

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*

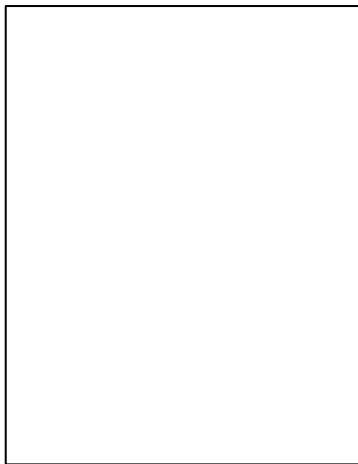
leave = mov ESP, EBP; pop EBP

ret = pop EIP; jmp EIP

0xbfffffff



0xbffffffc0



“top of the stack”

```
$ 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
mov   ebp,esp
sub   esp,0x10
[int b = 0;]
mov   BYTE PTR [ebp-0x1],0x0
[return;]
leave
ret
```

prologue

epilogue

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

prologue

epilogue

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

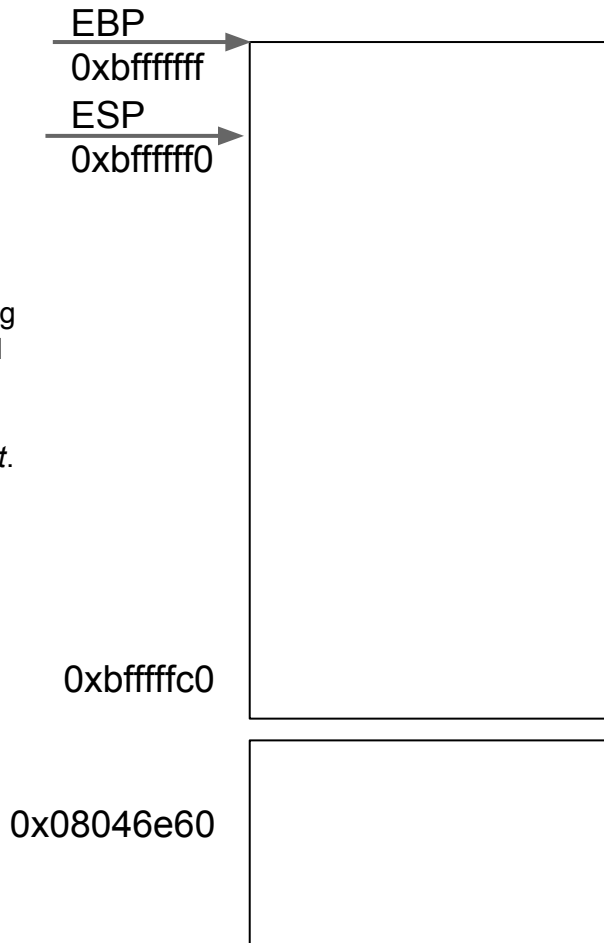
push *data* = sub ESP, 4; mov *data*, ESP

