6. tétel

a. Feltételes vezérlésátadás, a CPU fontosabb regiszterei és alkalmazásuk

Az assembly nyelv egyik szegényes tulajdonsága a vezérlési szerkezetek hiánya. Lényegében csak az alábbi vezérlési szerkezetek találhatóak meg:

- Szekvencia: a program utasításainak végrehajtása a memóriabeli sorrend alapján történik.
- Feltétel nélküli vezérlésátadás (ugró utasítás): a program folytatása egy másik memóriabeli pontra tevődik át, majd attól a ponttól kezdve a végrehajtás újra szekvenciális.
- Feltételes vezérlésátadás (feltételes ugró utasítás): mint az egyszerű ugró utasítás, de az ugrást csak akkor kell végrehajtani, ha az előírt feltétel teljesül.
- Visszatérés az ugró utasítást követő utasításra (azon helyre, ahonnan az ugrás történt).

Regiszterek

A regiszterek méretével jellemezhető egy CPU, azaz lehet 8, 16, 32 stb bites. A nagyobb méret (elméletileg) gyorsabb processzort jelent, de ez csak egyre nagyobb adatmennyiségeknél igaz. (16 bites esetben értendőek az alább felsorolt regiszterek.)

1. *Általános regiszterek*

- Akkumulátorregiszter
  Jelölése: **AX**
  Alsó bitje: **AL**
  Felső bitje: **AH**
  Szerepe van a szorzás, osztás és I/O utasításoknál.
- Bázisregiszter
  Jelölése **BX**
  Alsó bitje: **BL**
  Felső bitje: **BH**
  Szorzás és osztástól eltekintve minden művelethez használható, általában az adatszegmensben tárolt adatok báziscímét tartalmazza.
- Számlálóregiszter
  Jelölése: **CX**
  Alsó bitje: **CL**
  Felső bitje: **CH**
  Szorzás és osztás kivételével minden művelethez használható, általában ciklus, léptető, forgató és sztring utasítások ciklusszámlálója.
- Adatregiszter
  Jelölése: **DX**
  Alsó bitje: **DL**
  Felső bitje: **DH**
  Minden művelethez használható, de fontos szerepe van a szorzás, osztás és I/O műveletekben.

2. Vezérlő regiszterek

- Forrás cím
  Jelölése: **SI**
  A forrásadat indexelt címzésére. Szorzás és osztás kivételével minden műveletnél használható.
- Cél cím
  Jelölése: **DI**
  A céladat indexelt címzésére. Kitüntetett szerepe van a sztring műveletek végrehajtásában. Az SI regiszterrel együtt valósítható meg az indirekt és indexelt címzés. Szorzás és osztás kivételével minden műveletnél használható.
- Stack mutató
  Jelölése: **SP**

A verembe utolsóként beírt elem címe. A mutató értéke a stack műveleteknek megfelelően automatikusan változik.

- Bázis mutató
  Jelölése: **BP**
  A verem indexelt címzéséhez. Használható a stack-szegmens indirekt és indexelt címzésére. Más műveletben nem javasolt a használata.
- Utasításmutató
  Jelölése: **IP**
  A végrehajtandó utasítás címét tartalmazza, mindig a következő utasításra mutat. Az utasítás beolvasása közben az IP az utasítás hosszával automatikusan növekszik.

3. *Szegmensregiszterek*

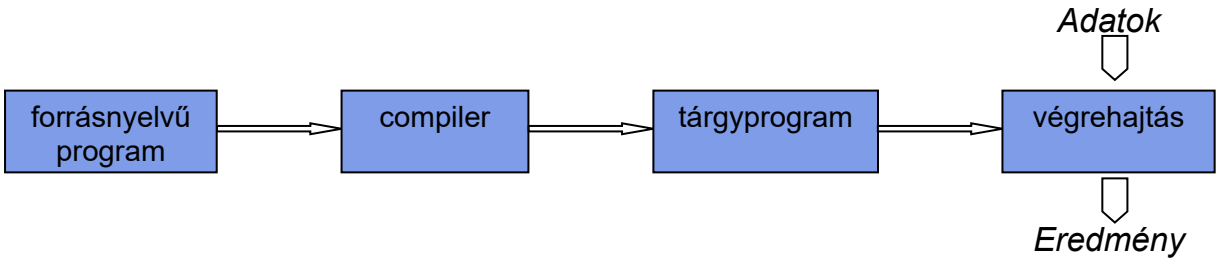
Ezek a regiszterek tárolják a különböző funkciókhoz használt memória- és szegmenscímeket.

