Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

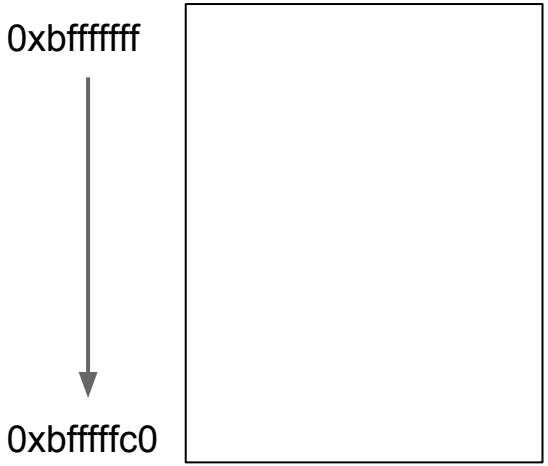
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ cat example.c
void func(int a) {
   int8_t b = 0;
   return;
}
int main(int argc, char** argv) {
   func(0);
   return 0;
}
```

```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

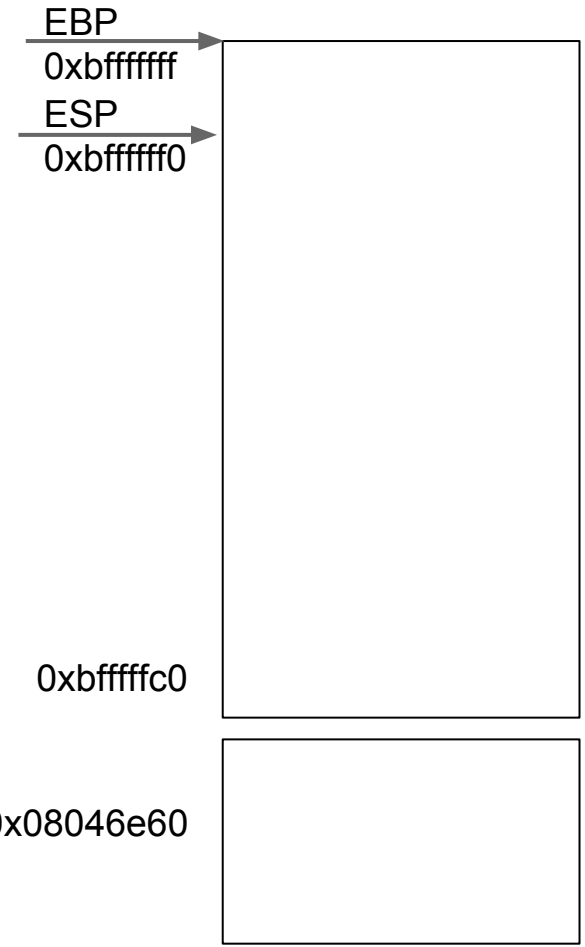
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

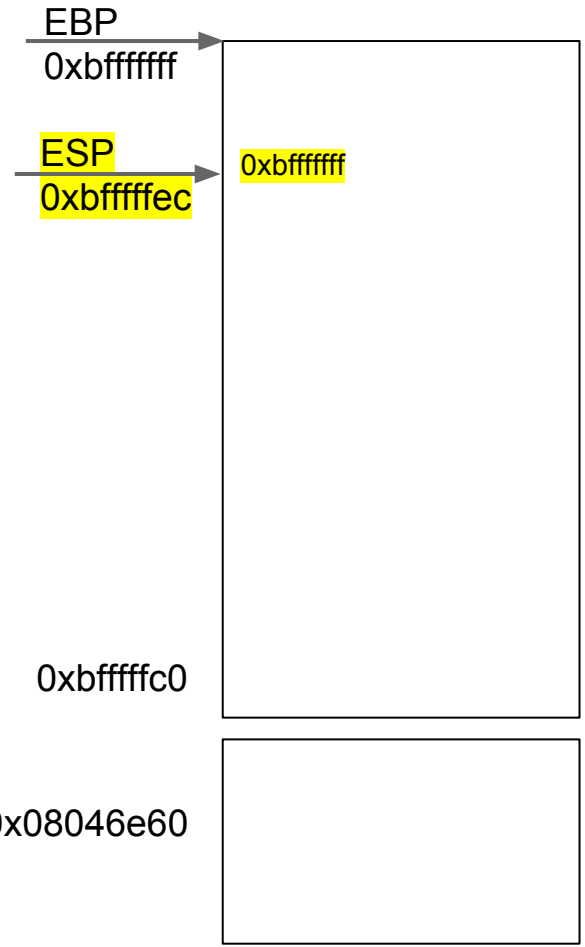
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

