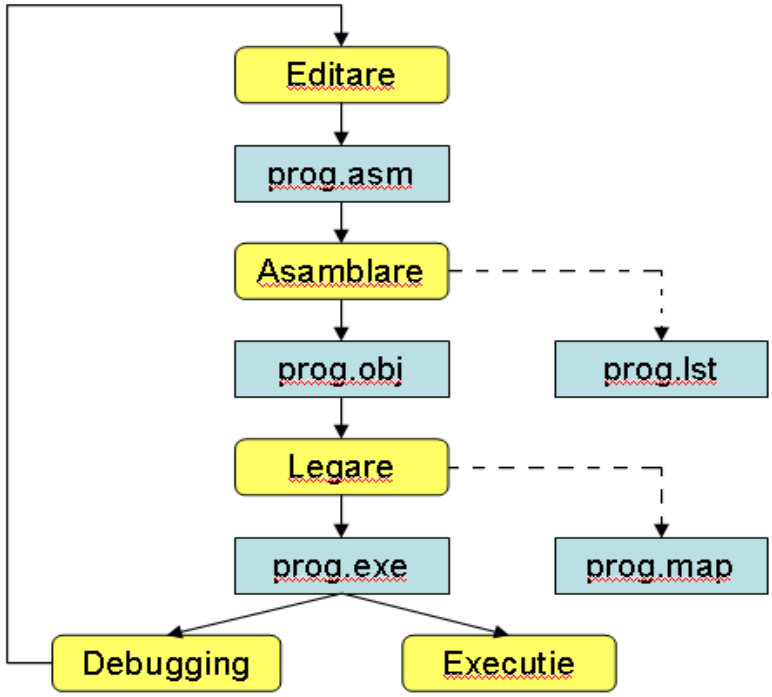
Editarea legaturilor si incarcarea programelor

Pentru a produce fisiere executabile este nevoie de un program de legatura, un linker, care combina fisierele obiect individuale create de asamblor cu biblioteci de legaturi si produce un singur program executabil. De asemenea se folosesc librarii de legaturi (link libraries) pentru intrari si iesiri. De exemplu, asamblorul MASM vine cu un program linker LINK32.EXE pentru legarea programelor pe 32 de biti.

Rolul editorului de legaturi in ciclul Asamblare – Link – Debug poate fi observat pe figura:



Editorul:

- scrie programe noi (.asm)
- face schimbari la cele existente

Asamblorul: (eg ML.XE)

- traduce fisiere (.asm) in fisiere obiect (.obj) in limbaj masina
- poate produce un fisier de listare (.lst) care arata activitatea asamblorului

Linkerul (eg LINK32.EXE)

- combina fisiere obiect (.obj) cu fisiere link libraries (.lib)
- produce fisiere executabile(.exe)
- poate produce un fisier optional(.map)

Debuggerul (eg WINDBG.EXE)

- identifica executia programului
  -> pas cu pas sau
  -> folosind breakpoints
- permite vizualizarea

-> codului sursa (.asm)
-> registrelor
-> memoriei dupa nume si dupa adresa
-> se pot modifica registrele de memorie si continutul memoriei

- permite identificarea erorilor si revenirea in editor pentru a rezolva bug-urile de program

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.

Trecerea de la procedura sursa la modulul obiect reprezinta o modificare de nivel, limbajul sursa si limbajul obiect avand notatii si instructiuni diferite. Procesul de editare de legaturi nu reprezinta insa 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 le 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.

Fiecare procedura sursa este translatata de compilatoare si asambloare ca o entitate separata. Altfel, modificarea unei declaratii intr-o procedura sursa ar necesita ca toate celelalte proceduri sursa sa fie retranslatate. Utilizand tehnica modulelor obiect separate se va retranslata 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 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.

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.

