# Editarea legaturilor si incarcarea programelor

Studentii:

- Bestea Vladut
- Gradinaru-Tascau Gabriel
- Perianu Razvan
- Pirvu Sorin

Majoritatea programelor sunt formate din mai mult de o procedura. In general, asambloarele translateaza o procedura la un moment dat si scriu iesirea pe disc. Pentru ca programul sa poata fi executat, toate procedurile trebuie gasite si legate in mod corespunzator. Daca nu este disponibila memorie virtuala, programul editat trebuie incarcat explicit in memoria principala. Programele care realizeaza aceste functii sunt denumite editor de legaturi (linker) sau incarcator si editor de legaturi (linking loader) de programe.

Traducerea completa a programului sursa necesita doua etape:

1. Compilarea sau asamblarea procedurilor sursa
2. Editarea legaturilor pentru modulele obiect

Prima etapa este realizata de catre compilator sau asamblor, cea de-a doua de catre editorul de legaturi.

Translatarea de la procedura sursa la modulul obiect reprezinta o modificare de nivel, deoarece limbajul sursa si limbajul obiect au notatii si instructiuni diferite. Procesul de editare de legaturi nu reprezinta totusi o modificare a nivelului deoarece atat intrarile cat si iesirile editorului de legaturi sunt programe pentru aceeasi masina virtuala. Functia editorului de legaturi este de a colecta procedurile translatate separat si de a lega impreuna pentru a fi executate unitar ca program binar executabil. Pentru sistemele MS-DOS, Windows 95/98 si NT modulele obiect au extensia *.obj,* iar programele binare executabile au extensia *.exe.* Pe UNIX modulele obiect au extensia *.o*, iar programele binare executabile nu au extensie.

Compilatoarele si asambloarele translateaza fiecare procedura sursa ca o entiate separata. Altfel, modificare unei declaratii intr-o procedura sursa ar necesita ca toate procedurile sursa sa fie retranslatate. Utilizand tehnica modulelor obiect separate se va retransalta doar procedura modificata, dar va fi necesar sa se reediteze legaturile pentru toate modulele obiect. Editarea legaturilor fiind mult mai rapida decat translatarea, se va economisi timp considerabil, mai ales in cazul programelor cu sute sau mii de module.

**Activitatile realizate de editorul de legaturi**

La inceputul primei treceri in cadrul procesului de asamblare, numaratorul de locatii de instructiuni este 0, echivalent cu presupunerea ca modulul obiect va fi plasat la adresa (virtuala) 0 la momentul executiei. Pentru a executa programul, editorul de legaturi aduce

modulele obiect in memoria principala pentru a forma imaginea programului binar executabil. Se doreste a se construi o imagine exacta a spatiului de adrese virtual al programului executabil in cadrul editorului de legaturi si de a pozitiona toate modulele obiect la locatiile lor corecte. Daca nu exista suficienta memorie(virtuala) pentru a forma imaginea, se poate utiliza un fisier pe disc. De obicei, o mica portiune din memorie incepand de la adresa zero este utilizata pentru vectorii de intrerupere, comunicatia cu sistemul de operare, capturarea referintelor de memorie neinitializate sau pentru alte scopuri, asa ca programul incepe adesea de la o adresa mai mare ca 0.

Problemele care apar cand programul este incarcat in imaginea fisierului binar executabil poarta numele de **probleme de relocare** si se datoreaza faptului ca fiecare modul obiect formeaza separat un spatiu de adresa. Majoritatea versiunilor de Windows si UNIX ofera numai un spatiu liniar de adrese, astfel ca toate modulele trebuie sa fie concatenate intr-un singur spatiu de adrese. O alta problema este cea a **referintelor externe** datorata faptului ca asamblorul nu are o modalitate de a sti la ce adresa sa insereze instructiunea pentru un alt modul, adresa acestuia nefiind cunoscuta pana in momentul editarii legaturilor.

Editorul de legaturi poate rezolva aceste probleme unind spatiile separate de adresa ale modulelor obiect intr-un singur spatiu liniar de adrese, in urmatorii pasi:

1. Construieste o tabela cu toate modulele obiect si dimensiunile acestora
2. Pe baza tabelei asigneaza o adresa de start pentru fiecare modul obiect
3. Determina toate instructiunile cu referire la memorie si aduna fiecareia o **constanta de realocare (relocation constant)**, egala cu adresa de start a modulului sau.
4. Determina toate instructiunile ce fac referire la alte proceduri si insereaza adresa acelor proceduri in pozitiile respective.

