2.6.5. Reprezentarea instrucţiunilor maşină

O instrucţiune maşină x86 reprezintă o secvenţă de 1 până la 15 octeţi, care prin valorile lor specifică o operaţie de executat, operanzii asupra cărora va fi aplicată, precum şi modificatori suplimentari care controlează modul în care aceasta va fi executată.

O instrucţiune maşină x86 are maximum doi operanzi. Pentru cele mai multe dintre instrucţiuni, cei doi operanzi poartă numele de *sursă*, respectiv *destinaţie*. Dintre cei doi operanzi, maximum unul se poate afla în memoria RAM. Celălalt se află fie într-un registru al **EU**, fie este o constantă întreagă. Astfel, o instrucţiune are forma:

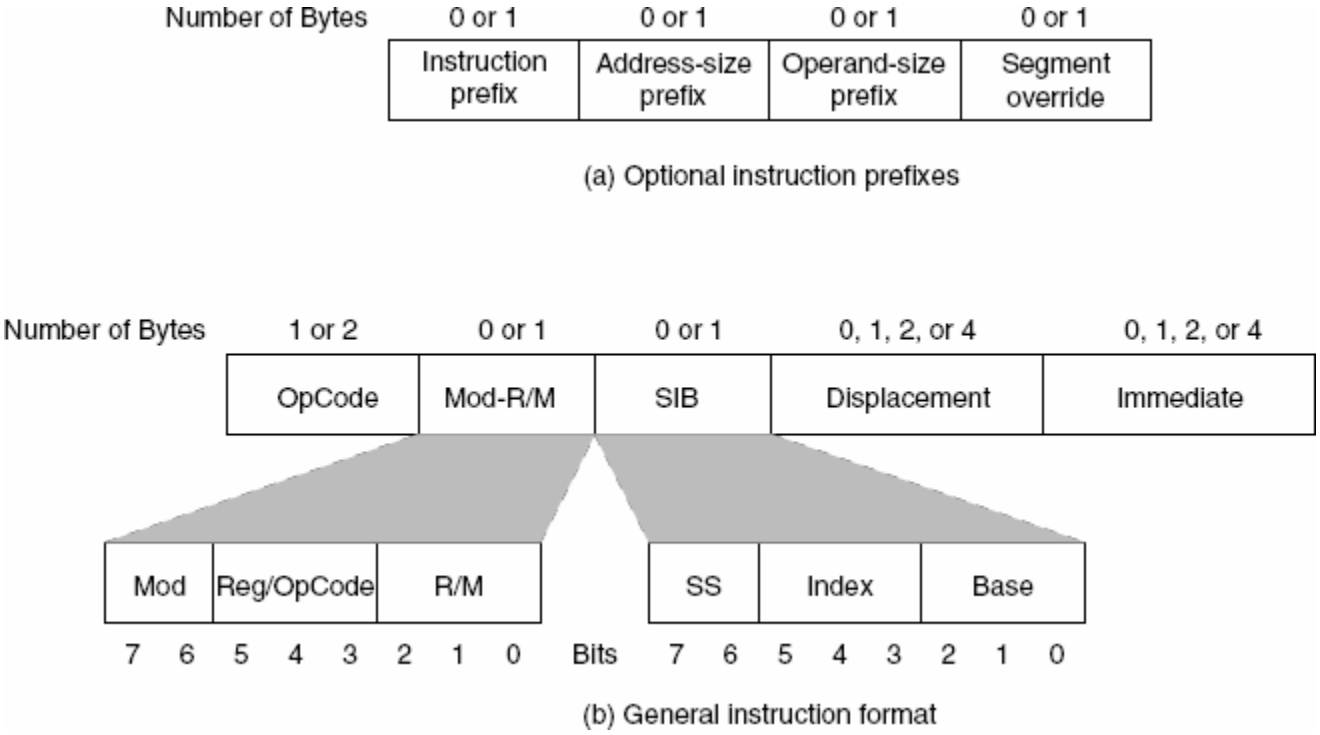
numeinstrucţiune *destinaţie****,*** *sursă*

Formatul intern al unei instrucţiuni este variabil, el putând ocupa între 1 şi 15 octeţi, având următoarea formă generală de reprezentare (*Instructions byte-codes from OllyDbg*):

