Assembly programming

First step

Quick registers tour

General Purpose Registers (GPRs)

- EAX
- EBX
- ECX
- EDX
- ESI
- EDI
- ESP
- EBP

Segment Registers

- CS
- DS
- SS
- ES
- FS
- GS

Instruction Pointer

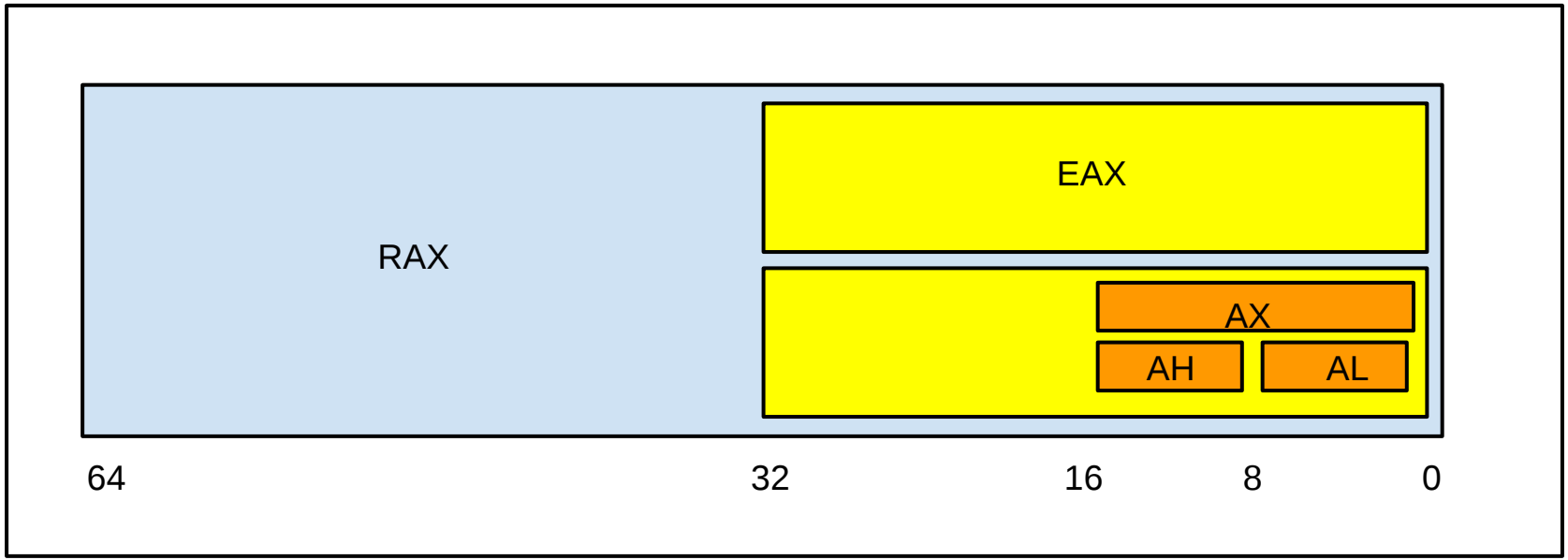
- EIP

Most GPRs use to have a specific usage:

