

**December 15, 2023**

- main task of an assembler = generating the corresponding bytes
- at any given moment ONLY ONE segment of every type may be ACTIVE
- in 16 bits programming the segment registers CS, DS, SS, ES contained the STARTING ADDRESSES of the currently active segments
- in 32 bits programming the segment registers CS, DS, SS, ES contain the values of the SELECTORS of the currently active segments
- at any given moment during run time the CS:EIP combination of registers expresses /contain the address of the currently executed instruction
- these values are handled exclusively by BIU
- an assembly language instruction doesn't support/allow both of its explicit operands to be from the RAM memory
- that is because BIU may "bring" only one memory operand at a time (for 2 memory operands we would need 2 BIU, 2 segment registers sets etc)

$$\text{offset\_address} = [\text{base}] + [\text{index} \times \text{scale}] + [\text{constant}]$$

(SIB)                      (displacement + immediate)

$[prefixes] + code + [ModeR/M] + [SIB] + [displacement] + [immediate]$

- the first 2 elements from the offset address computation formula (base and index\*scale) are expressed by the SIB byte from the internal format formula
- the third element: the constant, if present, is expressed by the displacement and/or immediate fields
- SIB and displacement participate ONLY to the offset computation of the memory operand, if there is any
- “immediate” field may be also involved in offset computation, but it can also appear INDEPENDENTLY from a memory operand, expressing in such a case an immediate operand (mov eax, 7 ; 7 is “immediate” and no memory operand present in the instruction)
- if Modr/m tells us that we have a register operand the next 3 fields from the internal format formula are absent (because if the operand is a register it can NOT be in the same time also a memory operand or an immediate value)
- if Modr/m tells us that we have a memory operand => SIB byte is mandatory, followed MAYBE by displacement and/or immediate
- the field “immediate” may participate to the offset computation of a memory operand (providing the “constant” field from the offset computation formula) or may appear only by itself expressing the immediate value of an operand (example: mov ebx, 12345678h)

- the displacement field expresses the direct addressing memory access

- immediate field = numerical constants

- in the instructions used in our programs we will use almost exclusively only offsets, these being implicitly prefixed by one of the segment registers CS, DS, SS or ES. (ex. in debugger image - push variable -> DS:[40100...])

- offset = an address

- direct addressing means direct access to the memory operand based on its offset, without needing / specifying any register in the offset specification formula (so no base or index !)

- if registers appear in the offset computation formula (base or index) => indirect addressing

CS:EIP – The FAR (complete, full) address of the currently executing instruction

EIP – automatically incremented by the current execution

CS – contains the segment selector of the currently active segment and it can be changed only if the execution will switch to another segment

Mov cs, [var] – forbidden - syntactic ok (illegal instruction in OllyDbg...)

Mov eip, eax - syntax error – symbol 'eip' undefined

Jmp FAR somewhere ; CS and EIP will be both modified !

Jmp start1 ; NEAR jmp – only the offset will be modified, so only EIP !