15.12.2023

Machine instruction representations

Internal format of an instruction

[prefixes] + code + [Mode R/M] + [SIB] + [displacement] + [immediate]

code: 1 or 2 bytes

Mode R/M: 0 or 1 bytes — registers/memory

SIB: SCALE INDEX BASE — 0 or 1 byte

displacement: 0, 1, 2, 4 bytes

immediate: 0, 1, 2, 4 bytes

OBS: All are optional, but the "code" field.

Mode R/M

Mod	Reg/Op code	R/M
7 6	5 4 3	2 1 0

SIB

Scale	Index	Base
7 6	5 4 3	2 1 0

Table 2-2 Mode R/M

ADC	Eb	Gb	(10)
	Ev	Gv	(11)
	Gb	Eb	(12)
	Gv	Ev	(13)

5Dh = 10011101

Mod   Reg. Mem.   R/M

[EBP]

first operand

memory addressed op.
composed of reg + 32 bit disp.

EBX

second operand

⇒ instruction [EBP] + disp_of 32 bits, EBX

2Ah = 00101010

Mod   Reg op.code   R/M

EBP   EDX

instr [EDX], EBP (or CH, or BP) memory register

F3h analyze at home

Table 2-3 for SIB byte

Scale Index Base

9Ch = 10 011 100

Scale of 4 EBX*4 ESP

00 → scale 1
01 → scale 2
10 → scale 4
11 → scale 8

=> offset = [ESP + EBX*4]

SIB byte will generate the offset formula

Mod RM byte îmi zice doar care registru sau var. din memorie

=> ne uităm la SIB byte

Jump instruction analysis.

Mecanand far jumps

You can perform a far jump only by using a pointer variables (on 6 bytes)

2 bytes segment
4 bytes offset.

CS: EIP currently execut instruction

you can only change the value of EIP by jumping

or CS:EIP by far jumping.

```
segment date
    aici dd here
segment code
        mov eax, [aici]
        mov ebx, aici
        jmp here
        jmp eax
        jmp [aici]

        jmp [ebx]

here : ....
```

$\underbrace{[offset(here)]}_{aici}$

jmp here, jmp eax, jmp [aici] } direct addressing.

jmp [ebx] indirect addressing

$\underbrace{[offset(aici)]}_{EBX}$