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  
    mov  ebp,esp  
    sub  esp,0x10  
    [int b = 0;]  
    mov  BYTE PTR [ebp-0x1],0x0  
    [return;]  
    leave  
    ret
```

Diagrammatic annotations for the first code block: A bracket on the right side of the first four lines (push ebp, mov ebp, esp, sub esp, 0x10, [int b = 0;]) is labeled 'prologue'. A bracket on the right side of the last two lines (leave, ret) is labeled 'epilogue'.

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

Diagrammatic annotations for the second code block: A bracket on the right side of the first four lines (push ebp, mov ebp, esp, sub esp, 0x4, [func(0);]) is labeled 'prologue'. A bracket on the right side of the last three lines (mov eax, 0x0, leave, ret) is labeled 'epilogue'.

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

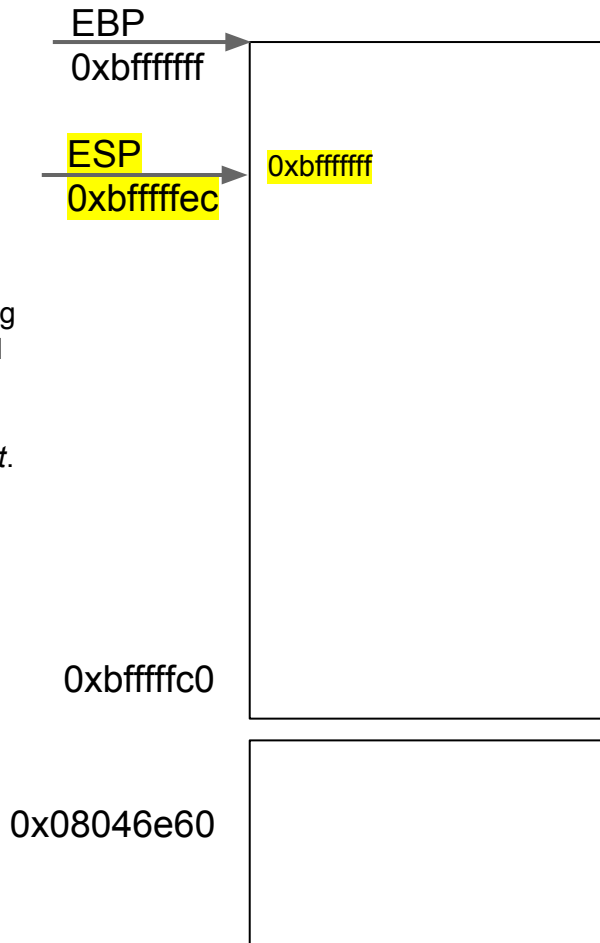
push *data* = sub ESP, 4; mov *data*, ESP

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
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

Diagrammatic annotations: A bracket on the right side of the first four lines is labeled "prologue". A bracket on the right side of the last two lines is labeled "epilogue".

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

Diagrammatic annotations: A bracket on the right side of the first four lines is labeled "prologue". A bracket on the right side of the last three lines is labeled "epilogue".

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  
  mov  ebp,esp  
  sub  esp,0x10  
  [int b = 0;]  
  mov  BYTE PTR [ebp-0x1],0x0  
  [return;]  
  leave  
  ret
```

prologue

epilogue

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

prologue

epilogue

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

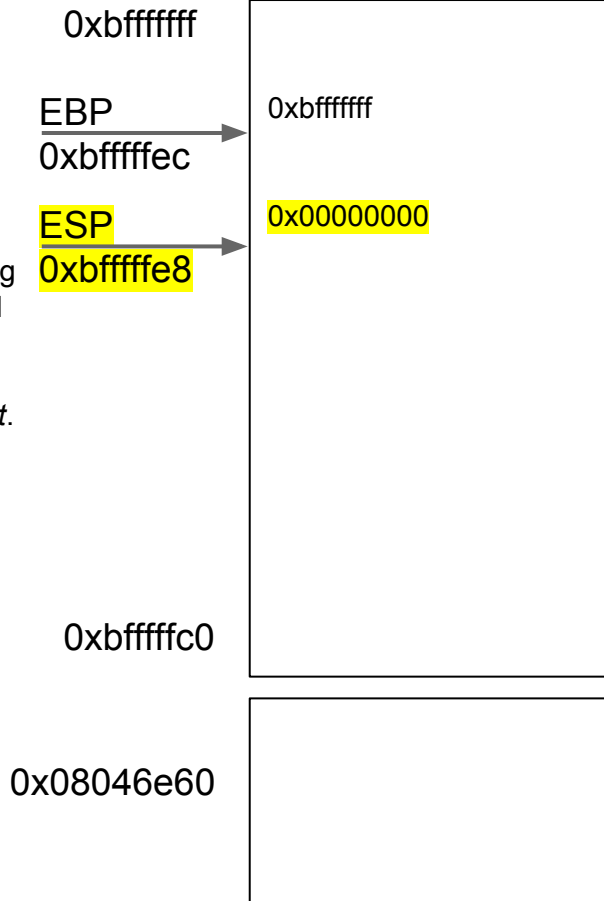
push *data* = sub ESP, 4; mov *data*, ESP

