Limbajul maşină al unui sistem de calcul (SC) este format din totalitatea instrucţiunilor maşină puse la dispoziţie de procesorul SC. Acestea se reprezintă sub forma unor şiruri de biţi cu semnificaţie prestabilită.

Limbajul de asamblare al unui calculator este un limbaj de programare în care setul de bază al instrucţiunilor coincide cu operaţiile maşinii şi ale cărui structuri de date coincid cu structurile primare de date ale maşinii. Limbaj simbolic. Simboluri - Mnemonice (nume de instructiuni sau directive) + etichete.

Elementele cu care lucrează un asamblor sunt:

\* etichete - nume scrise de utilizator, cu ajutorul cărora se pot referi date (nume de variabile – de obicei in segmentul de date – etichete de date) sau zone de memorie (destinatie a unui salt – de obicei in segmentul de cod – etichete de cod). Valoarea unei etichete în limbaj de asamblare este un număr întreg reprezentând adresa (FAR) instrucţiunii, directivei sau datelor ce urmează etichetei.

\* instrucţiuni - scrise sub forma unor mnemonice care sugerează acţiunea. Asamblorul generează octeţii care codifică instrucţiunea respectivă.

\* directive - sunt indicaţii date asamblorului în scopul generarii corecte a octetilor. Ex: relaţii între modulele obiect, definirea unor segmente, indicaţii de asamblare condiţionată, directive de generare a datelor.

\* contor de locaţii - număr întreg gestionat de asamblor. În fiecare moment, valoarea contorului coincide cu numărul de octeţi generaţi corespunzător instrucţiunilor şi directivelor deja întâlnite în cadrul segmentului respectiv (deplasamentul curent în cadrul segmentului). Programatorul poate utiliza această valoare (accesare doar în citire!) prin simbolul '$'.

FORMATUL UNEI LINII SURSA

[etichetă[:]] [prefixe] [mnemonică] [operanzi] [;comentariu]

aici: jmp acolo ; avem etichetă + mnemonică + operand + comentariu

repz cmpsd ; prefix + mnemonică + comentariu

start: ; etichetă + comentariu 3

; doar un comentariu (care putea lipsi si el)

a dw 19872, 42h ; etichetă + mnemonică + 2 operanzi + comentariu

RANDOM

lea eax, [v] ; încarcă în registrul eax adresa (offsetul) variabilei v (4 octeţi) – echiv. Mov eax, v

Există două tipuri de mnemonice: mnemonice de instrucţiuni şi nume de directive. Directivele dirijează asamblorul. Ele specifică modul în care asamblorul va genera codul obiect. Instrucţiunile dirijează procesorul.

Operanzii sunt parametri care definesc valorile ce vor fi prelucrate de instrucţiuni sau de directive. Ei pot fi regiştri, constante, etichete, expresii, cuvinte cheie sau alte simboluri. Semnificaţia operanzilor depinde de mnemonica instrucţiunii sau directivei asociate.

Expresiile (operanzi + operatori) sunt evaluate în momentul asamblării (adică, valorile lor sunt determinabile la momentul asamblării, cu excepţia acelor părţi care desemnează conţinuturi de regiştri şi care vor fi determinate la execuţie).

Valoarea operanzilor este calculată în momentul asamblării pentru operanzii imediaţi si pt operanzii cu adresare directa (e vorba doar de offset aici!) şi în momentul execuţiei (run-time) pentru operanzii registru şi cei adresaţi indirect. Deplasamentele (offset-urile) etichetelor de date şi de cod reprezinta valori determinabile la momentul asamblarii care rămân constante pe tot parcursul execuţiei programului. Ex.: mov eax, et

\*

- constanta hexazecimală B2A poate fi exprimată ca 0xb2a, 0xb2A, 0hb2a, 0b12Ah, 0B12AH, etc;

- valoarea zecimală 123 poate fi specificată ca 123, 0d123, 0d0123, 123d, 123D

\*

Modul de invocare/accesare directă - mov eax, ebx

Invocare/accesare indirecta - pentru a indica locaţiile de memorie - mov eax, [ebx]

Deplasamentul unui operand cu adresare directă este calculat în momentul asamblării (assembly time). Adresa fiecărui operand raportată la structura programului executabil (mai precis stabilirea segmentelor la care se raportează deplasamentele calculate) este calculată în momentul editării de legături (linking time). Adresa fizică efectivă este calculată în momentul încărcării programului pentru execuţie (loading time – acest proces final de ajustare a adreselor numindu-se RELOCAREA ADRESELOR = Address Relocation).

