Set CF=1 ; No way !

Mov ax, [v] ; a NASM specific – using ‘[]’ = DEREFERENCING operator = ‘*’ from C

A[7] = *(A+7) ; [] = addition pointer arithmetic operator

Mov eax, [8:1000h];

- Using assembly language directives
  BITS 16
  ………….

  A[7] = *(A+7);   “+ = []”

Mov eax, [a1] ;

Mov eax, [8:1000h];

```
Segment code
    Start:
          Jmp REAL_START
          V db 17
           V1 dw 54321
          REAL_START:   Mov ax, [v]
            Add ebx, eax
            Mul [v1]
            ………
```

Flags instructions do NOT have explicit operands !

In 16 bits programming the address of a memory location is handled explicitly by the programming with total freedom (segment + offset). In 32 bits programming only offsets are allowed to be handled by the programmer.

Limit = size of a segment

At the level of a source code, a programmer can use only ADDRESS SPECIFICATIONS, NEVER final effective addresses ! These may only be handled EXCLUSIVELY by the ADR component from BIU.

In contrast with the segment starting address, on which we don’t have any control as programmers in 32 bits programming, offsets must be handled by us !!

For running a program mandatory are the CODE segment and the STACK segment. In writing a source code the only mandatory segment is the CODE segment ! (because stack segment is automatically generated)

The binary form (machine code) of a .exe program does NOT contain ONLY the equivalent of the instructions to be run, but ALSO the WHOLE data segment generated correspondingly by the assembler/compiler !!!

In other words, the DATA segment is also saved together with the code segment on the disk as a part of the **.exe program !!**