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.

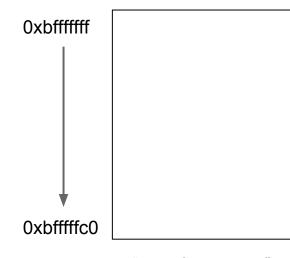
Instructions

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



"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
                                    prologue
mov
      ebp.esp
     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);]
      DWORD PTR [esp],0x0
call 24 <main+0xe>
[return 0:]
mov eax,0x0
leave
                                    epilogue
ret
```

EBP 0xbffffff \$ objdump -d -M intel -S example.o Quick Guide to Assembly in 161 void func(int a) { **ESP** Registers push ebp prologue 0xbfffff0 stack pointer (ESP): register containing the mov ebp.esp address of the top of the stack esp,0x10 base pointer (EBP): register containing the [int b = 0:] address of the bottom of the stack frame mov BYTE PTR [ebp-0x1],0x0 instruction pointer (EIP): register containing [return:] the address of the instruction to be executed epilogue leave Other examples: EAX (return value), etc. ret Instructions mov dest, src: copy 4 bytes from src to dest. int main(int argc, char ** argv) { For registers, move their value, and for push ebp prologue addresses, move the data at the address. mov ebp.esp (Intel notation) sub esp.0x4 imp address: execute the instruction at [func(0);] address DWORD PTR [esp],0x0 pop reg = mov reg, ESP; add ESP, 4 0xbfffffc0 call 24 <main+0xe> push data = sub ESP, 4; mov data, ESP [return 0:] enter N bytes= push EBP; mov EBP, ESP; mov eax,0x0 sub ESP, N bytes (required for locals) leave epilogue **call** *func* = push EIP; jmp *func* address 0x08046e60 ret leave = mov ESP, EBP; pop EBP ret = pop EIP; imp EIP

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 top of the stack
base pointer (ESP): 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

call func = push EIP; jmp func ad leave = mov ESP, EBP; pop EBP ret = pop EIP; jmp EIP 0xbfffffff

EBP

ESP 0xbfffffec

0xbfffffff

0xbfffffc0

0x08046e60

60

\$ 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 ret

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

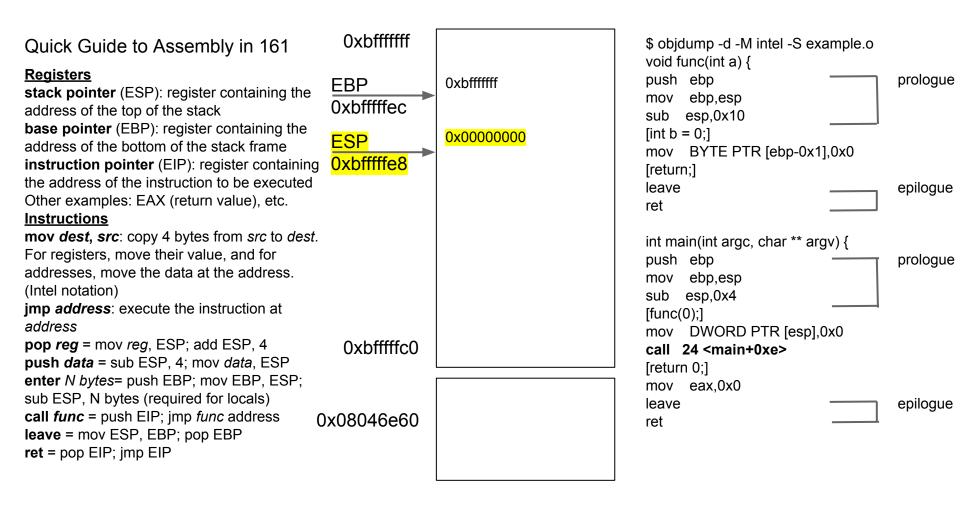
leave ret

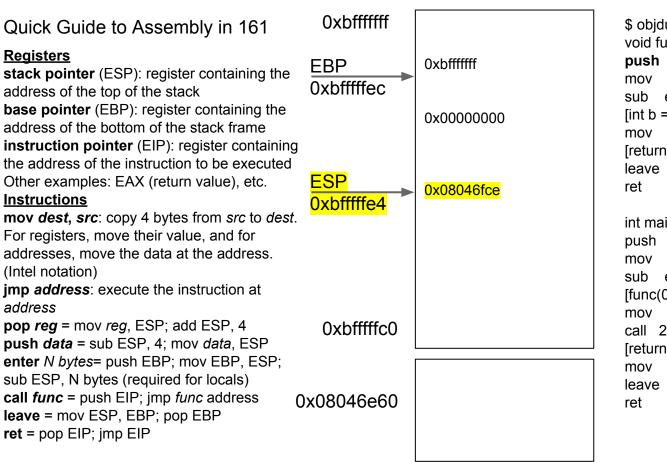
epilogue

epilogue

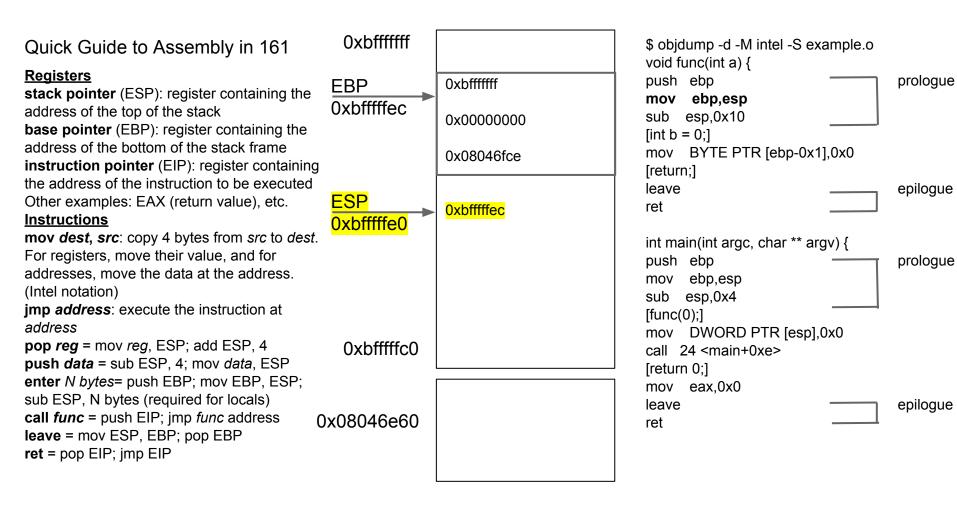
prologue

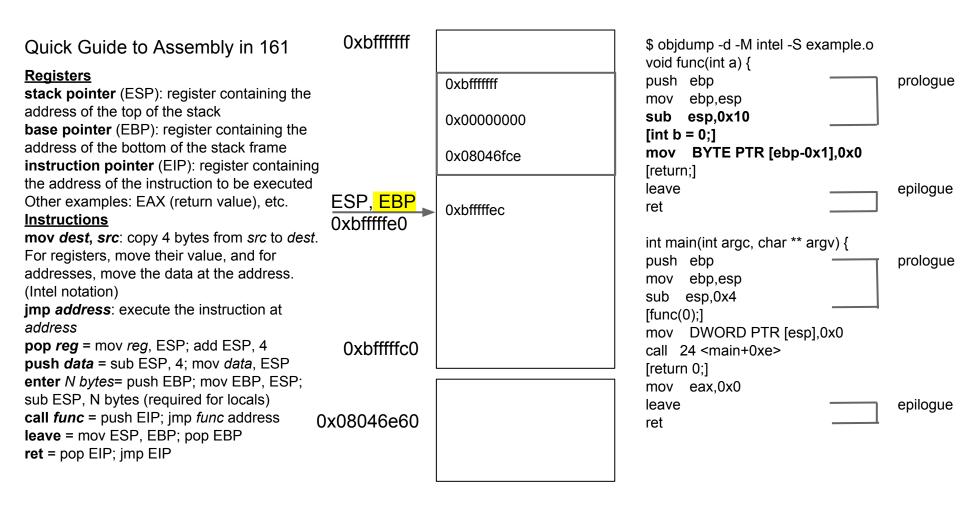
0xbffffff \$ objdump -d -M intel -S example.o Quick Guide to Assembly in 161 void func(int a) { Registers push ebp prologue ESP, EBP 0xbffffff stack pointer (ESP): register containing the mov ebp.esp 0xbfffffec address of the top of the stack esp,0x10 base pointer (EBP): register containing the [int b = 0:] address of the bottom of the