Továbbfejlesztés:

A csatoló paraméterként kapja meg a felhívandó eljárás nevét.

A csatoló program hoz létre egy csatoló táblázatot.

A csatoló a táblázatban ellenőrzi, hogy csatolva van-e a kívánt eljárás. Ha nincs, akkor elvégzi a csatolást, ha pedig csatolva van, akkor közvetlenül meghívja az eljárást.

Menü vezérelt rendszer esetében, ha a kiválasztott menü elemhez tartozó szövegből generálni lehet az – esetleg csatolandó, majd – végrehajtandó eljárás nevét és az eljárást tartalmazó file nevét, akkor a program bővítését, javítását a megfelelő file-ok hozzáadásával vagy cseréjével és a menü szöveg file-jának cseréjével akár üzemelés közben is elvégezhetjük.

A menü szövegének újraolvasása után már a bővített vagy javított rendszer funkciói érhetők el.

A csatolt program törlése: a törlendő eljárás csatoló blokkjában a **`CIM`** visszaállítása – illetve a csatoló tábla törlése – után az eljárás törölhető.

Szokásos, hogy egy dinamikusan szerkesztendő eljáráshoz tartozik egy számláló, melynek értéke a csatolás előtt **`0`**.
A hívó program először „bejelenti” az igényét az eljárásra. Ha a számláló **`0`**, akkor megtörténik a csatolás, és mindenképpen: számláló**`++`**.
Ha a továbbiakban már nem igényli az eljárást, akkor „elengedi”: számláló**`--`**.
Ha a számláló értéke **`0`**, akkor senki sem igényli az eljárást, tehát törölhető.

Cím hozzárendelés (binding)

Időosztásos (time sharing) rendszer.

Egy program elindításakor többek között a következő feladatokat kell végrehajtani:

betöltés – indítás – felfüggesztés – kimentés

a program folytatásakor:

visszamentés – futtatás – felfüggesztés – kimentés

Általában nem biztosítható, hogy a visszamentett program a memóriának ugyanarra a területére kerüljön vissza, ahol korábban futott!

Ha pl. a programunk egy **`JMP L`** ugró utasítást tartalmaz, akkor **`L`**-hez valamikor hozzá kell rendelnünk egy konkrét fizikai címet – binding = (cím) hozzárendelés.

A cím hozzárendelés különböző időpontokban történhet:

Program írásakor (gépi kódú programozás): Manapság már ritka. Fellelhető a szegmens definícióban alkalmazható **`AT kifejezés`** kombinációs típusban.

Fordításkor: pl. szintén az **`AT kifejezés`** kombinációs típussal kapcsolatban, ha a szegmensben szimbolikus címzést használnak. Általánosan használt volt, amikor még nem különült el a fordítás és szerkesztés folyamata.

Szerkesztéskor: Ma ez a cím hozzárendelés egyik legelterjedtebb módja. Általában ez valósul meg az assembly programokban is.

Betöltéskor: Ekkor a betöltő átveszi a szerkesztő feladatainak egy részét, pl. a **FAR** típusú cím konstansok értékének kiszámolását, hiszen ezek értéke a betöltés helyétől függ. A program betöltése valamivel hosszabb időt vesz igénybe, előnye viszont, hogy a betöltési cím tetszőleges lehet.

A címzéshez használt bázis regiszter kitöltésekor: Ez a módszer már biztosítja az időosztásos rendszerben futtatható alakú programok elkészítését, de a **FAR** címek nagy problémát jelentenek, mert a program áthelyezésekor módosítandók a **FAR** címek.

Megfelelően kialakított hardver: pl. a **Motorola 680x0** processzor család rendelkezik egy **program bázis regiszter**rel. Minden utasítás tartalmaz egy bitet, hogy a benne szereplő hivatkozás módosítandó-e a program bázis regiszterrel.

Az utasítás végrehajtásakor: Ehhez a cím hozzárendelést két lépésben valósítják meg. Először a szimbolikus címet virtuálissá alakítják (történhet pl. fordítási vagy szerkesztési időben), majd a virtuális címet fizikai címmé. Ez utóbbi a lap (page) tábla kitöltésekor, tehát gyakran az utasítás végrehajtásának megkezdése után történik. Amikor ugyanis a hivatkozott lap még nincs a memóriában, akkor azt az utasítás végrehajtása előtt be kell tölteni, a lap táblának a megfelelő elemét módosítani kell (cím hozzárendelés), majd ezután történhet a megkezdett utasítás végrehajtásának befejezése.

Programok hangolása

Egy programozó néhány évig egy nagyobb feladaton dolgozva havi átlagban csak kb. 100-200 (!!) ellenőrzött utasítást ír, függetlenül az alkalmazott programozási nyelvtől.

Két operációs rendszer	MULTIX	TSS/67
Alkalmazott programozási nyelv	95%-ban PL/I	assembly
Program lista	3.000 oldal	30.000 oldal
Programozók száma	50	300
Költség	10 millió $	50 millió $

Egy **PL/I** utasítás 5-10 assembly utasításnak felel meg.

Sokkal gyorsabb **TSS/67** ?

Irodalmi adatok alapján azt lehet mondani, hogy (hangolás előtti) nagyobb programok 1%-a felelős a program futási idejének kb. 50%-áért, 10%-a a 90%-áért.

Program hangoláson azt a folyamatot értjük, amikor megállapítjuk a kritikus részeket, és ezek gyorsításával az egész program futását felgyorsítjuk.

A kritikus részek felderítése: pl. idő szerinti megszakítások címének könyvelésével.

Tételezzük fel, hogy ugyanannak a feladatnak a megoldásához

- assemblyben 5-ször annyi utasításra (és időre) van szükség, mint probléma orientált nyelv esetén
- az elkészült program 3-szor olyan gyors.

A probléma orientált nyelven készült változatának kritikus 10%-át assemblyben újra programozzuk.

A költségek és futási idők alakulása:

	programozó év	futási idő
Assembly nyelv	**50**	**333**
Probléma orientált nyelv	**10**	**1000**
Hangolás előtt		
a kritikus 10%	**1**	**900**
a többi 90%	**9**	**100**
Összesen	**10**	**1000**
Hangolás után		
a kritikus 10%	**1+5**	**300**
a többi 90%	**9**	**100**
Összesen	**15**	**400**

- A program probléma orientált nyelven történő elkészítésének és hangolásának ideje (és költsége) kb. harmada (15 programozó év) annak, mintha az egészet assemblyben készítenénk (50 programozó év),
- a sebessége csak 20%-kal gyengébb (333 helyett 400).