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  
  mov  ebp,esp  
  sub  esp,0x10  
  [int b = 0;]  
  mov  BYTE PTR [ebp-0x1],0x0  
  [return;]  
  leave  
  ret
```

prologue

epilogue

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

prologue

epilogue

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

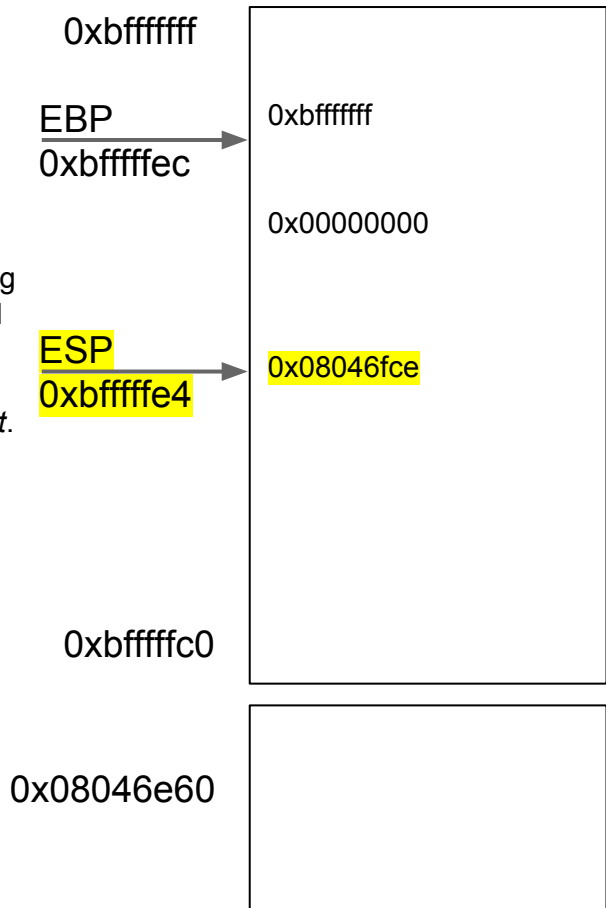
push *data* = sub ESP, 4; mov *data*, ESP

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  
    mov  ebp,esp  
    sub  esp,0x10  
    [int b = 0;]  
    mov  BYTE PTR [ebp-0x1],0x0  
    [return;]  
    leave  
    ret
```

prologue

epilogue

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

prologue

epilogue

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

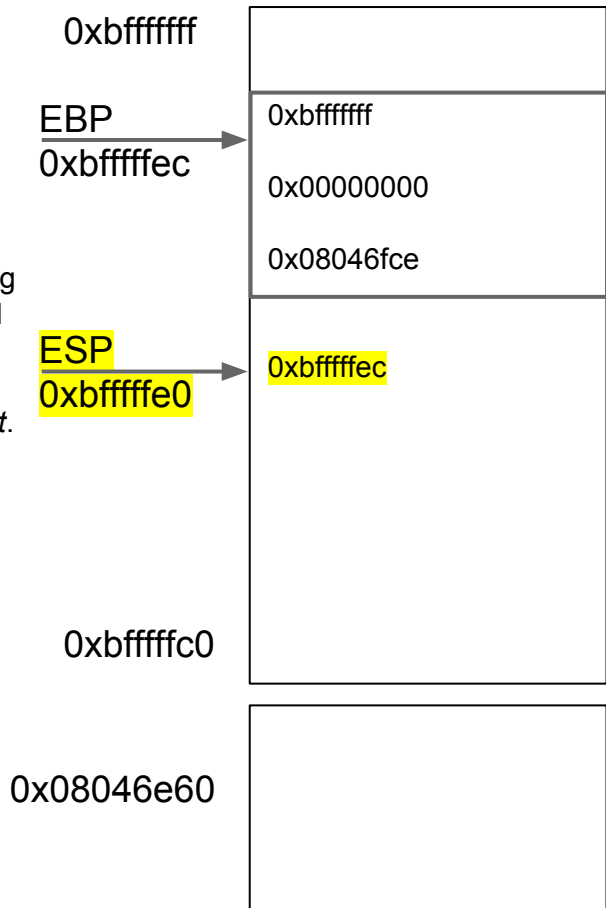
push *data* = sub ESP, 4; mov *data*, ESP