**Structura unui modul obiect**

Modulele obiect contin adesea sase parti. Prima contine numele modulului, anumite informatii necesare pentru editorul de legaturi, ca lungimea diferitelor parti ale modulului si uneori data asamblarii.

A doua parte a modulului obiect este o lista de simboluri definite in modul, pe care alte module le pot referi, impreuna cu valorile lor. Programatorul in limbaj de asamblare

indica care simboluri trebuie sa fie declarate ca puncte de intrare (entry points), prin utilizarea unei pseudoinstructiuni, ca PUBLIC.

A treia parte consta dintr-o lista de simboluri utilizate in modul dar care sunt definite in alte module, impreuna cu o lista a instructiunilor masina care folosesc aceste simboluri. Editorul de legaturi utilizeaza aceasta lista pentru a putea sa insereze adresele corecte si instructiunile ce utilizeaza simboluri externe. O procedura poate apela alte proceduri translatate independent prin declararea numelor procedurilor apelate ca fiind externe. Programatorul in limbaj de asamblare indica ce simboluri sunt declarate ca simboluri externe (external simbols) prin utilizarea unei pseudoinstructiuni ca EXTRN.
A patra parte este codul asamblat si constantele. Aceasta parte este singura care va fi incarcata in memorie pentru a fi executata. Celelalte cinci parti vor fi utilizate de editorul de legaturi si apoi eliminate inainte de inceperea executiei.

Cea de-a cincea parte este dictionarul de relocare. Instructiunile ce contin adrese de memorie trebuie sa aiba adunata o constanta de relocare. Deoarece editorul de legaturi nu are o modalitate de a spune, prin inspectie, care dintre cuvintele de date din partea a patra contin instructiuni masina si care contin constante, informatia despre adresele care vor fi relocate este furnizata de aceasta tabela. Informatia poate lua forma unei tabele de biti, cu 1 bit pentru adresa potential relocabila, sau o lista explicita de adrese ce vor fi relocate.

A sasea parte este o indicatie de sfarsit de modul, uneori o suma de verificare pentru a descoperi erori de citire a modulului si adresa la care incepe executia.
Cele mai multe editoare de legaturi necesita doua treceri. In prima editorul de legaturi citeste toate modulele obiect si construieste o tabela a lungimilor si numelor modulelor si o tabela de simboluri constand din toate punctele de intrare si referintele externe. In a doua trecere modulele obiect sunt citite, relocate si sunt editate legaturile in cate un modul la un moment dat.

**Momentul legarii si relocarii dinamice**

Intr-un sistem cu multiprogramare, un program poate fi incarcat in memoria principala, executat pentru un timp, scris pe disc si apoi incarcat din nou in memoria principala pentru a fi executat din nou. Intr-un sistem mare, cu multe programe, este dificil

sa se asigure ca un program este incarcat din nou in aceleasi locatii, de fiecare data. Informatia de relocare se poate pierde in urma scrierii pe disc. Chiar daca informatia de relocare ar fi inca disponibila, costul de a reloca toate adresele de fiecare data cand programul este readus in memorie ar putea fi mare.

Problema mutarii programelor care au fost deja legate si relocate este strans legata de momentul la care este terminata legarea finala a numerelor simbolice pentru adresele de memorie. Momentul la care este stabilita adresa curenta de memorie principala este denumit momentul atribuirii de adresa (binding time). Exista cel putin sase posibilitati pentru momentul atribuirii de adresa:

1. Cand programul este scris
2. Cand programul este translatat
3. Cand se face editarea legaturilor dar inainte de a fi incarcat
4. Cand programul este incarcat
5. Cand un registru de baza, utilizat pentru adresare, este incarcat
6. Cand se executa instructiunea ce contine adresa

Daca o instructiune ce contine o adresa de memorie este mutata dupa atribuirea adresei, ea va fi incorecta. Daca translatorul furnizeaza ca iesire un cod binar executabil, atribuirea adresei s-a facut la momentul translatiei si programul trebuie sa fie rulat la adresa la care translatorul se astepta sa fie executat. Metoda prezentata leaga nume simbolice de adrese absolute in timpul editarii de legaturi, de aceea mutarea programelor dupa editarea de legaturi esueaza.

