6.) Tétel:

Feltételes vezérlésátadás, a CPU fontosabb regiszterei és alkalmazásuk(a.):

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.
- Feltétlen 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).

Ezekből kellett összerakni a programot. Egy egyszerű elágazást ennek megfelelően az alábbi módon kellett kódolni:

HA feltétel AKKOR… feltétel kiértékelése …
Ut1UGRÁS_HA feltétel HAMIS CIMKE1-re
Ut2UT1
KÜLÖNBENUT2
Ut3UGRÁS CIMKE2-re
Ut4@CIMKE1:
HVÉGEUT3
folytatásUT4
@CIMKE2:
folytatás

A fentiből talán sejthető, hogy az assembly nyelvű programból kibogozni, hogy itt valójában feltételes elágazás történt – nem egyszerű. Hasonló problémákkal jár a ciklusok megtervezése és kódolása is – különösen az egymásba ágyazott ciklusok esete.

A 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.

Általános regiszterek

16 bites esetben értendőek az itt felsorolt 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.

Vezérlő regiszterek

16 bites esetben értendőek az itt felsorolt 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.

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

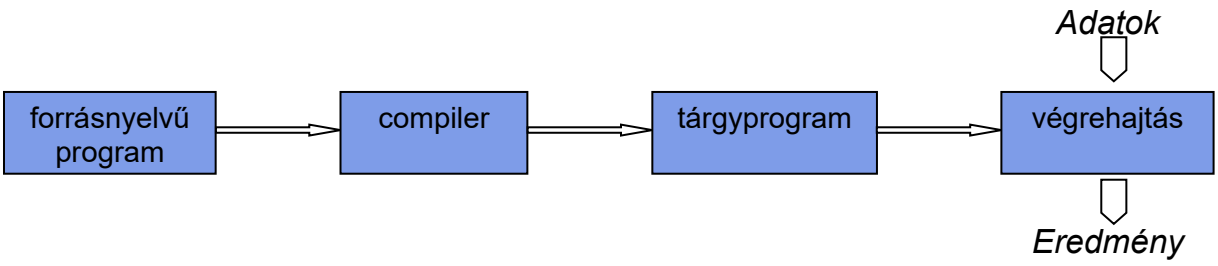
Jelölése: **SS**
A verem címzéséhez használatos, a stack-ként használt memóriaterület báziscímét tartalmazza.

Adatszegmens

Jelölése: **DS**
Az adatterület címzéséhez kell, az adatszegmens bázis címét tartalmazza.

Magasszintű programnyelvek fordítása, compiler,interpreter(b.):

Magasszintű programnyelv fordítása:
Ha a forrásnyelv egy magasszintű nyelv, akkor a forráskód és a gépi kód között jelentős a különbség.



Fogalmak:

- fordítási idő,
- futási idő,
- közbülső programforma (többmenetes fordítás) : 4. tételben tárgyalva.

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

Q = ***T(P)***

ha T=T1T2...Tn, akkor:

Pn-1 = Tn(P),
Pn-2 = Tn-1(Pn-1),
...
P1 = T2(P2),
Q = T1(P1),

Compiler és interpreter feladatai

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:

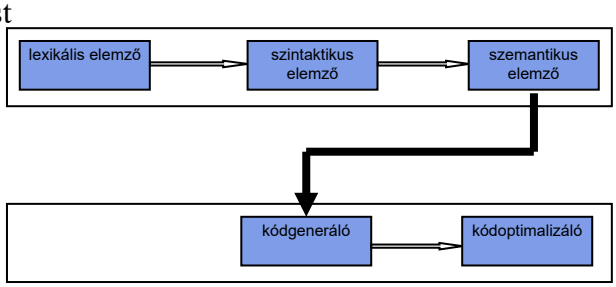
- Analízis: a forrásnyelvű program karaktersorozatát részekre bontja, még a szintézis az egyes részeknek megfelelő tárgykódokból építi fel a program teljes tárgykódját.
    - Lexikális elemző (karaktersorozat) (szimbólumsorozat, lexikális hibák): a karaktersorozatban meghatározza az egyes szimbolikus 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 magasszintű 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 magasszintű nyelv. Ez egy két szintű 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



Chomsky-féle nyelvosztályok(c.)

Adott egy G=(T,N,S,P) grammatika, és
$\alpha, \beta, \gamma \in (T \cup N)^*$ mondatformák, lehetnek ε értékűek is
$\omega \in (T \cup N)^+$ mondatforma, de nem lehet ε
A,B ∈ T terminális jelek
a, b ∈ N nem terminális jelek

Minden típusra igaz, hogy akkor hívunk egy nyelvet az adott típusúnak, ha van olyan grammatika, ami az adott nyelvet generálja.
Reguláris (3-típusú) nyelvek: itt egy szabály kétféle alakú lehet: $A \rightarrow a$, vagy $A \rightarrow aB$.
Környezetfüggetlen (2-típusú) nyelvek: egy helyettesítési szabály nem függ a környezetétől, sőt, környezete sem lehet, azaz $A \rightarrow \omega$, és megengedett az $S \rightarrow \varepsilon$.
Környezetfüggő (1-típusú) nyelvek: egy helyettesítési szabály csak bizonyos környezetben alkalmazható, azaz $\beta A \gamma \rightarrow \beta \omega \gamma$, és megengedett az $S \rightarrow \varepsilon$.

Általános (0-típusú) nyelvek: helyettesítési szabályokra nincs megszorítás, azaz $\beta A\gamma \rightarrow \alpha$.

Az egyes típusok közötti összefüggés: {reguláris nyelvek} $\subseteq$ {környezetfüggetlen nyelvek} $\subseteq$ {környezetfüggő nyelvek} $\subseteq$ {általános nyelvek}. Hasonló összefüggés igaz a grammatikákra is: {reguláris grammatikák} $\subset$ {környezetfüggetlen grammatikák} $\subset$ {környezetfüggő grammatikák} $\subset$ {általános grammatikák}.