

Directive de definire de date (Data definition directives)

OllyDbg – CODE segment începe la offset-ul 00401000 (sau 00402000)

OllyDbg – DATA segment începe la offset-ul 00402000 (sau 00401000)

Segment data

a1 db 0,1,2,'xyz' ; 00 01 02 'x' 'y' 'z' ; offset(a1- determinat la încărcarea lui OllyDbg)=00402000; **offset(a1) determinat la asamblare de către NASM = 0 !!!**
78 79 7A - codurile ASCII

db 300, "F"+3 ; 2C 49 - 'ascii code F + 3' - Warning – byte data (300) exceeds bounds!

a2 TIMES 3 db 44h ; 44 44 44 ; offset a2 = 00402008, însă offsetul determinat la asamblare de către NASM va fi 8 !!

a3 TIMES 11 db 5,1,3 ; 05 01 03 ... de 11 ori (33 octeți)

a4 dw a2+1, 'bc' ; offset(a2)=00402008h; a2+1=2009h; deci se generează
09 10 'b' 'c' = 09 10 62 63
, 'b', 'c' ; 'b' 00 'c' 00 = 09 10 62 00 63 00

a4 dw a2+1, 'bc' ; offset(a2)=00402008h; a2+1=2009h;
09 20 (correct, BUT... this particular 20h value it is only finally computable after LOADING !!! – deci offset-ul începuturilor de segmente este și el determinabil doar la momentul încărcării programului – LOADING TIME !!!)

Offset-ul variabilelor față de începutul segmentelor în care apar sunt constante (de tip pointer !, NU scalar !) determinabile la momentul asamblării !!

a44 dw 2009h ; 09 20

a5 dd a2+1, 'bcd' ; 09 20 40 00 62 63 64 00

a6 TIMES 4 db '13' ; 31 33 31 33 31 33 31 33
dw '13' ; 31 33 31 33 31 33 31 33

a7 db a2 ; syntax error OBJ format can only handle 16- or 32- relocation
(echiv. cu mov ah,a2)

a8 dw a2 ; 08 20

a9 dd a2 ; 08 20 40 00

a10 dq a2 ; 08 20 40 00 00 00 00 00

a11 db [a2] ; - expression syntax error – pt că [a2] NU este o expresie validă acceptată de către asamblor, nereprezentând “o valoare constantă determinabilă la momentul asamblării” ! Dereferențierea implicată aici este ceea ce deranjează asamblorul !!

a12 dw [a2] ; expression syntax error

a13 dd dword [a2] ; expression syntax error

a14 dq [a2] ; expression syntax error

a15 dd eax; expression syntax error

a16 dd [eax]; expression syntax error

mov ax, v ; Warning – 32 bit offset in 16 bit field !!!

Segment code (incepe intotdeauna la offset 00401000 - CINE decide asta ?)

Linkeditorul ia deciziile de acest tip. Adresa de baza pentru incarcarea PE-urilor, cel putin cea implicita (setata de catre linkeditorul de la Microsoft si nu numai) este 0x400000 in cazul executabilelor (respectiv 0x10000000 pentru biblioteci). Alink respecta aceasta **conventie** si completeaza in campul ImageBase al structurii IMAGE_OPTIONAL_HEADER din fisierul P.E. nou construit valoarea 0x400000. Cum fiecare "segment"/sectiune din program poate prevedea drepturi diferite de acces (codul este executabil, putem avea segmente read-only etc...), acestea sunt planificate sa inceapa fiecare la adresa cate unei noi pagini de memorie (4KiB, deci multiplu de 0x1000), fiecare pagina de memorie putand fi configurata cu drepturi specifice de catre incarcatorul de programe. In cazul unor programe de mici dimensiuni, implicatia este ca se va obtine urmatoarea harta a programului in memorie (la executare):