Prima problema apare cand se face asocierea intre numele simbolice si adresele virtuale. A doua la momentul cand adresele virtuale sunt asociate cu cele fizice. Numai dupa ce ambele operatii au avut loc atribuirea adresei este completa. Cand editorul de legaturi uneste spatiile separate de adrese ale moduleleor obiect intr-un singur spatiu liniar de adrese, se creeaza de fapt un spatiu virtual de adrese. Acest lucru este adevarat indiferent daa memoria virtuala este utilizata sau nu.

Daca spatiul de adrese este paginat, adresele virtuale ce corespund numelor simbolice au fost deja determinate chiar daca adresele fizice din memoria principala vor depinde de continutul tabelei de pagini la momentul la care ele sunt utilizate. Un program binar executabil este de fapt o asociere a numerelor simblice cu adresele virtuale. Mutarea programelor in memoria principala, chiar dupa ce ele au fost asociate unui

spatiu virtual de adrese, va fi inlesnita de orice mecanism care permite modificarea usoara a punerii in corespondenta a adreselor virtuale. Un astfel de mecanism este paginarea. Dupa ce un program a fost mutat in memoria principala, numai tabela sa de pagini trebuie modificata, nu programul insusi.

Un al doilea mecanism este utilizarea in timpul executiei a unui registru de relocare. Pe masinile ce utilizeaza aceasta tehnica de relocare registrul refera intotdeauna adresa de memorie fizica de start a programului curent. La toate adresele de memorie se va aduna prin hardware continutul registrului de relocare inainte de a fi transmise la memorie. Intregul proces de relocare este transparent pentru programele utilizator. Ele nu stiu nici macar daca a fost utilizat. Cand un program este mutat, sistemul de operare trebuie sa actualizeze registrul de relocare. Acest mecanism este mai putin general decat paginarea deoarece intregul program trebuie sa fie mutat ca un intreg.

Un al treilea mecanism este posibil pe masinile ce pot referi memoria relativ la contorul program. Ori de cate ori un program este mutat in memoria principala trebuie sa fie actualizat numai numaratorul de instructiuni. Un program ale carui referinte la memorie sunt fie relative la contorul program, fie absolute se spune ca este independent de pozitie. O procedura independenta de pozitie poate fi plasata oriunde in spatiul virtual de adrese, fara a fi nevoie de relocare.

La inceputul primei treceri in cadrul procesului de asamblare, numaratorul de locatii de instructiuni ia valoarea 0, echivalent cu a presupune ca la momentul executiei modulul obiect va incepe la adresa virtuala 0. Pentru a executa programul, editorul de legaturi aduce modulele obiect in memoria principala pentru a forma imaginea programului binar executabil. De aceea el trebuie sa construiasca o imagine exacta a spatiului de adrese virtual al programului si sa pozitioneze toate modulele obiect la locatiile lor corecte.(Daca nu exista suficienta memorie virtuala se poate utiliza un fisier de pe disc).

De obicei, o mica portiune de memorie incepand de la adresa 0 este utilizata pentru comunicarea cu sistemul de operare, pentru vectorii de intreruperi sau pentru alte scopuri, asa ca programul incepe de la o adresa mai mare decat 0. Sa presupunem ca avem un modul A care apeleaza un modul B ce contine un salt la o instructiune. Daca doar s-ar realiza imaginea programului prin plasarea liniar in memorie a modulelor obiect si apoi s-ar executa, programul nu ar putea realiza instructiunea. Aceasta nu s-ar mai afla la aceeasi locatie de memorie precizata prin salt, adresa sa fiind deplasata.

**Momentul legarii si relocarii dinamice**

In cadrul procesului de calcul este necesar ca un program sa fie rulat de mai multe ori, la diferite intervale de timp. In cazul unui sistem cu multiprogramare, este dificil sa se asigure ca un program sa fie incarcat de fiecare data in aceleasi locatii. Informatia de relocare se poate pierde in urma scrierii pe disc. Chiar daca informatia de relocare ar fi inca disponibila, nu este eficient a reloca toate adresele de fiecare data cand programul este readus in memorie.

