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 catre 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.

  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. **(precizate EXPLICIT de catre programator).**

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

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 ultime doua tipuri de prefixe apar ca rezultat al unor moduri particulare de utilizare a instructiunilor (exemple mai jos), utilizari care vor provoca aparitia acestor prefixe la nivelul formatului intern al instructiunii. Aceste prefixe NU se precizeaza EXPLICIT prin cuvinte cheie sau rezervate ale limbajului de asamblare.

Instruction prefix - REP MOVSB

Segment override prefix - ES xlat sau mov ax, [cs:ebx]
(ES:EBX) (implicit se prefixeaza cu DS)

mov ax, DS:[ebx] – prefixare implicita
mov ax, [cs:ebx] – prefixare explicita de catre programator

Operand size prefix – 0x66

Bits 32 - default mode of the below code

……..

cbw ; 66:98 - deoarece rez este pe 16 biti (AX)

cwd ; 66:99 - deoarece rez este format din 2 reg pe 16 biti (DX:AX)

cwde ; 98 - deoarece aici se respecta modul default pe 32 biti – rez in EAX

push ax ; 66:50 – deoarece se incarca pe stiva o val pe 16 biti, stiva fiind organizata pe 32 biti

push eax ; 50 - ok – utilizare consistenta cu modul default

mov ax, a ; 66:B8 0010 – deoarece rez este pe 16 biti

Address size prefix – 0x67

Bits 32

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

Bits 16

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

Bits 16

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

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

Bits 16 - default mode of the below code

cbw ; 98 - deoarece rez este pe 16 biti (AX)

cwd ; 99 - deoarece rez este format din 2 reg pe 16 biti (DX:AX)

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