stack frame mov BYTE PTR [ebp-0x1],0x0 instruction pointer (EIP): register containing [return:] the address of the instruction to be executed epilogue leave Other examples: EAX (return value), etc. ret Instructions mov dest, src: copy 4 bytes from src to dest. int main(int argc, char ** argv) { For registers, move their value, and for push ebp prologue addresses, move the data at the address. mov ebp.esp (Intel notation) sub esp.0x4 imp address: execute the instruction at [func(0);] address DWORD PTR [esp],0x0 pop reg = mov reg, ESP; add ESP, 4 0xbfffffc0 call 24 <main+0xe> push data = sub ESP, 4; mov data, ESP [return 0:] enter N bytes= push EBP; mov EBP, ESP; mov eax,0x0 sub ESP, N bytes (required for locals) leave epilogue **call** *func* = push EIP; jmp *func* address 0x08046e60 ret leave = mov ESP, EBP; pop EBP ret = pop EIP; imp EIP

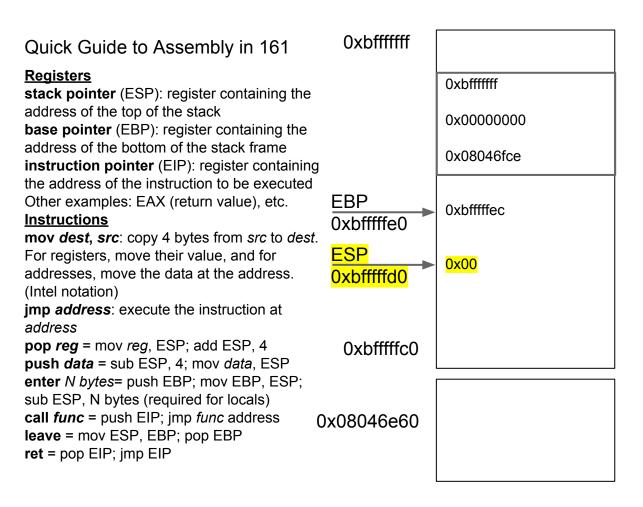




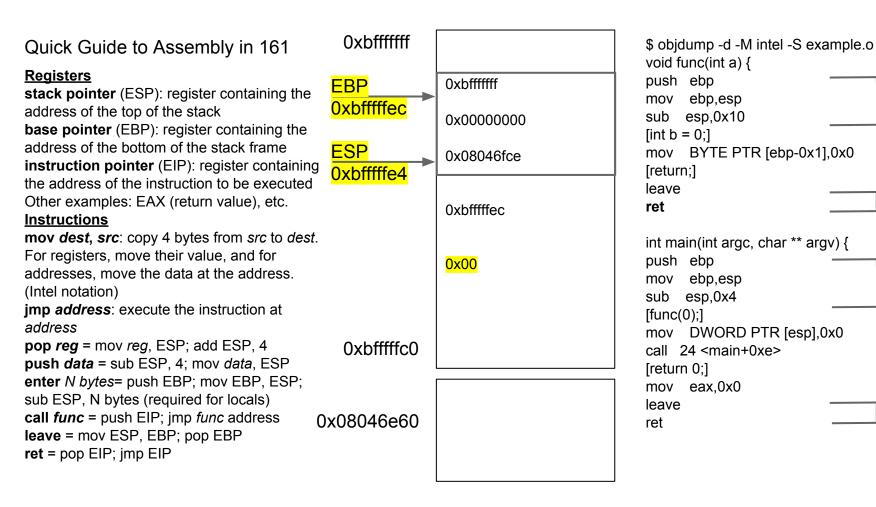
\$ objdump -d -M intel -S example.o void func(int a) { push ebp prologue ebp.esp esp,0x10 [int b = 0:] mov BYTE PTR [ebp-0x1],0x0 [return:] epilogue int main(int argc, char ** argv) { push ebp prologue ebp.esp sub esp.0x4 [func(0);] DWORD PTR [esp],0x0 call 24 <main+0xe> [return 0:] mov eax,0x0 epilogue