Apare problema mutarii programelor care au fost deja legate si relocate. Momentul la care este stabilita adresa curenta de memorie principala este denumit momentul atribuirii de adresa (binding time). Exista cel putin sase posibilitati pentru momentul atribuirii de adresa:

1. Cand programul este scris
2. Cand programul este translatat
3. Cand se face editarea legaturilor dar inainte de a fi incarcat
4. Cand programul este incarcat
5. Cand un registru de baza, utilizat pentru adresare, este incarcat
6. Cand se executa instructiunea ce contine adresa

Metoda de editare de legaturi descrisa pana acum leaga nume simbolice de adrese absolute. Mutarea programelor dupa editarea legaturilor esueaza. Prima problema apare cand se face asocierea intre numele simbolice si adresele virtuale. A doua la momentul cand adresele virtuale sunt asociate cu cele fizice. Numai dupa ce ambele operatii au avut loc atribuirea adresei este completa. Cand editorul de legaturi uneste spatiile separate de adrese ale modulelor obiect intr-un singur spatiu liniar de adrese, se creeaza de fapt un spatiu virtual de adrese. Acest lucru este adevarat indiferent daca memoria virtuala este utilizata sau nu.

Un program binar executabil este de fapt o asociere a numelor simbolice cu adresele virtuale. Pentru a muta programele in memoria principala ( dupa ce li s-au asociat un spatiu virtual de adrese) este necesar un mecanism de modificare a corespondentei adreselor virtuale cu cele fizice, cum este paginarea. La mutarea in memoria principala nu se modifica programul, ci tabela sa de pagini.

Un al doilea mecanism este utilizarea in timpul executiei a unui registru de relocare. Acesta refera intotdeauna adresa de memorie fizica de start a programului curent. Inainte de a fi transmise la memorie, se aduna prin hardware la toate adresele de memorie continutul registrului de relocare. Intregul proces de relocare este transparent pentru programele utilizator. Cand un

program este mutat, sistemul de operare trebuie sa actualizeze registrul de relocare. Acest mecanism este mai putin general decat paginarea deoarece intregul program trebuie sa fie mutat ca un intreg.

Un al treilea mecanism este posibil pe masinile ce pot referi memoria relativ la contorul program. Ori de cate ori un program este mutat in memoria principala trebuie sa fie actualizat numai numaratorul de instructiuni. Un program ale carui referinte la memorie sunt relative la contorul program spune ca este independent de pozitie. Deci o procedura independenta de pozitie va putea fi plasata oriunde in spatiul virtual de adrese, fara a fi nevoie de relocare.

**Legarea dinamica**

Atunci cand editorul de legaturi foloseste legarea dinamica pentru construirea unei aplicatii vor fi rezolvate toate referintele catre modulele necesare dar acestea nu vor fi copiate in varianta executabila. Linkerul adauga etichete de start-up cu ajutorul carora vor fi incarcate librariile necesarae la rularea aplicatiiei. Fiecare apel al unei librarii este pastrat intr-un tabel unde se trece locatia respectivei librarii, astfle ca la urmatoarea apelare se va folosi referinta indirecta din tabel.

Este acea legare cu “viata scurta” ce se face imediat inaintea unei singure executii, precedata de o analiza a sistemului si a aplicatiei prin care se evidentiaza modulele necesare programului in acel moment.

Legarea dinamica desi pare mai laborioasa este mai utila la sistemele complexe formate din mai multe module care nu ar putea fi incarcate toate in memorie la un moment dat. Inainte de executie se va face un bilant al modulelor necesare si acestea vor fi legate si apoi incarcate in momentul executiei.

Legarea dinamica se face in doua faze:

- in faza I se verifica modulele si se completeaza ESTAB-ul
- in faza II se va face legarea propriu-zisa si se va completa cu valori externe

In organigramele ce descriu legarea dinamica vom folosi mai multe structuri: PROGADR – numarator cu adresele de incarcare a programelor rulate

CSADR – adresa de inceput pentru modulul care se incarca in momentul curent

CSLTH – numarator cu lungimea modulului care se incarca in momentul curent

ESTAB – tabel cu variabile externe si completat cu tabelul cu variabile publice, contine referirile comune din modulul complex