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
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

prologue

epilogue

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

prologue

epilogue

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

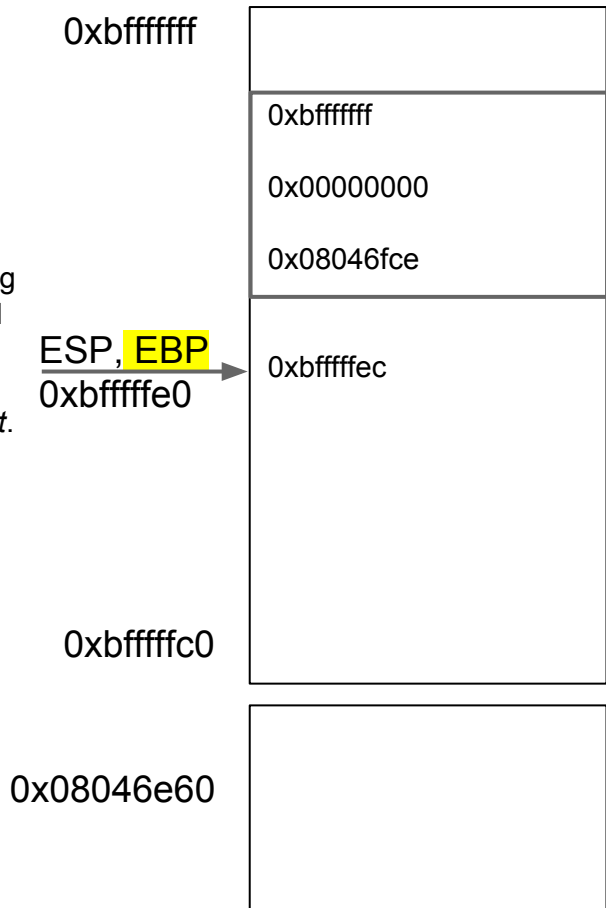
push *data* = sub ESP, 4; mov *data*, ESP

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
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

prologue

epilogue

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

prologue

epilogue

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

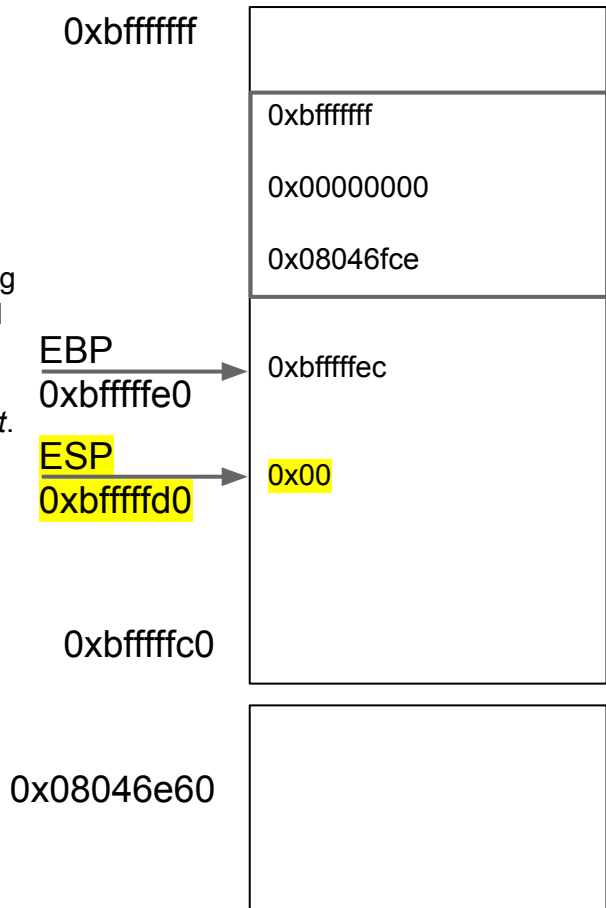
push *data* = sub ESP, 4; mov *data*, ESP