- programul este planificat a fi incarcat in memorie la exact adresa 0x400000 (insa aici vor ajunge structurile de metadate ale fisierului, nu codul sau datele programului in sine)
- primul "segment" va fi incarcat la 0x401000 (pun ghilimele deoarece nu este un segment propriu-zis ci doar o diviziune logica a programului, nu este asociat direct "segmentul" unui registru de segment – din aceasta pricina se prefera de multe ori denumirea de sectiune in loc de cea de segment)
- al doilea "segment" va fi incarcat la 0x402000 (segmentul pe care il va folosi procesorul pentru segmentare incepe la adresa 0 si are limita de 4GiB, indiferent de adresele si dimensiunile sectiunilor)
- va fi pregatit "segment" (sectiune) de importuri, "segment" de exporturi si "segment" de stack in ordinea decisa de catre linkeditor (si de dimensiuni prevazute tot de catre acesta), segmente ce vor fi incarcate de la 0x403000, 0x404000 si asa mai departe (incremente de 0x1000 cat timp au dimensiune suficient de mica, in caz contrar fiind nevoie a se folosi pentru increment cel mai mic multiplu de 0x1000 care permite suficient spatiu pentru continutul intregului segment)

Conform logicii de decizie a adreselor de inceput ale sectiunilor, putem concluziona ca aici avem o sectiune (de date probabil) inaintea celei de cod, continand sub 0x1000 octeti, motiv pentru care codul porneste imediat dupa, de la 0x402000, harta programului fiind la final: metadate (antete) de la 0x400000, cod la 0x401000 si date la 0x402000 (urmat bineinteles de alte "segmente" pentru stiva, importuri si, optional, exporturi).

Start:

```
    Jmp Real_start   (2 octeți) - 00401000
    a db 17          - offset(a) = 00401002
    b dw 1234h       - offset(b) = 00401003
    c dd 12345678h   - offset(c) = 00401005
```

Real_start:

```
    .....
    Mov eax, c ; EAX = 00401005
    Mov edx, [c] ; mov edx, DWORD PTR DS:[00401005]
    .....
    Mov edx, [CS:c] ; mov edx, DWORD PTR CS:[401005]
    Mov edx, [DS:c] ; mov edx, DWORD PTR DS:[401005]
    Mov edx, [SS:c] ; mov edx, DWORD PTR SS:[401005]
    Mov edx, [ES:c] ; mov edx, DWORD PTR ES:[401005]
```

Efectul fiind in toate cele 5 cazuri **EDX:=12345678h** **DE CE ???**

Explicația este direct legată de modelul de memorie flat – toate segmentele descriu în realitate întreaga memorie, începând de la 0 și până la capătul primilor 4GiB ai memoriei. Ca atare, [CS:c] sau [DS:c] sau [SS:c] sau [ES:c] vor accesa aceeași locație de memorie însă cu drepturi de acces potențial diferite. Deși toți selectorii indică segmente identice ca adresă și dimensiune, aceștia pot avea diferențe în cum le sunt completate alte campuri de control și de acces ale descriptorilor de segment indicați de către ei.

Modelul flat ne asigura ca mecanismul de segmentare este transparent pentru noi, noi nu sesizam diferente între segmente și, ca atare, scapăm complet de grija segmentării (înșă ne interesează împărțirea logică în segmente a programului, motiv pentru care folosim secțiuni/"segmente" separate pentru cod date). Acest lucru este valabil înșă doar cât timp ne limităm la CS/DS/ES și SS! **Selectorii FS și GS indică înspre segmente speciale care nu respectă întotdeauna modelul flat (rezervate interacțiunii programului cu S.O-ul), mai precis, [FS:c] nu garantează indicarea aceleiași zone de memorie ca și [CS:c]!**

De verificat:

```
    Mov edx, [FS:c] ; mov edx, DWORD PTR FS:[401005]
    Mov edx, [GS:c] ; mov edx, DWORD PTR GS:[401005]
```