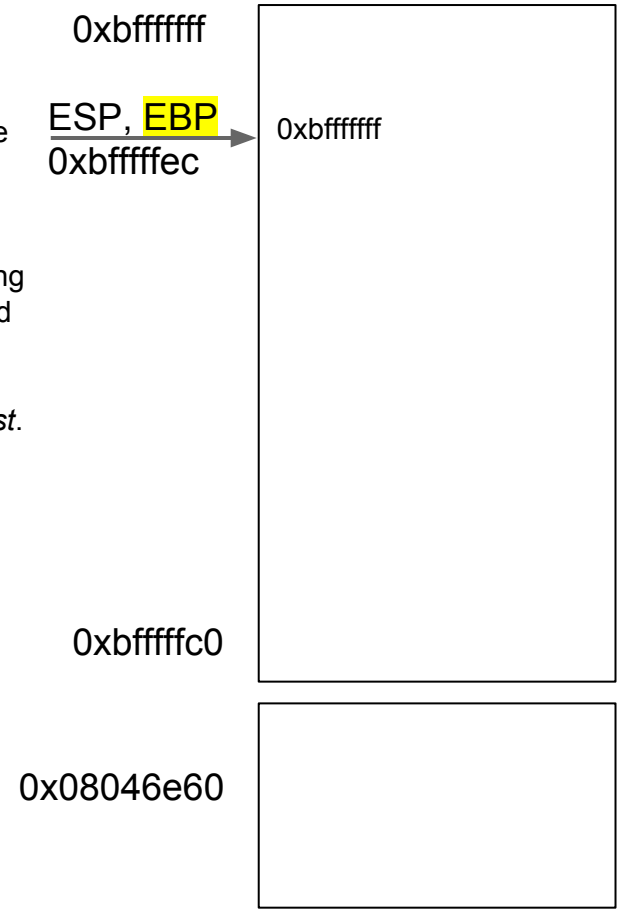
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

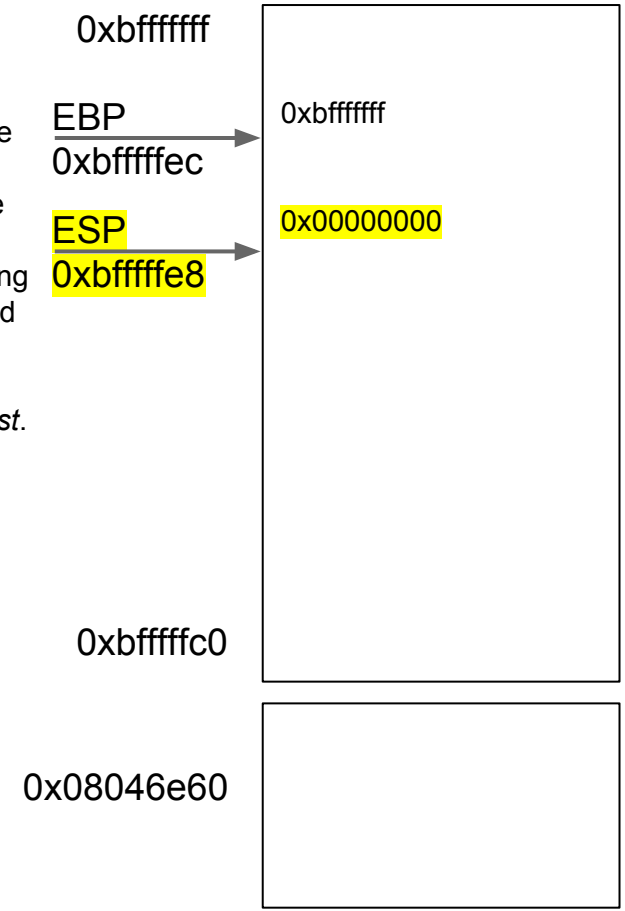
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