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
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

prologue

epilogue

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

prologue

epilogue

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

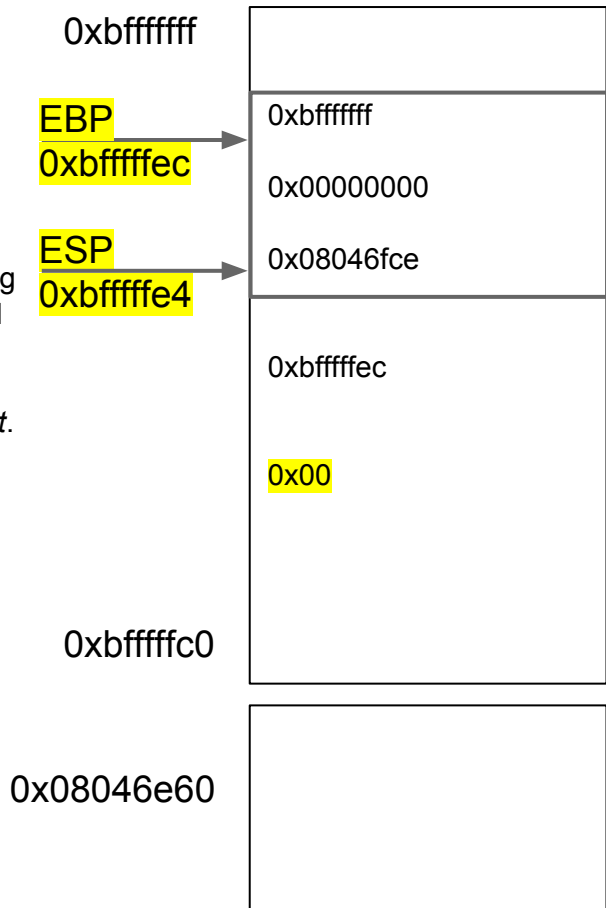
enter *N bytes* = push EBP; mov EBP, ESP;

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
mov ebp,esp
sub esp,0x10
[int b = 0;]
mov BYTE PTR [ebp-0x1],0x0
[return;]
leave
ret
}
```

prologue

epilogue

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

prologue

epilogue

epilogue

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

prologue

epilogue

epilogue

epilogue

epilogue

epilogue

epilogue

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

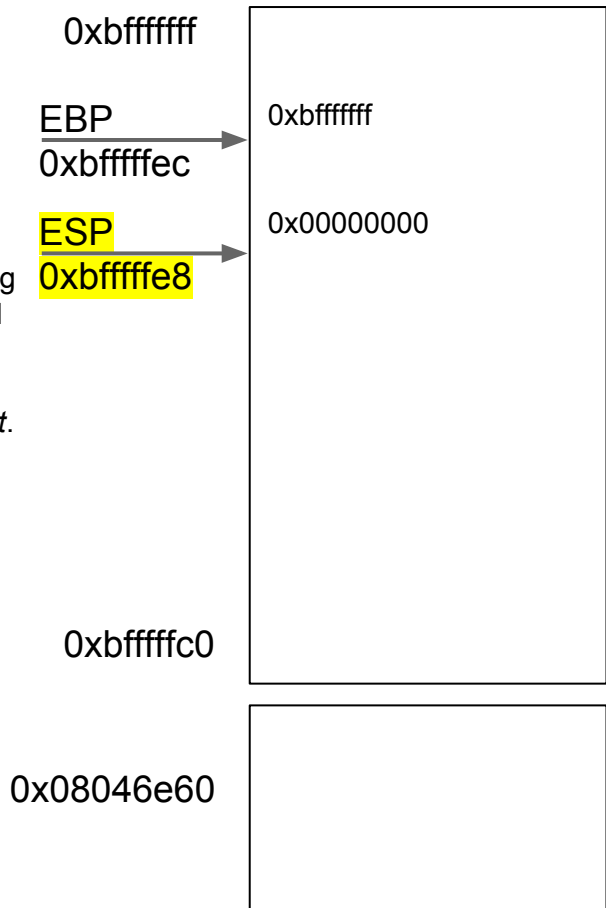
push *data* = sub ESP, 4; mov *data*, ESP

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
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

prologue

epilogue

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

prologue

epilogue

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

EBP, ESP → 0x08046abc...

0xbfffffff

0xbfffffff

0x00000000

0xbffffffc0

0x08046e60

```
$ objdump -d -M intel -S example.o
```

```
void func(int a) {
    push ebp
    mov  ebp,esp
    sub  esp,0x10
    [int b = 0;]
    mov  BYTE PTR [ebp-0x1],0x0
    [return;]
    leave
    ret
}
```

prologue

epilogue

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

prologue

epilogue