24. x86 Instruction Prefix Bytes

- x86 instruction can have up to 4 prefixes.
- Each prefix adjusts interpretation of the opcode:

String manipulation instruction prefixes (prefixe precizate EXPLICIT de către programator !)

F3h = REP, REPE
F2h = REPNE

where

- REP repeats instruction the number of times specified by *iteration count* ECX.
- REPE and REPNE prefixes allow to terminate loop on the value of **ZF** CPU flag.

- 0xF3 is called REP when used with MOVS/LODS/STOS/INS/OUTS (instructions which don't affect flags)
  0xF3 is called REPE or REPZ when used with CMPS/SCAS
  0xF2 is called REPNE or REPNZ when used with CMPS/SCAS, and is not documented for other instructions.
  Intel's insn reference manual REP entry only documents F3 REP for MOVS, not the F2 prefix.

Instruction prefix - REP MOVSB F3:A4

Related string manipulation instructions are:

- MOVS, move string ; STOS, store string
- SCAS, scan string ; CMPS, compare string, etc.

See also string manipulation sample rogram: rep_movsb.asm

Segment override prefix causes memory access to use *specified segment* instead of *default segment* designated for instruction operand.
(acestea sunt prefixe precizate EXPLICIT de catre programator).

2Eh = CS	**26h = ES**
36h = SS	**64h = FS**
3Eh = DS	**65h = GS**

Segment override prefix - ES xlat 26 : D7 (ES:EBX)

mov eax, [ebx] - 8B03 mov eax, [SS:ebx] – 36:8B03

mov ax, [DS:ebx] – prefixare implicită cu 66 3E : 66 : 8B03 - prefixare explicită de către programator cu DS (3E – prefix DS, 66 – prefix op.size, 8B – cod mov Gv, Ev, 03 – EBX)
mov eax, [CS:ebx] – prefixare explicită de către programator cu CS 2E : 8B03
(2E – prefix CS, 8B – cod mov Gv, Ev, 03 – EBX)

ES LODSB – 26 : AC ; LODS byte ptr ES:[ESI]
ES CMPSB - 26 : A6 ; CMPS byte ptr ES:[ESI], byte ptr ES:[EDI]

ES STOSB - 26 : AA ; STOS byte ptr ES:[EDI] – superfluous segment override prefix

MOVSB - A4 ; MOVS byte ptr ES:[EDI], byte ptr DS:[ESI]
ES MOVSB - 26 : A4 ; MOVS byte ptr ES:[EDI], byte ptr ES:[ESI]
ES SCASB - 26 : AE ; SCAS byte ptr ES:[EDI] - superfluous segment override prefix

Operand override, **66h**. Changes size of data expected by default mode of the instruction e.g. 16-bit to 32-bit and vice versa.

Address override, **67h**. Changes size of address expected by the default mode of the instruction. 32-bit address could switch to 16-bit and vice versa

Aceste două tipuri de prefixe apar ca rezultat al unor moduri particulare de utilizare a instrucţiunilor (exemple mai jos), utilizări care vor provoca apariţia acestor prefixe la nivelul formatului intern al instrucţiunii. Aceste prefixe NU se precizează EXPLICIT prin cuvinte cheie sau rezervate ale limbajului de asamblare ci sunt generate de către asamblor atunci când se impune.

Operand size prefix – 0x66

Bits 32 - default mode of the below code

cbw ; 66:98 - deoarece rez este pe 16 biţi (AX)
cwd ; 66:99 - deoarece rez este format din 2 reg. pe 16 biţi (DX:AX)
cwde ; 98 - aici se respectă modul default pe 32 biţi – rez în EAX

push ax ; 66:50 – deoarece se încarcă pe stivă o val pe 16 biţi, stiva fiind organizată implicit (default) pe 32 biţi
push eax ; 50 - ok – utilizare consistentă cu modul default

mov ax, a ; 66:B8 0010 – deoarece rezultatul este pe 16 biţi

Bits 16 - default mode of the below code
cbw ; 98 - deoarece rez este pe 16 biţi (AX)
cwd ; 99 - deoarece rez este format din 2 reg pe 16 biţi (DX:AX)

cwde ; 66:98 - deoarece aici NU se respectă modul default pe 16 biţi – rez in EAX

Address size prefix – 0x67

Bits 32
mov eax, [bx] ; 67:8B07 - deoarece DS:[BX] este adresare pe 16 biţi

Bits 16
mov BX, [EAX] ; 67:8B18 – deoarece DS:[EAX] este adresare pe 32 biţi

Bits 16
push dword[ebx] ; 66:67:FF33 – Aici modul default este Bits 16; deoarece adresarea [EBX] este pe 32 biţi apare 67 şi deoarece se face push la un operand DWORD in loc de unul pe 16 biţi apare 66 ca prefix

push dword[CS:ebx] ; 2E:66:67:FF33 - 3 prefixe !!

rep push dword[CS:ebx] ; F3:2E:66:67:FF33 - 4 prefixe !!!
(rep push word ptr CS:[BP+DI] - superfluous REPxx fix ! - OllyDbg)

Bits 32

67:8B07 mov eax, [bx]; Offset_16_biti = [BX|BP] + [SI|DI] + [const]

(mov eax, dword ptr DS:[BX]) **67 – address size override prefix**

Bits 16

66:8B07 mov eax, [bx]; mov ax, word ptr DS:[edi]

(66 – operand size override prefix)

Definiţie.

Prefixele de instrucţiuni sunt construcţii ale limbajului de asamblare care apar opţional în componenţa unei linii sursă (prefixe explicite) sau a formatului intern al unei instrucţiuni (prefixe generate în mod implicit de către asamblor în două situaţii) şi care modifică comportamentul standard al acelor instrucţiuni (în cazul prefixelor explicite) sau care semnalează procesorului modificarea dimensiunii implicite de reprezentare a operanzilor sau/şi a adreselor, dimensiuni implicite stabilite prin directive de asamblare (BITS 16 su BITS 32).