EXECADR – va contine adresa de start a intregului program

Aceasta problema poarta numele de **problema de relocatare (**relocation problem**)** si apare pentru ca fiecare modul obiect formeaza separat un spatiu de adresa. Majoritatea versiunilor de Windows si UNIX ofera numai un spatiu liniar de adrese, astfel ca toate modulele trebuie sa fie concatenate intr-un singur spatiu de adrese. Pentru a fi relocatabil, un modul obiect trebuie sa poata fi incarcat la adrese de memorie stabilite ulterior momentului creatiei lui, iar rularea sa sa produca rezultate identice, independente de adresa de memorie la care este plasat. O alta problema este cea a **referintelor externe** datorata faptului ca asamblorul nu are o modalitate de a sti la ce adresa sa insereze instructiunea pentru un alt modul, adresa acestuia nefiind cunoscuta pana in momentul editarii legaturilor.

Editorul de legaturi rezolva aceste probleme unind spatiile separate de adresa ale modulelor obiect intr-un singur spatiu liniar de adrese, in urmatorii pasi:

1. Construieste o tabela cu toate modulele obiect si dimensiunile acestora
2. Pe baza tabelei asigneaza o adresa de start pentru fiecare modul obiect
3. Determina toate instructiunile cu referire la memorie si aduna fiecareia o **constanta de relocatare (relocation constant)**, egala cu adresa de start a modulului sau.
4. Determina toate instructiunile ce fac referire la alte proceduri si insereaza adresa acelor proceduri in pozitiile respective.

Astfel, tabelele modulului obiect final va furniza numele, lungimea si adresa de start a fiecarui modul.

**Legarea dinamica in Windows**

Pentru editarea dinamica a legaturilor exista un format special de fisier, DLL (Dinamic Link Library) ce poate contine proceduri, date, sau ambele. Fisierele DLL sunt utilizate pentru ca doua procese sa poata partaja datele sau procedurile din biblioteca. Majoritatea fisierelor DLL au extensia *.dll*, dar pot fi folosite si *.drv* (pentru biblioteci de drivere) si *.fon* (pentru biblioteci de fonturi).

Editorul de legaturi construieste un fisier DLL plecand de la o colectie de fisiere de intrare, de obicei o colectie de proceduri de biblioteca utilizabile de mai multe procese, ca proceduri de interfata catre biblioteca de apeluri de sistem Windows si biblioteci mari de grafica. Efectul va fi

economisirea spatiului de memorie si de disc. Legate static la fiecare program care le utilizeaza, bibliotecile uzuale ar aparea in mai multe programe binare executabile pe disc si in memorie. Legate dinamic, ele apar o singura data pe disc, o singura data in memorie.

Aceasta metoda usureaza si munca furnizorului de software, care poate actualiza procedurile bibliotecii si distribui doar noile fisiere DLL fara a modifica programele binare principale.

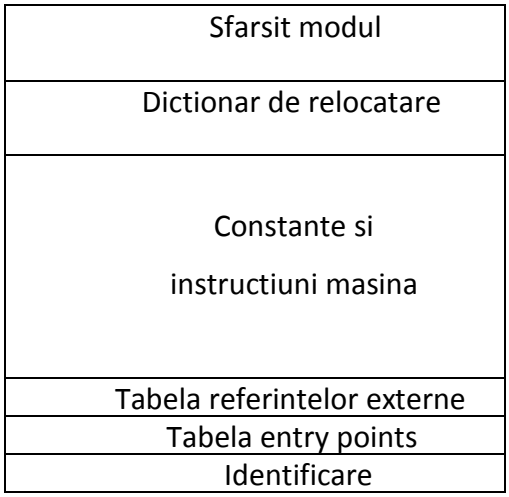
Un fisier DLL este diferit de unul binar executabil pentru ca neavand un program principal nu poate fi executat de sine statator. In schimb contine proceduri suplimentare, fara legatura cu cele din biblioteca, pentru administrarea resurselor cerute de DLL, proceduri care pot aloca sau elibera memorie.

Un program se poate lega la un DLL in doua moduri. In primul mod, denumit **legarea implicita,** programul utilizator este legat static cu un fisier special denumit **biblioteca pentru import**, generat de un program utilitar care extrage anumite informatii din DLL. Programul utilizator este legat la DLL prin aceasta biblioteca pentru importat, putand fi legat la mai multe biblioteci. La incarcarea in memorie pentru executie, Windows examineaza ce DLL-uri sunt folosite, incarca pe cele care nu se afla deja in memorie si modifica structurile de date din biblioteci pentru a le lega static si pune in corespondenta in spatiul virtual de adrese al programului.