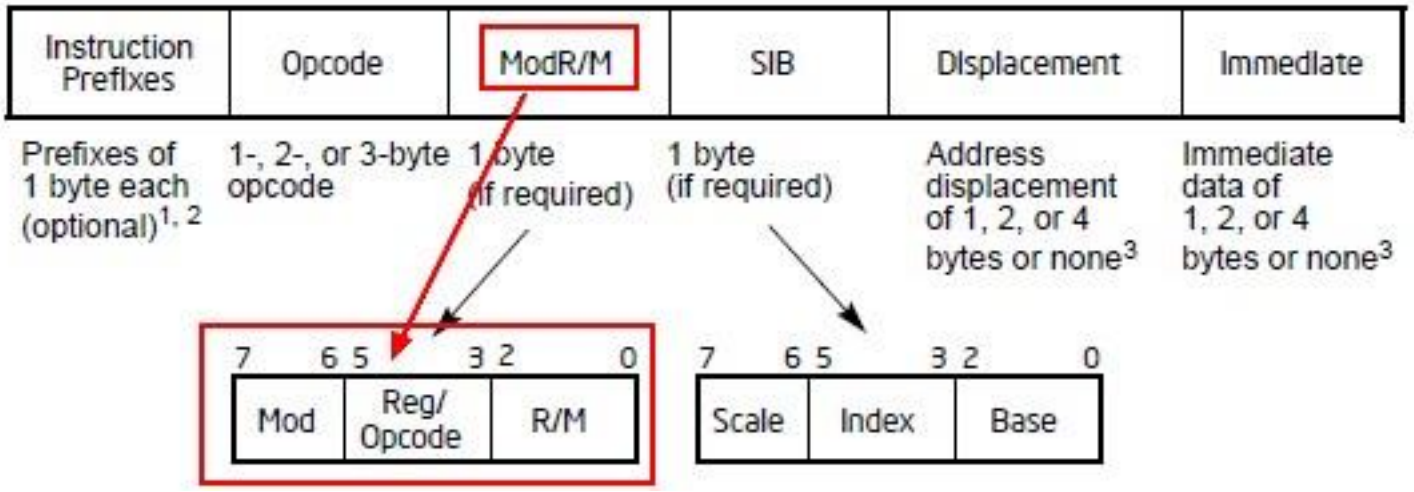
[prefixe] + cod + [ModR/M] + [SIB] + [deplasament] + [imediat]

Prefixele controlează modul în care o instrucţiune se execută. Acestea sunt opţionale (0 până la maximum 4) şi ocupă câte un octet fiecare. De exemplu, acestea pot solicita execuţia repetată (în buclă) a instrucţiunii curente sau pot bloca magistrala de adrese pe parcursul execuţiei pentru a nu permite accesul concurent la operanzi şi rezultate.

Operaţia care se va efectua este identificată prin intermediul a 1 sau 2 octeţi de *cod* (opcode), aceştia fiind singurii octeţi obligatoriu prezenţi, indiferent de instrucţiune.



Although the diagram seems to imply that instructions can be up to 16 bytes long, in actuality the x86 will not allow instructions greater than 15 bytes in length.



Octetul *ModR/M* (mod registru/memorie) specifică pentru unele dintre instrucţiuni natura şi locul operanzilor (registru, memorie). Acesta permite specificarea fie a unui registru, fie a unei locaţii de memorie a cărei adresă este exprimată prin intermediul unui offset (*http://datacadamia.com/intel/modrm*)

Pentru cazuri mai complexe de adresare decât cele codificabile direct prin ModR/M, combinarea acestuia cu octetul SIB (Scale – Index – Base) permite următoarea formulă generală de definire a unui offset:

offset = [bază] + [index × scală] + [constantă]
(SIB) (deplasament + imediat)

unde pentru bază şi index vor fi folosite valorile a doi regiştri iar scală este 1, 2, 4 sau 8. Regiştrii permişi ca bază sau / şi index sunt: EAX, EBX, ECX, EDX, EBP, ESI, EDI. Registrul ESP este disponibil ca bază însă nu poate fi folosit cu rol de index. (http://www.c-jump.com/CIS77/CPU/x86/X77_0100_sib_byte_layout.htm)

Majoritatea instrucţiunilor folosesc pentru reprezentare fie numai campul de cod, fie cod urmat de ModR/M.

Deplasament (displacement) apare în cazul unor forme de adresare particulare (operanzi din memorie) şi urmează direct după ModR/M sau SIB, când SIB este prezent. Acest câmp poate fi codificat fie pe octet, fie pe cuvânt, fie pe dublu cuvânt (32 biţi).

The most common addressing mode, and the one that's easiest to understand, is the *displacement-only* (or ***direct***) addressing mode. **The displacement-only addressing mode consists of a 32-bit constant that specifies the address of the target location**. The displacement-only addressing mode **is perfect for accessing simple scalar variables**. Intel named this the displacement-only addressing mode because a 32-bit constant (displacement) follows the MOV opcode in memory. **On the 80x86 processors, this displacement is an offset** from the beginning of memory (that is, address zero).

Displacement mode, **the operand's offset** is contained as part of the instruction as an 8-, 16-, or 32-bit displacement. The displacement addressing mode is found on few machines because, as mentioned earlier, it leads to long instructions. In the case of the x86, the displacement value can be as long as 32 bits, making for a 6-byte instruction. **Displacement addressing can be useful for referencing global variables.**

Ca şi consecinţă a imposibilităţii prezenţei mai multor câmpuri de ModR/M, SIB şi deplasament într-o instrucţiune, arhitectura 80x86 NU permite codificarea a două adrese de memorie în aceeaşi instrucţiune.

Valoare imediată oferă posibilitatea definirii unui operand ca fiind o constantă numerică pe 1, 2 sau 4 octeţi. Când este prezent, acest câmp apare întotdeauna la sfârşitul instrucţiunii.

All conditional branches such as JNE, etc. use a **disp8** specification, known also as a **short branch** or **short jump** (a signed 8-bit byte or a range from -128 to +127 bytes). Longer branches can be done using an **unconditional branch** such as a JMP instruction specifying for example **disp16** (2-byte 16-bit signed displacement) or **disp32** (4-byte 32-bit signed displacement).