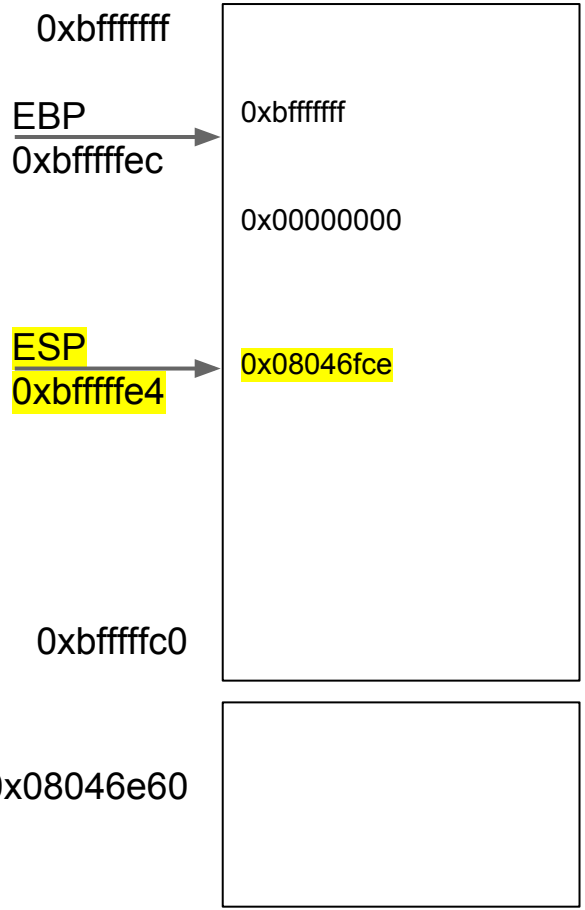
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

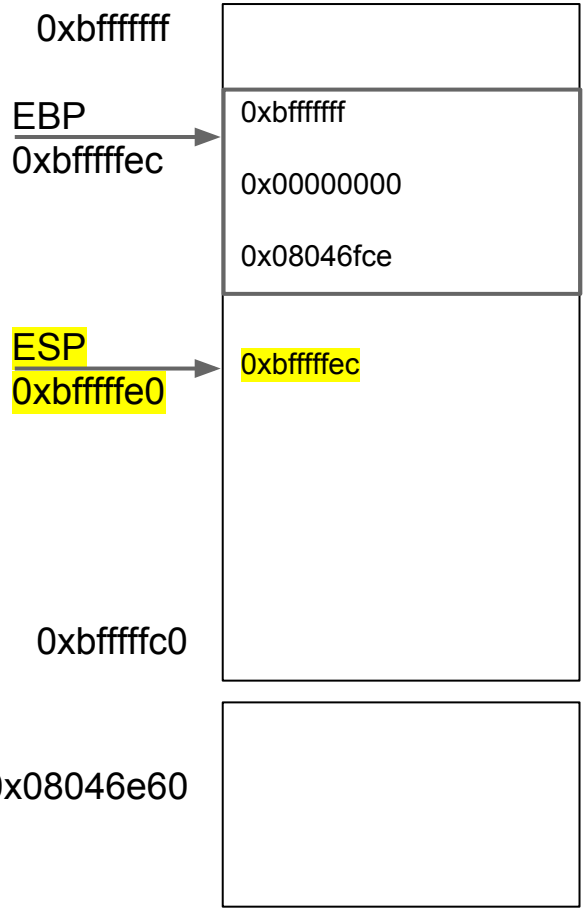
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

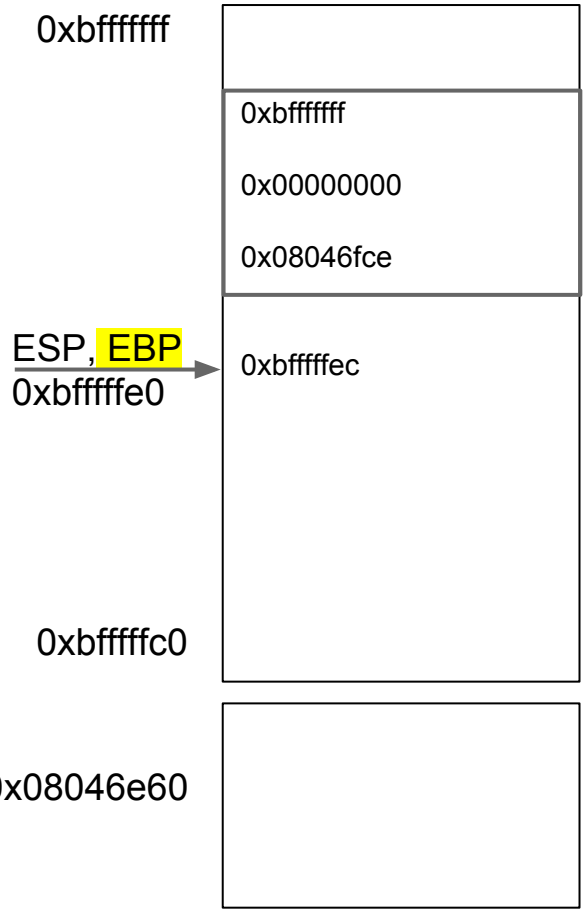
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

