**24. x86 Instruction Prefix Bytes**

* x86 [instruction](http://www.c-jump.com/CIS77/images/x86_instruction_format.png) can have up to 4 prefixes.
* Each prefix adjusts interpretation of the opcode:

**String manipulation** instruction prefixes

**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.

Related string manipulation instructions are:

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

See also string manipulation sample program: [rep\_movsb.asm](http://www.c-jump.com/CIS77/samples/rep_movsb.htm)

**Segment override** prefix causes memory access to use *specified segment* instead of *default segment* designated for instruction operand.

**2Eh = CS**

**36h = SS**

**3Eh = DS**

**26h = ES**

**64h = FS**

**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 instruction. 32-bit address could switch to 16-bit and vice versa.

**Instruction prefix - REP MOVSB**

**Segment override prefix - ES xlat or mov ax, [cs:ebx]**

**(ES:EBX) mov ax, word ptr [CS:ebx]**

**mov ax, DS:[ebx]**

**mov ax, [cs:ebx]**

**Operand size prefix –**

**Bits 32**

**……..**

**cbw ; 66:98 - because result is on 16 bits (AX)**

**cwd ; 66:99 - because result is composed of 2 reg on 16 bits (DX:AX)**

**cwde ; 98 - because the default 32 bits mode is followed – rez in EAX**

**push ax ; 66:50 – because is pushed onto the stack a 16 bits value , the stack being organized on 32 bits**

**push eax ; 50 - ok – consistent usage with default mode**

**mov ax, a ; 66:B8 0010 – because rez is on 16 bits**

**Address size prefix – 0x67**

**Bits 32**

**mov eax, [bx] ; 67:8B07 because 16 bit addressing mode is used !!!**

**Bits 16**

**mov BX, [EAX] ; 67:8B18**

**Bits 16**

**push dword [ebx] ; 66:67:FF33**