- EAX: accumulator
- ECX: counter
- ESI: source index
- EDI: destination index

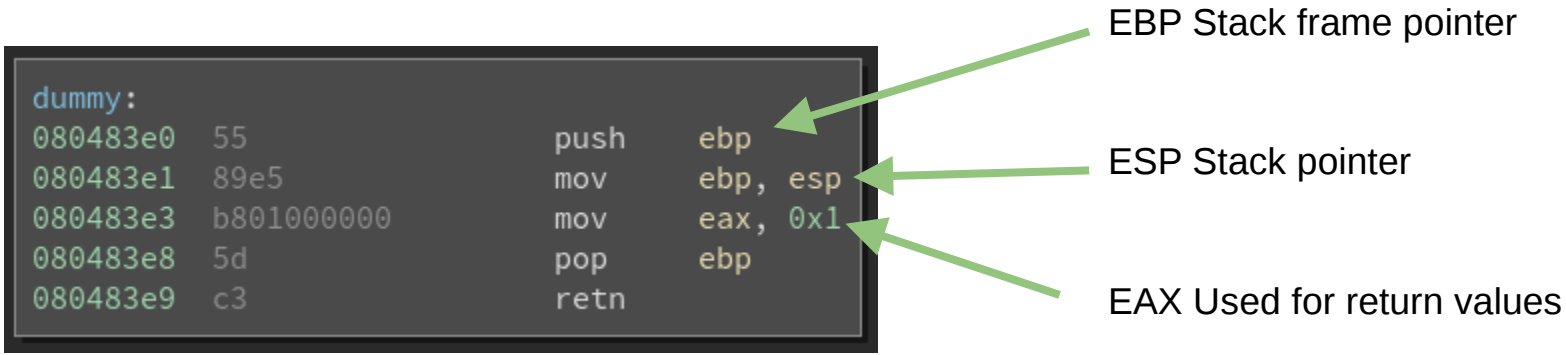
This is still the case!

Registers are further divided

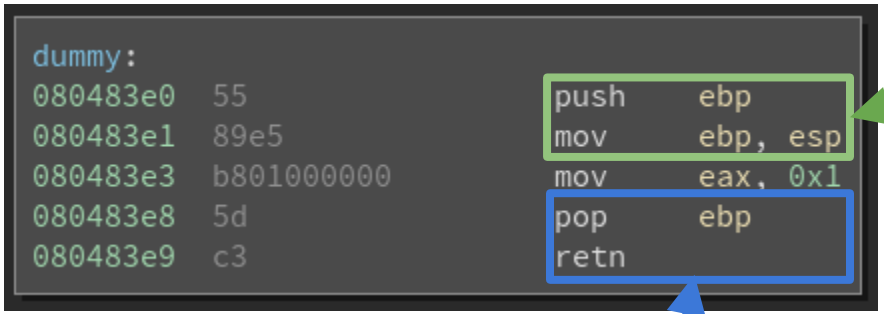


RAX = 64 bits, EAX = 32 bits, AX = 16 bits, AH = 8bits, AL = 8 bits.
We will be using this later.

EAX / EBP / ESP - What do you need to remember



Function Prolog and Epilog



Stack frame “setup”

We will explain the stack later. For now, you can simply assume
functions should “always” start with these two lines.

Stack frame “tear down” and return

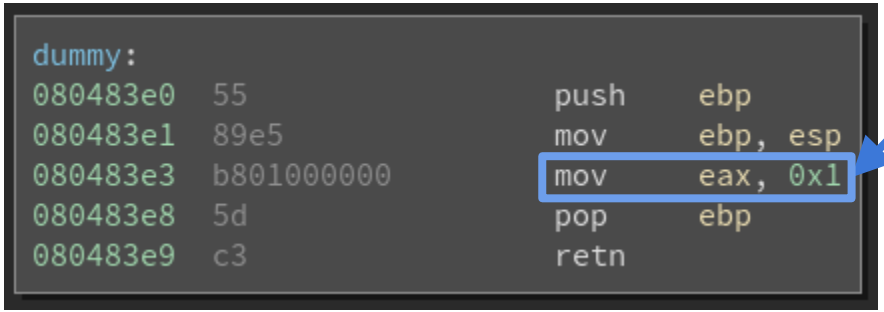
Function Return Value

```
dummy:
080483e0  55              push    ebp
080483e1  89e5            mov     ebp, esp
080483e3  b801000000      mov     eax, 0x1
080483e8  5d              pop     ebp
080483e9  c3              retn
```

EAX is to be used for function return value. This is defined by the ABI used by Linux.

Both Windows and Linux uses EAX for function return values.

MOV instruction



MOV is possibly the most frequently used instruction.

It uses 2 operands (destination and source) and can copy data to and from registers.

The only real limitation here is that only one of the two operand can be dereferenced from memory (we'll talk about memory later)

General form goes like:
MOV destination, source

MOV eax, 0x1
results in value 0x1 being copied into register eax

Register should be seen as "variables"

In order to validate your setup, we will now write, build, and run a function that returns the integer “1”.