10. Tétel

a. Adatszegmensben megadott szöveg kiírása, képernyő-memória kezelése:

Az adatszegmens:

Az adatszegmens használatához deklarálnunk kell azt a programban a .DATA megadásával.

```
.MODEL SMALL
.STACK
.DATA
  adat DB 65
.CODE
main PROC
  MOV AX, Dgroup          ;Adatszegmens helyének lekérdezése
  MOV DS, AX              ;DS beállítása, hogy az adatszegmensre mutasson
  MOV DL, adat            ;Az adat betöltése DL-be
  CALL write_char         ;Karakter kiírása
  MOV AH,4Ch              ;Kilépés
  INT 21h
main endp
```

Mivel előre nem lehet megmondani, hogy a memóriában hova kerül az adatszegmens, ezért szükséges egy DGROUP változó, melynek értékét a betöltő program állítja be. Ezt az értéket kell betölteni a DS regiszterbe. Mivel a memóriából közvetlenül nem lehet adatot tölteni a DS-be, azért közbe kell iktatni az AX regisztert. Adatszegmens használatakor a DS beállítása kötelező.

Képernyő-memória kezelése

A képernyőn megjelenő kép nem más, mint a képernyő-memória leképezése. Ha a memóriába beírunk egy karaktert, akkor az azonnal megjelenik a monitoron. A videó-memória szegmenscíme 0B800h, ez a képernyő első karakterpozíciójának felel meg. Minden karakterpozíciót egy 16 bites szóval jellemezhetünk. Az alsó byte tartalmazza a karakterkódot, a felső a megjelenítő attribútumot.

- Memóriaelem: 0-7 bit karakterkód, 7-15 bit attribútum
- Attribútum: 0-3 bit karakterszín, 4-6 bit háttérszín, 7 bit villogás. A színek RGB szerint tevődnek össze, a 3. bit a világosságbit.

Az attribútum értékének kiszámítása: 128*villogás + 16* háttérszín + karakterszín. Az AX regiszter használata esetén az AL-be kerül a karakterkód és AH-ba az attribútumot.
A képernyőn egy karakter pozíciójának offset címe kiszámítható: 2*((sorszám-1)*sorhossz + karakterpozíció-1). A kettővel szorzás azért kell, mert 2 byte-ot foglal el. Így egy 80 karakter/sor és 50 soros képernyő közepének offset címe 3920.

```
.MODEL SMALL
.STACK
.CODE
main proc
  MOV AX,0B800h           ; Képernyő-memória szegmenscíme ES-be
  MOV ES,AX
  MOV DI,3920             ;A pozíció offset címe
  MOV AL,"*"              ;Kiírandó karakter
  MOV AH, 16*7+4          ;Színkód: szürke háttér, piros karakter
  MOV ES:[DI],AX          ;Karakter beírása a képernyő-memóriába
```

```
  MOV AH,4Ch                ;Kilépés
  INT 21h
main endp
END main
```

A képernyő kezdő karaktere a bal felső sarokban van, így az X tengely jobbra, az Y pedig lefelé növekszik. Az első karakter (0,0) offset címe a memóriában viszont 0.

b. A Fordítóprogram szerkezete:

