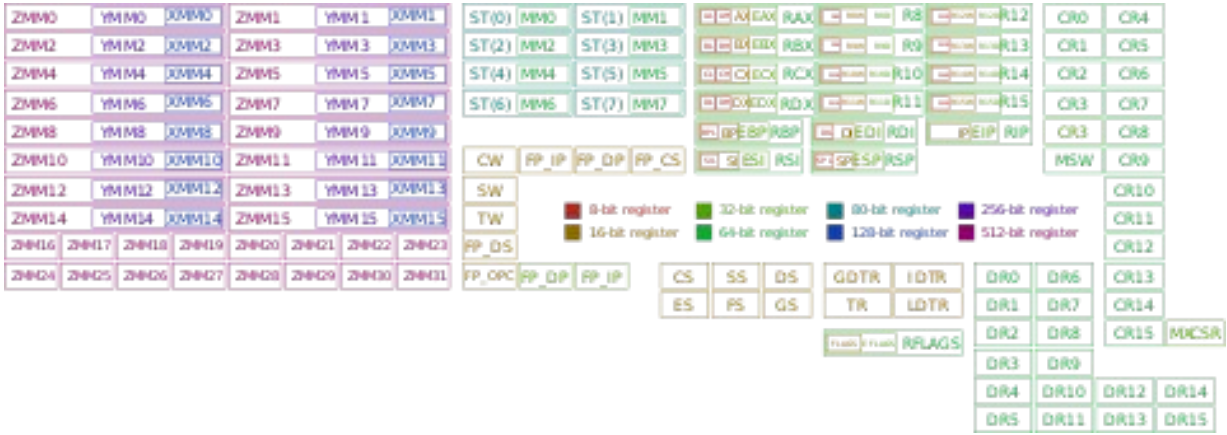
Intel 8086 in the modern era

Luis Alvernaz

x86 registers

In almost every CPU, we have what's called registers. These are here to store data in quick access locations, store arguments for some select functions, act as commonly used variables and for various math operations.

Most of them we don't care about though



The ones we do care about

rax/eax: In x86 the wax register is called the accumulator. Although none of these registers actually have to follow conventions (often in malware to confuse researchers), eax stores values we want to add to or a return value for a function.

rbx/ebx: ebx is the second commonly used general purpose register, however, nobody really cares about it's convention, so just see it as another register to use.

rcx/ecx: The ecx register is often called the counter register. By compiler convention, we usually see it act as the I value in a for(I =0; I< n, I++).

rdx/edx is again, just another general purpose register.

The ones we do care about (extended)

rbp/ebp: This register has the same purpose almost all of the time for stack based programs. It stores the position of the top of the stack. (The place you aren't pushing stuff in)

rsp/esp: The second of the two stack registers, esp holds the address of the bottom of the stack (in x86, where you push stuff in).

rip/eip: eip is the instruction pointer. It stores the address of the next instruction in memory the CPU is gonna execute.

rsi/esi/rdi/edi: these are the source and destination registers. Sometimes they point at memory a loop might right to or act as variables for functions that might not use the stack for it.

Basic instructions 1

mov: mov moves data from it's second argument to the first one.. however there are some restrictions… (on board)

lea: Stands for load effective address. It may also be used for simple additions using a trick: when a number is written as [0x1234] it means the value at 0x1234. A lea eax [0x1234 + 0x5678] will store the address of the value at 0x1234+0x5678 in eax, which is 0x1234+0x5678.

push: puts a value at the address of esp+4

pop: takes a value from the address esp and puts it in a specified register.

Basic instructions 2

add, sub, mul, div: Adds, subtracts, multiplies or divides the given register by a given value.

Inc/dec: Increment or decrement a given register.

xor/and/or/neg: preform the said logical operations on a register or value.

cmp/test: compare two given values.

jmp/j[cond]: jump stuff

EXPLOIT TIME