Un deplasament utilizat ca operand in cadrul unui program este întotdeauna raportat la un registru de segment. Acest registru poate fi specificat explicit sau, în caz contrar, se asociază de către asamblor în mod implicit un registru de segment. Regulile limbajului de asamblare pentru asocierile implicite sunt:

- CS pentru etichete de cod destinaţie ale unor salturi (jmp, call, ret, jz etc);

- SS în adresări SIB ce foloseste EBP sau ESP drept bază (indiferent de index sau scală);

- DS pentru restul accesărilor de date

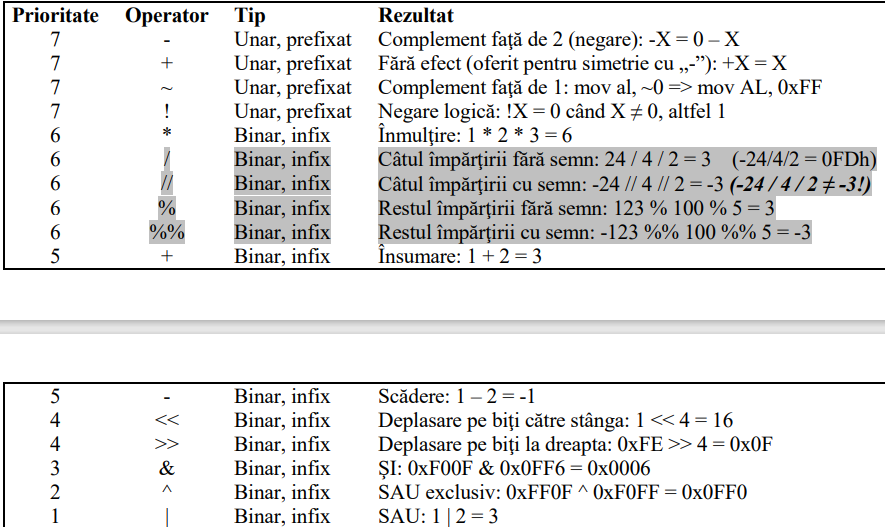
ES poate fi utilizat doar in specificari explicite de adresa (ex: ES:[Var] sau ES:[ebx+eax\*2-a] ) sau in cazul unor instructiuni specific pe siruri (MOVSB)

\*

mov dh, [edx + ecx \* 4 + 3] octetul superior dintr-un dword dintr-un sir parcurs cu edx de pe poz indicate de ecx

\*

Operatorii efectuează calcule cu valori constante SCALARE determinabile la momentul asamblării (cu exceptia adunarii si scaderii de constante la un pointer care furnizeaza o valoare de tip adresa si cu exceptia formulei de calcul a offset-ului unui operand). Instrucţiunile efectuează calcule cu valori ce pot fi necunoscute până în momentul execuţiei. Operatorul de adunare (+) efectuează adunarea în momentul asamblării; instrucţiunea ADD efectuează adunarea în timpul execuţiei. Evaluarea expresiilor numerice se face pe 64 de biţi la operatori



Operatorul de specificare a segmentului: Operatorul de specificare a segmentului (:) comandă calcularea adresei FAR a unei variabile sau etichete în funcţie de un anumit segment. Sintaxa este: segment:expresie