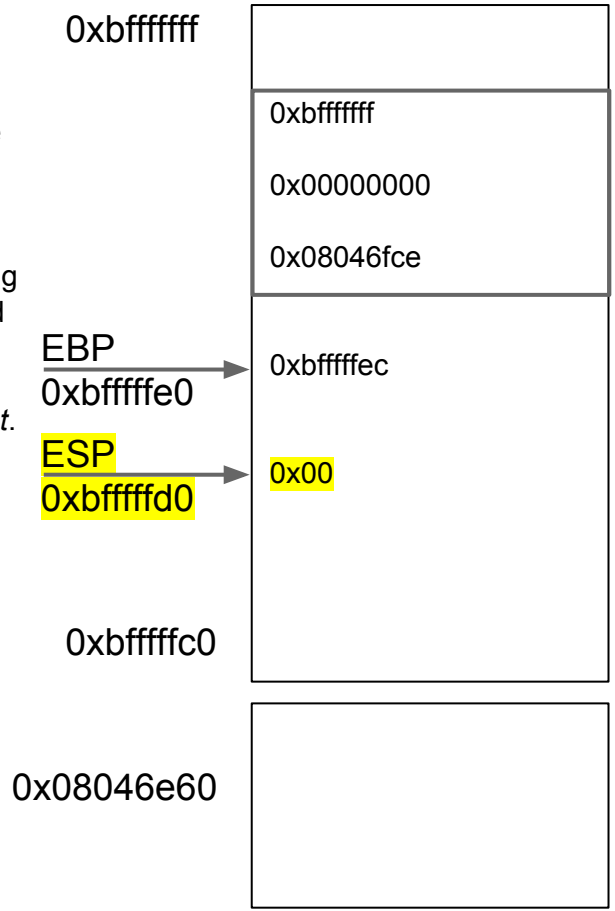
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

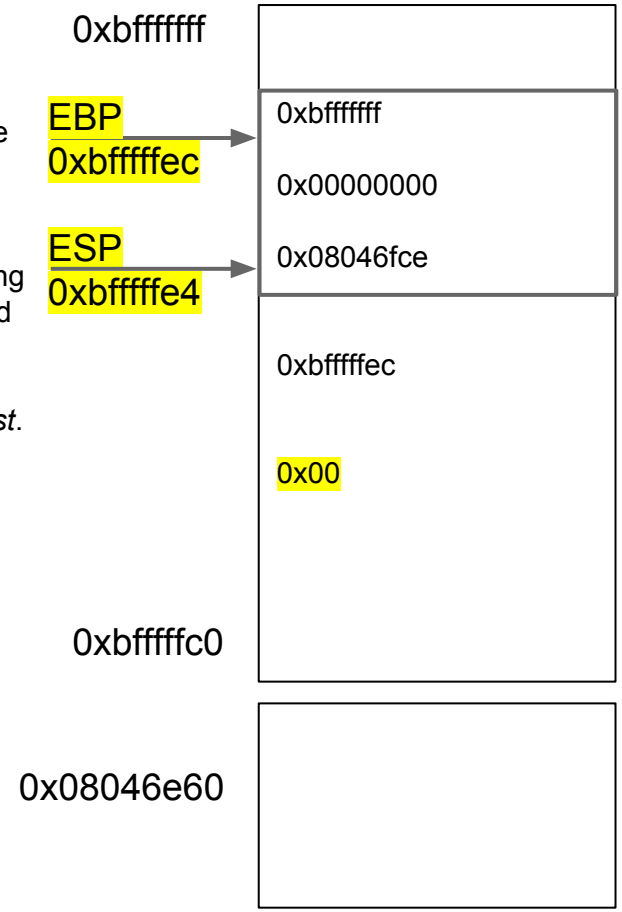
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