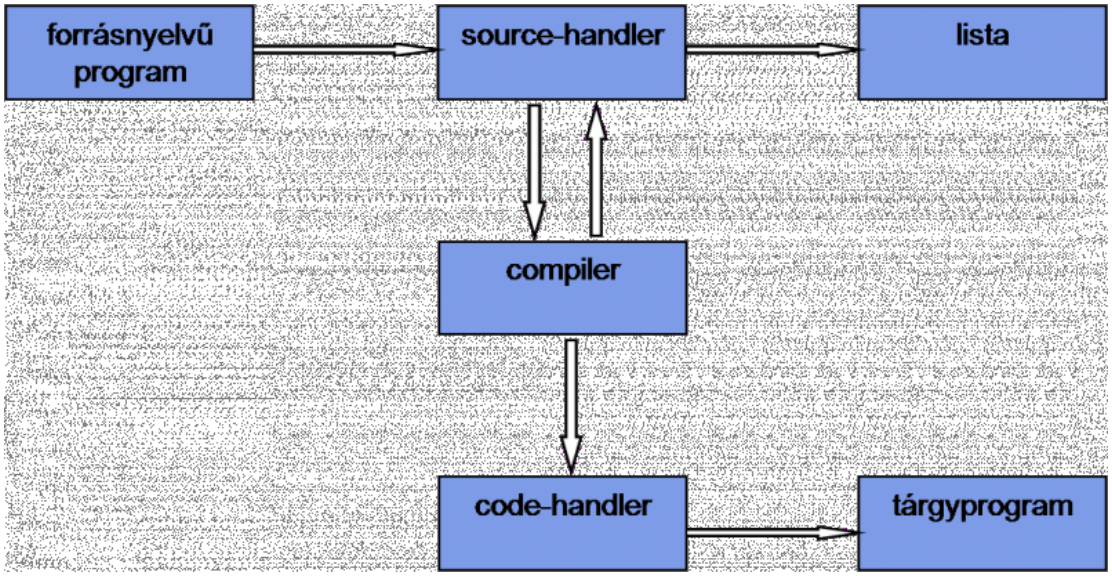
Fordítóprogramok

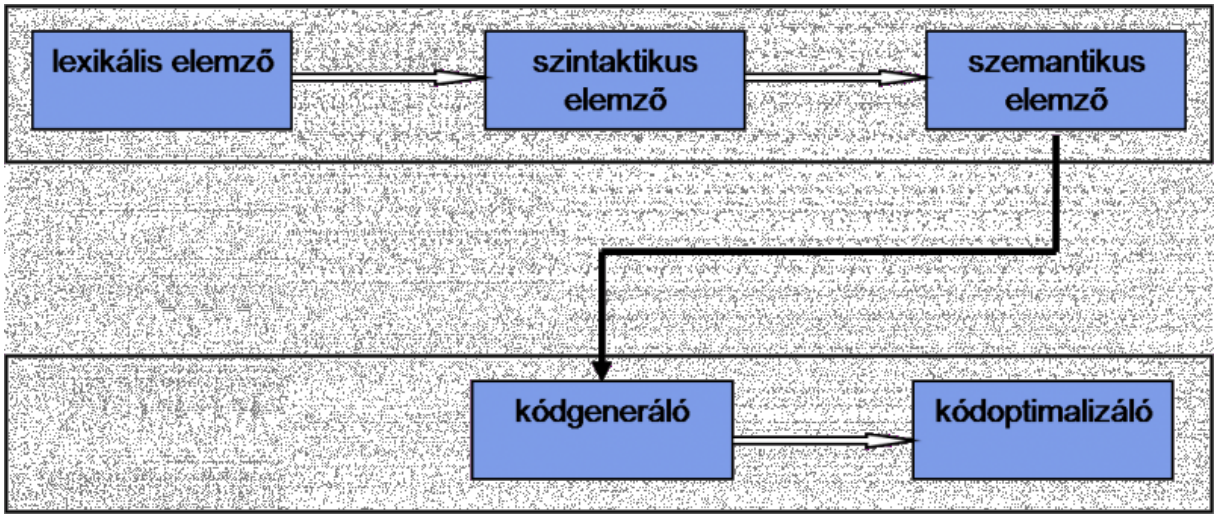
A fordítóprogram (angolul compiler) olyan számítógépes program, amely valamely programozási nyelven írt programot képes egy másik programozási nyelvre lefordítani. A fordítóprogramok általánosan forrásnyelvi szövegből állítanak elő tárgykódot. A fordítóprogramok feladata, hogy nyelvek közti konverziót hajtsanak végre. A fordítóprogram a forrásprogram beolvasása után elvégzi a lexikális, szintaktikus és szemantikus elemzést, előállítja a szintaxis fát, generálja, majd optimalizálja a tárgykódot.