La legarea explicita, in momentul executiei, programul utilizator face un apel explicit pentru legarea la un DLL, apoi face apeluri suplimentrea pentru obtinerea adreselor procedurilor de care are nevoie. La terminare, face un apel final pentru desfacerea legaturii cu DLL-ul. O procedura dintr-un DLL este executata in firul de executie al programului apelant si utilizeaza stiva acestuia pentru variabilele sale locale. Se comporta ca o procedura legata static, diferenta fiind modul cum se face legarea la ea.

**Structura unui modul obiect**

Modulele obiect contin adesea sase parti, dupa cum se poate observa in figura. Prima contine informatii necesare editorul de legaturi pentru identificare, ca numele modulului, lungimea diferitelor parti ale acestuia si uneori data asamblarii.




A doua parte a modulului obiect este o lista de simboluri definite in modul, pe care alte module le pot referi, impreuna cu valorile lor. Programatorul in limbaj de asamblare indica ce simboluri trebuie sa fie declarate ca puncte de intrare (entry points), prin utilizarea unei pseudoinstructiuni, ca PUBLIC.

A treia parte consta dintr-o lista de simboluri utilizate in modul dar care sunt definite in alte module, impreuna cu o lista a instructiunilor masina care folosesc aceste simboluri. Editorul de legaturi utilizeaza aceasta lista pentru a putea sa insereze adresele corecte si instructiunile ce utilizeaza simboluri externe. O procedura poate apela alte proceduri translatate independent prin declararea numelor procedurilor apelate ca fiind externe. Programatorul in limbaj de asamblare indica ce simboluri sunt declarate ca simboluri externe (external simbols) prin utilizarea unei pseudoinstructiuni ca EXTRN.

A patra parte este codul asamblat si constantele. Aceasta parte este singura care va fi incarcata in memorie pentru a fi executata. Celelalte cinci parti vor fi utilizate de editorul de legaturi si apoi eliminate inainte de inceperea executiei.

Cea de-a cincea parte este dictionarul de relocatare. Instructiunile ce contin adrese de memorie trebuie sa aiba adunata o constanta de relocatare. Deoarece editorul de legaturi nu are o modalitate de a spune, prin inspectie, care portiune din cod din partea a patra reprezinta instructiuni masina si care constante, informatia despre adresele care vor fi relocate este furnizata de aceasta tabela. Informatia poate fi o lista explicita de adrese ce vor fi relocate sau o harta in care fiecarui octet ii corespunde un bit cu valoarea 1 sau 0, dupa cum octetul face sau nu parte dintr-o adresa relocatabila.

A sasea parte este o indicatie de sfarsit de modul, o suma de verificare (verificarea paritatii) pentru a descoperi erori de citire a modulului si adresa la care incepe executia.

Cele mai multe editoare de legaturi necesita doua treceri. In prima editorul de legaturi citeste toate modulele obiect si construieste o tabela a lungimilor si numelor modulelor si o tabela de simboluri constand din toate punctele de intrare si referintele externe. In a doua trecere modulele obiect sunt citite, relocate si sunt editate pe rand legaturile.

Un exemplu de utilizare a directivelor PUBLIC si EXTRN:

```
modulul A                          modulul B
        definire s
                                   definire p
        declarare p
                                   declarare
        s
start:  ...                        p proc
        call p                             call s
        ...                                ... end start    end p
```

Simbolul s este alocat in modulul A si folosit in modulul B. Modulul A, care este si programul principal, apeleaza procedura p, care foloseste in calculele sale valoarea simbolului s definit in modulul A. Simbolul p va trebui definit in B ca PUBLIC p, devenind cunoscut in exterior. Pentru a putea referi valoarea s, definita in A, in acelasi modul B vom scrie EXTRN s. In modulul A simbolul s va fi facut accesibil prin PUBLIC s.

**Cautarea automata in biblioteci de module obiect**

Biblioteca este o multime organizata de module obiect elaborate professional care rezolva probleme dintr-un anumit domeniu de aplicatii. Exista doua tipuri de biblioteci:

- subprograme standard ce pot simplifica munca programatorilor.
- subprograme standard ce se adreseaza unei anumit domeniu.