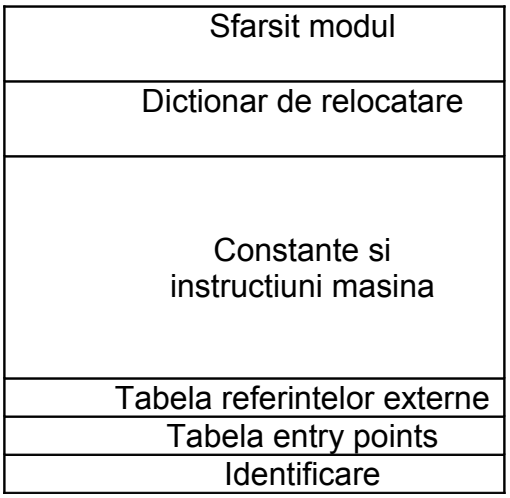
Editorul de legaturi rezolva aceste probleme unind spatiile separate de adresa ale modulelor obiect intr-un singur spatiu liniar de adrese, in urmatorii pasi:

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.

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 este dat in [3]:

```
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.

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

Metoda de editare de legaturi prezentata pana acum presupune legarea tuturor modulelor unui program inainte de executia acestuia. Acesta nu utilizeaza eficient memoria virtuala, de exemplu in cazul procedurilor pentru prelucrarea conditiilor de eroare, care apar rar intr-un program. Legarea dinamica presupune editarea legaturilor procedurilor compilate separat la momentul primului apel al procedurii. Metoda a fost implementata pentru prima data de MULTICS.

Legarea dinamica in MULTICS

Fiecarui program i se asociaza un segment, numit **segment de legaturi,** ce contine un bloc de informatie pentru fiecare procedura ce ar putea fi apelata. Blocul incepe cu un cuvant rezervat pentru adresa virtuala a procedurii, urmat de nume, memorat ca sir de caractere.

Apelurile procedurii in limbajul sursa sunt translatate in instructiuni care adreseaza indirect primul cuvant al blocului de legaturi corespunzator, adica adresa virtuala. Compilatorul completeaza acest spatiu fie cu o adresa invalida, fie cu o anumita configuratie de biti ce forteaza o intrerupere de tip capcana. Atunci editorul de legaturi gaseste sirul de caractere din cuvantul ce urmeaza in bloc dupa adresa invalida si cauta in catalogul de fisiere al utilizatorului o procedura compilata cu acest nume. Procedurii i se atribuie o adresa virtuala, de obicei in segmentul in care se afla, scrisa peste adresa invalida. Instructiunea care a determinat eroarea de editare de legaturi este din nou executata, rularea programului incepand de la locul dinainte de capcana.

Cand este apelata o procedura dintr-un segment diferit, incercarea de a adresa cuvantul invalid va determina tot o intrerupere de tip capcana, procedeul repetandu-se ca in cazul anterior. Apelurile ulterioare la aceeasi procedura vor fi executate fara eroare de legaturi, cuvantul adresat indirect continand acum o adresa virtuala valida. Astfel editorul dinamic de legaturi este invocat doar la primul apel.

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.

Legarea dinamica in UNIX

Mecanismul in UNIX este in pricipiu identic cu DLL-urile din Windows. Poarta denumirea de **biblioteca partajata** (shared library). La fel ca un DLL, ea reprezinta un fisier arhiva ce contine mai multe proceduri sau module de date, prezente in memorie la momentul executiei si care pot fi legate simultan la mai multe procese. Biblioteci partajate sunt biblioteca C standard si o mare parte a codului de conectare la retea.

In UNIX este posibila doar legarea implicita, astfel ca o biblioteca partajata este alcatuita din doua parti: o biblioteca gazda (host library), legata static cu fisierul executabil si una tinta (target library), apelata in momentul executiei.

Rezumat

Modulele obiect concepute separat pot fi legate impreuna pentru a fi rulate din acelasi spatiu de memorie sub forma unui program binar executabil. Aceasta operatie este realizata de editorul de legaturi. Sarcina sa este legarea numelor la adrese virtuale si fizice si relocarea modulelor, adica incarcarea la adrese de memorie stabilite ulterior creatiei, de unde sa se lanseze si sa produca rezultate identice in urma rularii. Prin legare dinamica anumite proceduri nu sunt legate decat in momentul apelului lor efectiv. DLL-urile din Windows si bibliotecile partajate din UNIX utilizeaza legarea dinamica.

Bibliografie:

1.. Tanenbaum, Andrew S - Organizarea Structurata a Calculatoarelor (editia a IV-a)
2. Stancescu, Stefan – Note de curs
3. Somnea Dan, Vladut Teodor – Programarea in Assembler, Ed. Tehnica, 1992