Szerkezet

Használjuk a következő jelölést: program (bemenet) (kimenet). A fordítóprogram struktúrája:

- Compiler (forrásnyelvű program) (tárgyprogram, lista)
- Forrás olvasása a háttértárról: input-handler (forrásnyelvű program) (kódsorozat)
- A listák készítése: output-handler (forrásnyelvű program, hibák) (lista)
    - Mindkét program a forrásból dolgozik, ezért össze lehet vonni: source-handler (forrásnyelvű program, hibák) (kódsorozat, lista)
- Tárgykód mentése: code-handler (tárgykód) (tárgyprogram)
- Compiler felépítése





A compiler feladata:

- Analízis:
  - Lexikális elemző (karaktersorozat) (szimbólumsorozat, lexikális hibák)
    - Szimbólumtábla létrehozása (szimbólum típusa, szimbólum címe)
    - Szóköz karakterek és kommentek kiszűrése
    - Többsoros utasítások összeállítása
  - Szintaktikus elemző (szimbólumsorozat) (szintaktikusan elemzett program, hibák)
    - Szimbólumok helyének ellenőrzése
    - Ellenőrzi, hogy a szimbólumok sorrendje megfelel-e a programnyelv szabályainak
    - Szintaxisfa előállítása
  - Szemantikus elemző (szintaktikusan elemzett program) (analizált program, hibák)
    - Konstansok, változók érték- és típusellenőrzése
    - Aritmetikai kifejezések ellenőrzése
- Szintézis:
  - Kódgenerátor (analizált program) (tárgykód)
  - Kódoptimalizáló (tárgykód) (tárgykód)

c. A programozási nyelvek csoportosítása:

Programozás során olyan nyelvet válasszunk, amely tartalmaz olyan konstrukciókat (utasításokat), amely a feladat megfogalmazásában is szerepel.
A programozási nyelvek csoportosítása:

- gépi kód,
- assembly nyelv,
- magasszintű nyelvek,
  - imperatív nyelvek(PHP,PASCAL): Ezen nyelvek közös jellemzője, hogy a program fejlesztése értékadó utasítások megfelelő sorrendben történő kiadására koncentrálódik. Az értékadó utasítás baloldalán egy változó áll, a jobb oldalán pedig egy megfelelő típusú kifejezés. A szelekció (elágazás) csak azt a célt szolgálja, hogy bizonyos értékadó utasításokat csak adott esetben kell végrehajtani. A ciklusok pedig azért vannak, hogy az értékadó utasításokat többször is végrehajthassunk. Az értékadó utasítások során részeredményeket számolunk ki – végül megkapjuk a keresett végeredményt.
  - deklaratív nyelvek,
    - funkcionális nyelvek (Miranda, SML, LISP): A funkcionális nyelveken a kiszámolandó kifejezést adjuk meg, megfelelő mennyiségű bemenő adattal. A programozó munkája a kifejezés kiszámításának leírására szolgál. A program futása közben egyszerűen kiszámítja a szóban forgó kifejezést.

Egy funkcionális nyelvben nincs változó, általában nincs ciklus (helyette rekurzió van). Értékadó utasítás sincs, csak függvény visszatérési értékének megadása létezik. A funkcionális nyelvek tipikus felhasználási területének a természettudományos alkalmazások tekinthetőek.

  - logikai nyelvek (Prolog): Az ilyen jellegű nyelveken tényeket fogalmazunk meg, és logikai állításokat írunk le. A program ezen kívül egyetlen logikai kifejezést is tartalmaz, melynek értékét a programozási nyelv a beépített kiértékelő algoritmusa segítségével, a tények és szabályok figyelembevételével meghatároz.

    A logikai nyelvek tipikus felhasználási területe a szakértői rendszerek létrehozásához kapcsolódik.

- objektumelvű nyelvek (java, C++, Delphi): Az OOP nyelveken a program működése egymással kölcsönhatásban álló objektumok működését jelenti. Az objektumok egymás műveleteit aktiválják, melyeket interface-ek írnak le. Ha egy művelet nem végrehajtható, akkor az adott objektum a hívó félnek szabványos módon (kivételkezelés) jelzi a probléma pontos okát.