Folosirea acestor biblioteci ofera o foarte mare flexibilitate la schimbarile de structura si pastreaza compatibilitatea programelor de aplicatie proprii, de utilizare generala.

Accesul la modulele obiect din cadrul bibliotecilor se face prin adaugarea la inceputul programului, ca variabila externa, a numelui modulului dorit din bibliotecile disponibile. Este recomandata specificarea la inceputul programului a numelui bibliotecilor dorite, pentru o

mai mare claritate.

Mecanismul de acces la bibliotecile de module obiect devine activ dupa ce sa facut asamblarea si legarea programului, cand pot ramane in ESTAB variabile nesatisfacute. In acest caz editorul de legaturi cauta in bibliotecile specificate variabilele nedefinite. Avand in vedere ca apelul unei variabile dintr-o biblioteca poate declansa un apel in cascada exista posibilitatea ca una din biblioteci sa lipseasca. Atunci variabila va fi semnalata cu eroare de definire in ESTAB.

Timpul de cautare in biblioteci poate fi micsorat prin sortarea prealabila a modulelor obiect componente. Sortarea are la baza alcatuirea de cataloage de chei ce premit gasirea foarte rapida a numelui cautat. Listele de chei sunt rezidente in memoria operativa pe durata operatiunii de legare a modulelor obiect.

**Conducerea procesului de incarcare**

Structurarea programelor sursa se poate realiza pe trei niveluri ierarhice :

- sectiunea
- segmentul
- programul

**Sectiunea:**

Sectiunea - unitate de program independenta alcatuita dintr-o secventa de date, ce include atat definitii cat si instructiuni. Acestea pot fi folosite si in afara sectiunii unde au fost definite si atunci se vor numi simboluri externe si vor fi memorate in dictionarul de legaturi al sectiunii. Mai exact se memoreaza numele si adresele relative fata de inceputul sectiunii in care au fost definite.

Dictionarul de legaturi se generat atunci cand sectiunea este compilata, permitand schimbul de informatii intre diferite sectiunii ale unui program. Informatiile pot reprezenta apeluri la simbolurile externe, sau definitii de simboluri externe.

Unul dintre avantajele oferite de folosirea sectiunilor este acela ca mai multi programatori pot scrie sectiunile uni program, ce pot fi mai apoi compilate separat pentru a obtine modulele obiect ale programului.

**Segmentul**

Segmentul este format din mai multe sectiuni ce au toate legaturile rezolvate. Este un modul obiect relocatabil format din sectiuni puse impreuna. Un segment este definit prin nume si prin adresa de intrare (adresa primei instructiuni executabile).

Pentru a realiza un segment, editorul de legaturi :

- defineste complet dictionarul de legaturi a fiecarei sectiuni : pentru fiecare simbol extern verifica daca exista o intrare intr-un alt dictionar de legaturi si memoreaza adresa acestuia
- aloca segmentului o zona continua de memorie, prin alocarea de zone de memorie continue si succesive tuturor sectiunilor ce alcatuiesc segmentul; astfel se determina adresele relative (la adresa 0 a segmentului ) de incarcare a sectiunilor in memorie;
- relocateaza adresele simbolurilor externe, prin adaugarea la aceste adrese a adresei de incepu a sectiunii unde sunt definite.

**Programul**

Este o structura arborescenta formata din segmente, dintre care unul este segmentu radacina (principal) , celelalte fiind subordonate acestuia. Segmentele pot fi subordonate si unul fata de altul, rapotrurile de subordonare fiind determinate de ordinea de executie a acestora. Segmentele aceluias nivel ierarhic nu comunica intre ele si pot fi executate in paralel.

Segmentarea programului reprezinta procesul de impartire a programului in mai multe segmente, pentru a face posibila incarcarea segmentului radacina in memoria fizica in timpul executiei programului, restul segmentelor putand fi incarcate ulterior. Numarul segmentelor incarcate simultan in memorie depinde de memoria disponibila cat si de relatiile de subordonare dintre segmente.

In timpul generarii programului executabil, editorul de legaturi construieste si tabelul de legaturi asociat ce contine:

- numele segmentului radacina si a celorlalte segmente
- adresele relative ( la adresa 0H a programului ) ale segmentelor
- lungimea segmentului radacina si a celorlalte segmente
- adresa primei instructiuni a programului (adresa de intrare in program)