- Kódszegmens
  Jelölése: **CS**
  Az utasítások címzéséhez szükséges, az éppen futó programmodul báziscímét tartalmazza. Minden utasításbetöltés használja. Tartalma csak vezérlésátadással módosulhat.
- Veremszegmens
  Jelölése: **SS**
  A verem címzéséhez használatos, a stack-ként használt memóriaterület báziscímét tartalmazza.
- Adatszegmens
  Jelölése: **DS**
  Az adatterület címzéséhez kell, az adatszegmens bázis címét tartalmazza.

b. Magas szintű programnyelvek fordítása, compiler, interpreter

Magas szintű programnyelv fordítása

Ha a forrásnyelv egy magas szintű nyelv, akkor a forráskód és a gépi kód között jelentős a különbség.



Ha ***P*** a forrásnyelvű, ***Q*** a tárgynyelvű programot, ***T*** a fordítás transzformációját jelöli, akkor a fordítási folyamat:

$$Q = T(P)$$

ha $T=T_1T_2...T_n$, akkor:

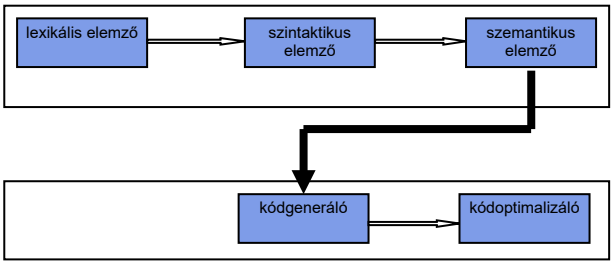
$$P_{n-1} = T_n(P),$$
$$P_{n-2} = T_{n-1}(P_{n-1}),$$
$$...$$
$$P_1 = T_2(P_2),$$
$$Q = T_1(P_1),$$

Compiler

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.

A compiler feladata:

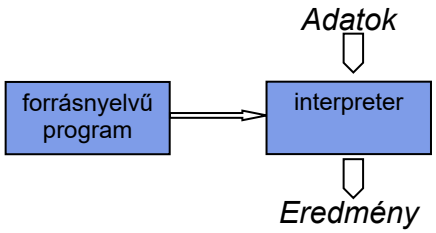
- Analízis: a forrásnyelvű program karaktersorozatát részekre bontja, még a szintézis az egyes részeknek megfelelő tárgykódokból építi fel a program teljes tárgykódját.
    - Lexikális elemző (karaktersorozat) (szimbólumsorozat, lexikális hibák): a karaktersorozatban meghatározza az egyes szimbolikus egységeket, a konstansokat, változókat, kulcs szavakat és operátorokat. A karaktersorozatból szimbólumsorozatot készít, ki kell szűrnie a szóköz karaktereket a kommenteket, mivel ezek tárgykódot nem adnak. A magas szintű programnyelvek utasításai általában több sorban írhatók a lexikális elemző feladata, egy több sorba írt utasítás összeállítása is.
        - Szimbólumtábla létrehozása (szimbólum típusa, szimbólum címe)
        - Szóköz karakterek és kommentek kiszűrése
        - Többsoros utasítások összeállítása
    - Szintaktikus elemző (szimbólumsorozat) (szintaktikusan elemzett program, hibák): a program struktúrájának a felismerése. A szintaktikus elemző működésének az eredménye lehet például az elemzett program szintaxisfája vagy ezzel ekvivalens struktúra.
        - Szimbólumok helyének ellenőrzése
        - Ellenőrzi, hogy a szimbólumok sorrendje megfelel-e a programnyelv szabályainak
        - Szintaxisfa előállítása
    - Szemantikus elemző (szintaktikusan elemzett program) (analizált program, hibák): feladata bizonyos szemantikai jellegű tulajdonságok vizsgálata. A szemantikus elemző feladat például az a+b kifejezés elemzésekor az, hogy az összeadás műveletének a felismerésekor megvizsgálja, hogy az „a” és a „b” változók deklarálva vannak-e, azonos típusúak-e és hogy van-e értékük.
        - Konstansok, változók érték- és típusellenőrzése
        - Aritmetikai kifejezések ellenőrzése
- Szintézis:
    - Kódgenerátor (analizált program) (tárgykód): a tárgykód gépfüggő, generációs rendszertől függő, a leggyakrabban assembly vagy gépi kódú program.
    - Kódoptimalizáló (tárgykód) (tárgykód): A kódoptimalizálás a legegyszerűbb esetben a tárgykódban lévő azonos programrészek felfedezését és egy alprogramba való helyezését vagy a hurkok ciklus változásától független részeinek megkeresését és a hurkon kívül való elhelyezését jelenti. Egy jó kódoptimalizálónak jobb és hatékonyabb programot kell előállítania, mint amit egy gyakorlott programozó tud elkészíteni.