\$ objdump -d -M intel -S example.o void func(int a) { push ebp prologue mov ebp.esp esp,0x10 [int b = 0:] mov BYTE PTR [ebp-0x1],0x0 [return:] epilogue leave ret int main(int argc, char ** argv) { push ebp prologue mov ebp.esp sub esp.0x4 [func(0);] DWORD PTR [esp],0x0 call 24 <main+0xe> [return 0:] mov eax,0x0 leave epilogue ret

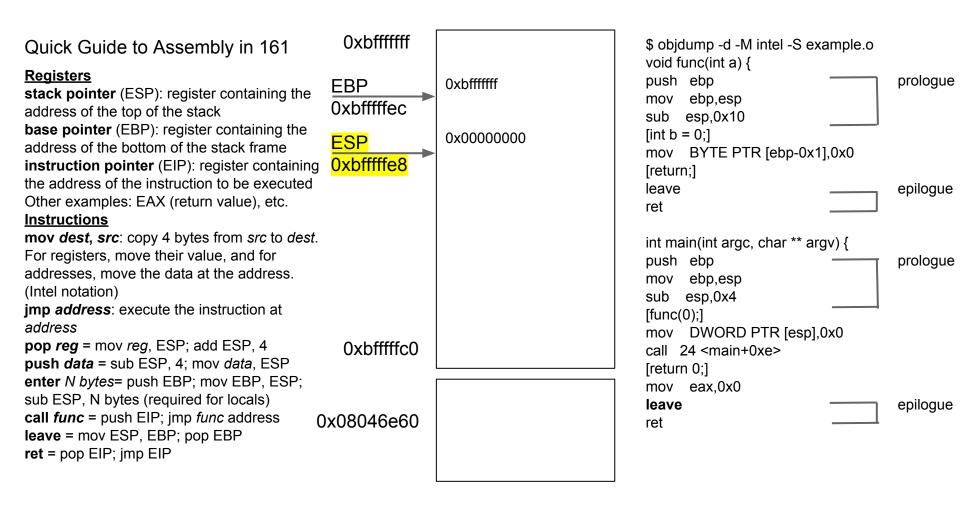


prologue

epilogue

prologue

epilogue



EBP. ESP 0x08046abc... 0xbfffffff Quick Guide to Assembly in 161 Registers 0xbfffffff stack pointer (ESP): register containing the address of the top of the stack base pointer (EBP): register containing the 0x00000000 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) imp address: execute the instruction at address pop reg = mov reg, ESP; add ESP, 4 0xbfffffc0 push data = sub ESP, 4; mov data, ESP

0x08046e60

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

sub ESP, N bytes (required for locals)

leave = mov ESP, EBP; pop EBP

ret = pop EIP; imp EIP

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

void func(int a) { push ebp prologue mov ebp.esp sub esp.0x10 [int b = 0:] mov BYTE PTR [ebp-0x1],0x0 [return:] epilogue leave ret int main(int argc, char ** argv) { push ebp prologue mov ebp.esp sub esp.0x4 [func(0);] DWORD PTR [esp],0x0 call 24 <main+0xe> [return 0:] mov eax,0x0 leave epilogue ret

\$ obidump -d -M intel -S example.o