[ss: ebx+4]; deplasamentul e relativ la SS

[es:082h] ; deplasamentul e relativ la ES

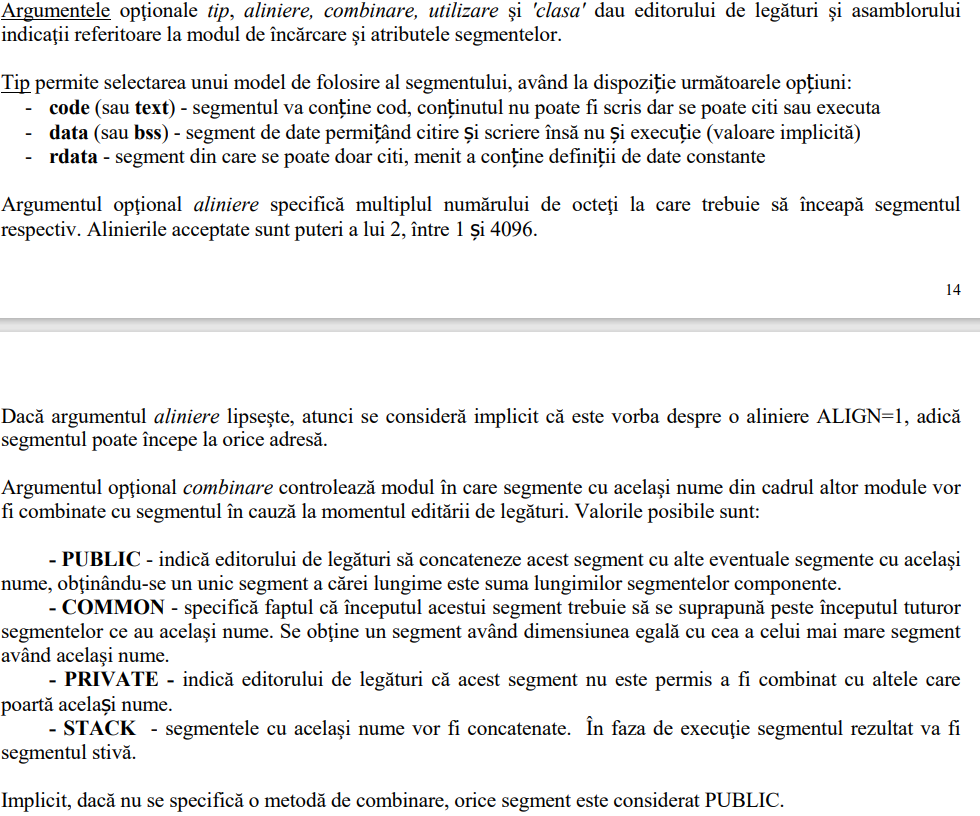
10h:var ; segmentul este indicat de selectorul 10h, iar offsetul este valoarea etichetei var.

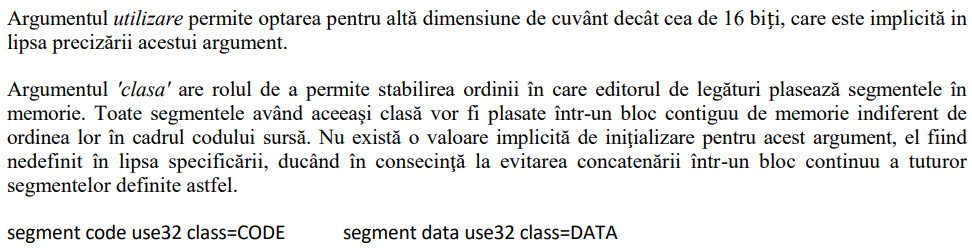
Operatori de tip : byte/word/dwors/qword expresie sau NEAR/FAR pt etichete de cod. Specificatorul QWORD nu intervine niciodata explicit in cod pe 32 de biti!!

DIRECTIVE

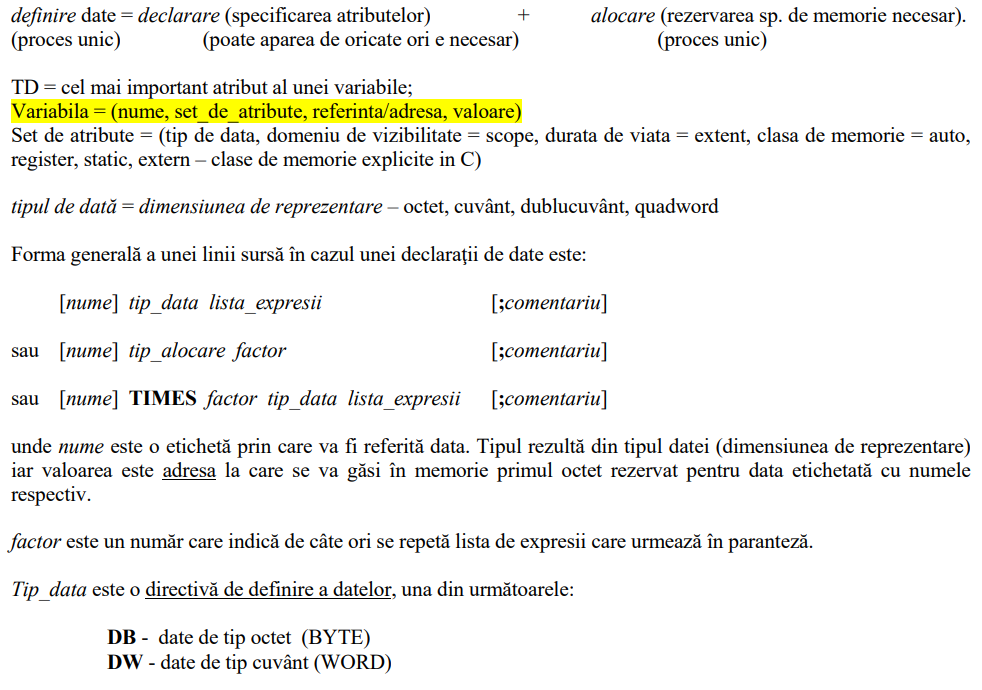
Directivele indică modul în care sunt generate codul şi datele în momentul asamblării.