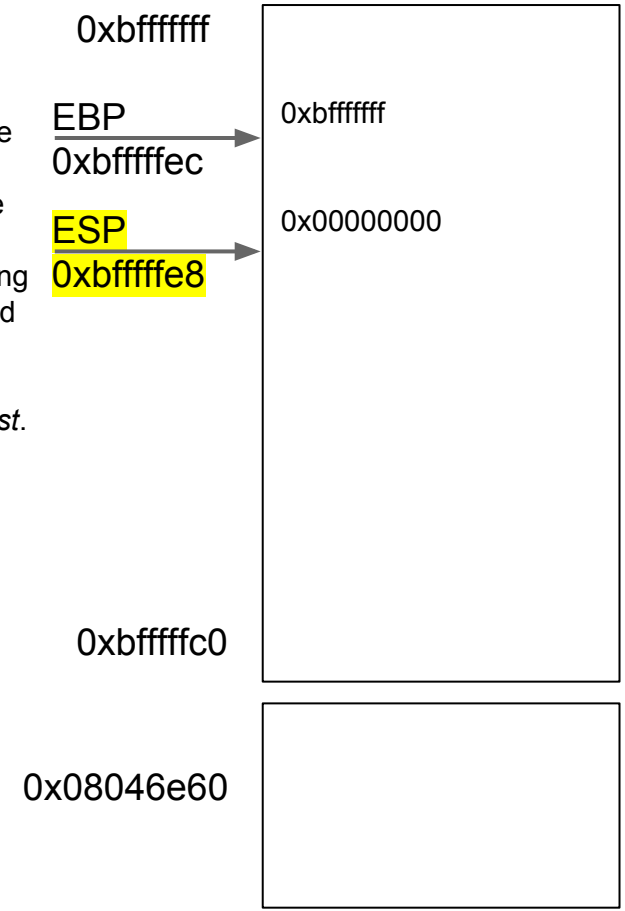
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```

Quick Guide to Assembly in 161

Registers

stack pointer (ESP): register containing the address of the top of the stack

base pointer (EBP): register containing the address of the bottom of the stack frame

instruction pointer (EIP): register containing the address of the instruction to be executed

Other examples: EAX (return value), etc.

Instructions

mov *dest*, *src*: copy 4 bytes from *src* to *dest*. For registers, move their value, and for addresses, move the data at the address. (Intel notation)

jmp *address*: execute the instruction at *address*

pop *reg* = mov *reg*, ESP; add ESP, 4

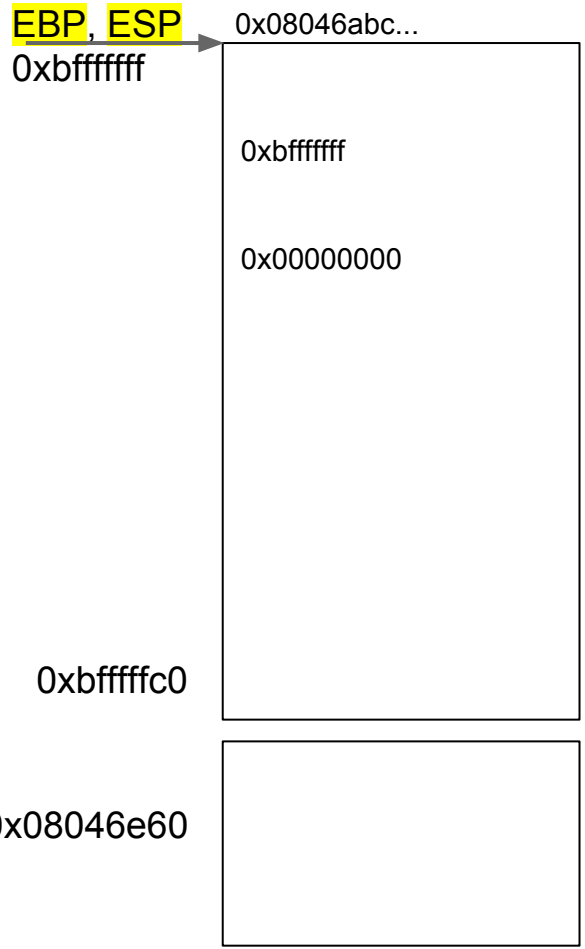
push *data* = sub ESP, 4; mov *data*, ESP

enter *N bytes*= push EBP; mov EBP, ESP; sub ESP, N bytes (required for locals)

call *func* = push EIP; jmp *func* address

leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP




```
$ objdump -d -M intel -S example.o
void func(int a) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x10
[int b = 0;]
mov    BYTE PTR [ebp-0x1],0x0
[return;]
leave                               epilogue
ret

int main(int argc, char ** argv) {
push   ebp                          prologue
mov    ebp,esp
sub    esp,0x4
[func(0);]
mov    DWORD PTR [esp],0x0
call   24 <main+0xe>
[return 0;]
mov    eax,0x0
leave                               epilogue
ret
```