Interpreter

Az interpreter hardveres, vagy virtuális gép, mely értelmezni képes a magas szintű nyelvet, vagyis melynek gépi kódja a magas szintű nyelv. Ez egy kétszintű gép, melynek az alsó szintje a hardver, a felső az értelmező és futtató rendszer programja. Feladatai: beolvassa a következő utasítást, és eldönti, hogy az utasításkészlet melyik utasítása.

- Ellenőrzi, hogy az adott utasítás szintaktikailag helyes-e, az átadott paraméterek típusa, esetleg mérete megfelel-e az utasítás kívánalmainak
- Ha hibátlan, akkor meghívja az utasításhoz tartozó előre elkészített kódrészletet
- Figyeli, hogy a végrehajtás során nem jön-e létre valamilyen hiba. Ha hibára fut a rendszer, akkor hibakódot kell generálnia. Ha hibátlan a végrehajtás, akkor veszi a következő utasítást
- fordítási és futási időpont egybeesik,
- nincs tárgyprogram,
- használhatnak közbülső nyelvet.



c. Chomsky-féle nyelvosztályok

Chomsky négy nyelvosztályt definiált. Ezeket számokkal jelölte, így van **0**-ás, **1**-es, **2**-es és **3**-as nyelvosztály.

Az egyes nyelvosztályokban a helyettesítési szabályok alakjára vonatkozóan Chomsky az osztály sorszámának növekedésével egyre szigorúbb megkötéseket írt elő. Ezek szerint legkevésbé kötött nyelvtanú

A **0-**ás nyelvosztály. Itt semmiféle külön megkötés nincsen. Természetesen az az előírás, hogy minden produkciós szabály baloldala tartalmazzon legalább egy nemterminális szimbólumot, – mint minden grammatikánál, – itt is érvényes.

A **3**-as nyelvosztály nyelvtanaiban csak kétféle szabálytípus engedélyezett. Ezek:

$$A \to a \text{ illetve } A \to aB$$

A helyettesítési szabály baloldala mindig egyetlen nemterminális, jobb oldalán pedig vagy egyetlen terminális szimbólum, vagy egyetlen terminális és egyetlen nemterminális szimbólumból álló jelsorozat.

Az ilyen alakú grammatikákat **reguláris**, pontosabban jobbreguláris nyelvtanoknak nevezzük. Amennyiben a második szabálytípusnál a két jobb-oldali szimbólum sorrendjét felcseréljük, vagyis a nemterminális szimbólumot követi a terminális szimbólum, akkor balreguláris grammatikáról beszélünk.

Ezek a nyelvtanok generálják a reguláris nyelveket. Itt a különbségtételnek nincs értelme, hiszen – mint azt később igazoljuk – a kétféle nyelvtan ugyanazt a halmazt generálja.

A **2**-es nyelvosztály helyettesítési szabályainak alakja:

$$A \to \alpha$$

ahol α tetszőleges, mind terminális mind nemterminális szimbólumokat tartalmaz-ható jelsorozat. A szabály baloldala itt is egyetlen, szükségképpen nemterminális szimbólum. Ezt úgy interpretálhatjuk, hogy egy adott A nemterminális szimbólum mindig helyettesíthető az α jelsorozattal, függetlenül attól mi a nemterminális környezete.

Minthogy a szabályok alkalmazása a környezettől, kontextustól független, ennek a nyelvosztálynak a nyelveit **környezetfüggetlen**, vagy angol rövidítésük alapján **CF** (*Context Free*) nyelveknek nevezzük.

Az **1**-es nyelvosztály helyettesítési szabályainak korlátait kétféle módon is jellemezhetjük. Az első megadási mód szerint a levezetési szabályok alakja:

$$\beta A \gamma \to \beta \alpha \gamma$$

amit úgy interpretálhatunk, hogy az $A \to \alpha$ szabály csakis a β-γ környezetben alkalmazható. Ez az értelmezés magyarázza ezen nyelvosztály elnevezését, ezek a nyelvek a **környezetfüggő**, vagy angol rövidítésük alapján **CS** (*Context Sensitive*) nyelvek.

Az egyes típusok közötti összefüggés: {reguláris nyelvek} ⊆ {környezetfüggetlen nyelvek} ⊆ {környezetfüggő nyelvek} ⊆ {általános nyelvek}. Hasonló összefüggés igaz a grammatikákra is: {reguláris grammatikák} ⊂ {környezetfüggetlen grammatikák} ⊂ {környezetfüggő grammatikák} ⊂ {általános grammatikák}.