-40	
RBP-48	
RBP-56	
RBP-64	
RBP-72	
RBP-80	

RBP ← RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>        endbr64
0x555555555178 <div+4>      push   rbp
0x555555555179 <div+5>      mov    rbp,rsp
0x55555555517c <div+8>      sub    rsp,0x20
0x555555555180 <div+12>     mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>     mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>     mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>     mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>     mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>     cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>     jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>     mov    eax,0xffffffff
0x55555555519e <div+42>     jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>     mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>     cdq
0x5555555551a4 <div+48>     idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>     mov    edx,eax
0x5555555551a9 <div+53>     mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>     mov    DWORD PTR [rax],edx
0x5555555551af <div+59>     mov    eax,0x0
0x5555555551b4 <div+64>     call   0x555555555169 <another>
0x5555555551b9 <div+69>     mov    eax,0x0
0x5555555551be <div+74>     leave
0x5555555551bf <div+75>     ret
```

Leave:
mov rsp, rbp
pop rbp

rip	555…51bf	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0fd8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44		
RBP+0	fff...1000	← RBP
RBP-8	Ptr fs:0x28	
RBP-16	0x0000	
RBP-24	0x555…51df	← RSP
RBP-32	fff….0ff8	
RBP-40	0x0 |	
RBP-48		
RBP-56	0x4 | 0x8	
RBP-64	0xff…0fe8	
RBP-72		
RBP-80		
RBP-88		
RBP-96		
RBP-104		
RBP-112		
RBP-120		

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>          endbr64
0x555555555178 <div+4>        push   rbp
0x555555555179 <div+5>        mov    rbp,rsp
0x55555555517c <div+8>        sub    rsp,0x20
0x555555555180 <div+12>       mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>       mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>       mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>       mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>       mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>       cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>       jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>       mov    eax,0xffffffff
0x55555555519e <div+42>       jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>       mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>       cdq
0x5555555551a4 <div+48>       idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>       mov    edx,eax
0x5555555551a9 <div+53>       mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>       mov    DWORD PTR [rax],edx
0x5555555551af <div+59>       mov    eax,0x0
0x5555555551b4 <div+64>       call   0x555555555169 <another>
0x5555555551b9 <div+69>       mov    eax,0x0
0x5555555551be <div+74>       leave
0x5555555551bf <div+75>       ret
```

Leave:
mov rsp, rbp
pop rbp

rip	555…51bf	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

STACK

RBP+44	
RBP+0	fff...1000
RBP-8	Ptr fs:0x28
RBP-16	0x0000
RBP-24	0x555…51df
RBP-32	fff….0ff8
RBP-40	0x0 |
RBP-48	
RBP-56	0x4 | 0x8
RBP-64	0xff…0fe8
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x555555555174 <div>        endbr64
0x555555555178 <div+4>      push   rbp
0x555555555179 <div+5>      mov    rbp,rsp
0x55555555517c <div+8>      sub    rsp,0x20
0x555555555180 <div+12>     mov    DWORD PTR [rbp-0x14],edi
0x555555555183 <div+15>     mov    DWORD PTR [rbp-0x18],esi
0x555555555186 <div+18>     mov    QWORD PTR [rbp-0x20],rdx
0x55555555518a <div+22>     mov    DWORD PTR [rbp-0x4],0x0
0x555555555191 <div+29>     mov    eax,DWORD PTR [rbp-0x18]
0x555555555194 <div+32>     cmp    eax,DWORD PTR [rbp-0x4]
0x555555555197 <div+35>     jg     0x5555555551a0 <div+44>
0x555555555199 <div+37>     mov    eax,0xffffffff
0x55555555519e <div+42>     jmp    0x5555555551be <div+74>
0x5555555551a0 <div+44>     mov    eax,DWORD PTR [rbp-0x14]
0x5555555551a3 <div+47>     cdq
0x5555555551a4 <div+48>     idiv   DWORD PTR [rbp-0x18]
0x5555555551a7 <div+51>     mov    edx,eax
0x5555555551a9 <div+53>     mov    rax,QWORD PTR [rbp-0x20]
0x5555555551ad <div+57>     mov    DWORD PTR [rax],edx
0x5555555551af <div+59>     mov    eax,0x0
0x5555555551b4 <div+64>     call   0x555555555169 <another>
0x5555555551b9 <div+69>     mov    eax,0x0
0x5555555551be <div+74>     leave
0x5555555551bf <div+75>     ret
```

ret:
Pop %rip

rip	555…51bf	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Ejemplo

Universidad de SanAndrés

STACK

RBP+44	
RBP+0	fff...1000
RBP-8	Ptr fs:0x28
RBP-16	0x0000
RBP-24	0x555…51df
RBP-32	fff….0ff8
RBP-40	0x0 |
RBP-48	
RBP-56	0x4 | 0x8
RBP-64	0xff…0fe8
RBP-72	
RBP-80	
RBP-88	
RBP-96	
RBP-104	
RBP-112	
RBP-120	

RBP

RSP

0x0000 ↓ Direcciones bajas

```
0x5555555551a7 <main>          endbr64
0x5555555551ab <main+4>        push   rbp
0x5555555551ac <main+5>        mov    rbp,rsp
0x5555555551af <main+8>        sub    rsp,0x10
0x5555555551b3 <main+12>       mov    rax,QWORD PTR fs:0x28
0x5555555551bc <main+21>       mov    QWORD PTR [rbp-0x8],rax
0x5555555551c0 <main+25>       xor    eax,eax
0x5555555551c2 <main+27>       mov    DWORD PTR [rbp-0x10],0x0
0x5555555551c9 <main+34>       lea    rax,[rbp-0x10]
0x5555555551cd <main+38>       mov    rdx,rax
0x5555555551d0 <main+41>       mov    esi,0x8
0x5555555551d5 <main+46>       mov    edi,0x4
0x5555555551da <main+51>       call   0x555555555169 <div>
0x5555555551df <main+56>       mov    DWORD PTR [rbp-0xc],eax
0x5555555551e2 <main+59>       cmp    DWORD PTR [rbp-0xc],0x0
0x5555555551e6 <main+63>       je     0x555555555201 <main+90>
0x5555555551e8 <main+65>       mov    eax,DWORD PTR [rbp-0x10]
0x5555555551eb <main+68>       mov    esi,eax
0x5555555551ed <main+70>       lea    rax,[rip+0xe10]        # 0x555555556004
0x5555555551f4 <main+77>       mov    rdi,rax
0x5555555551f7 <main+80>       mov    eax,0x0
0x5555555551fc <main+85>       call   0x555555555070 <printf@plt>
0x555555555201 <main+90>       mov    eax,0x0
0x555555555206 <main+95>       mov    rdx,QWORD PTR [rbp-0x8]
0x55555555520a <main+99>       sub    rdx,QWORD PTR fs:0x28
0x555555555213 <main+108>      je     0x55555555521a <main+115>
0x555555555215 <main+110>      call   0x555555555060 <__stack_chk_fail@plt>
0x55555555521a <main+115>      leave
```

rip	555…51df	inst pointer						
rsp	fff..0fd8	stack pointer	**rax**	0	return	**rsi**	0x8	2ndo arg
rbp	fff..0ff8	base pointer	**rdi**	0x4	1er arg	**rdx**	0x0	3er arg

Temas del Stack

- Stack Attack
- Dirección de retorno? (Return Address)
- Mas de 6 parametros?

Extra Slides

https://www.cs.virginia.edu/~evans/cs216/guides/x86.html