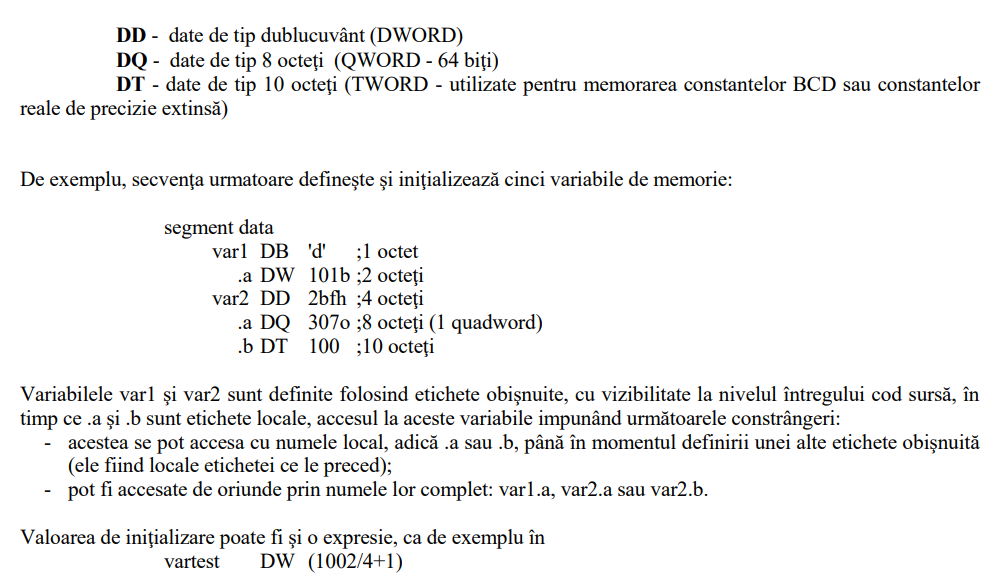
1.segment: Directiva SEGMENT permite direcţionarea octeţilor de cod sau date emişi de către un asamblor înspre segmentul precizat, segment care poartă un nume şi are asociate diverse caracteristici. SEGMENT nume [tip] [ALIGN=aliniere] [combinare] [utilizare] [CLASS=clasă]. $$ - inceput de segment. Cu excepţia numelui, toate celelalte câmpuri sunt opţionale atât din punct de vedere a prezenţei cât şi a ordinii în care sunt specificate.

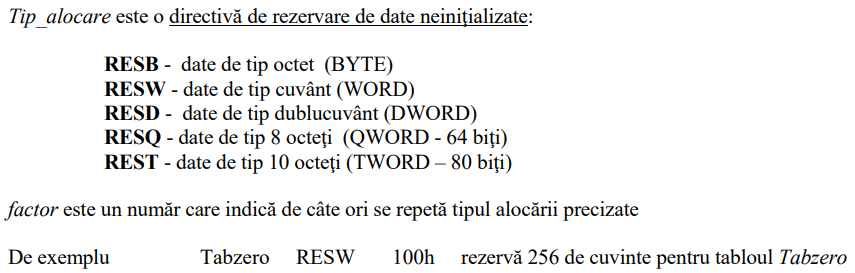




DATA







DIRECTIVA TIMES: TIMES factor tip\_data expresie (exemplu Tabchar TIMES 80 DB ‘a’)

DIRECTIVA EQU: permite atribuirea, în faza de asamblare, unei valori numerice sau şir de caractere unei etichete fără alocarea de spaţiu de memorie sau generare de octeţi. nume EQU expresie ( exemplu BUFFER\_SIZE EQU 1000h)