

5. tétel

a. Konstansok definiálása, STACK szerepe a vezérlésátadásban

Konstansok

A konstansok általában négyféle számrendszerben is megadhatók: binárisan, oktálisan, decimálisan és hexadecimálisan. Assembly esetén a szám után írt betűvel lehet jelezni az egyes számrendszereket:

- bináris: B vagy Y betű
- oktális: O vagy Q betű
- decimális: D vagy T betű
- hexadecimális: H – hogy a változókkal ne keverjük össze, a szám elé „vezető 0” kerül, pl.: 0FFh

Alapértelmezett számrendszer a decimális. Átállítása a **.RADIX n** (n a számrendszer alapja) direktíva alkalmazásával történik, ekkor a számok végéről az adott betű elhagyható (pl.: .RADIX 16 esetén a H).

A karakter-konstansok megadásánál használhatjuk a karakter ASCII kódja, de magát a karaktert is. (pl.: MOV AL, „A” MOV AL, 65)

STACK

A stack (verem) egy olyan – az operációs rendszer által kijelölt memóriaterület – ahova közvetlen utasításokkal lehet adatot elmenteni, és onnan visszaolvasni. Itt átmeneti adatok tárolhatóak, ilyen lehet szubrutin hívásakor a visszatérési cím, vagy egyes függvények, eljárások paraméterei. Az operációs rendszer a megszakításokkor ide menti a futáshoz szükséges regiszterek tartalmát.

A stack egy LIFO (last in, first out) szervezésű memória. Amikor valami bekerül a verembe, a veremmutató (a verem tetejére, az utoljára betett adat memóriacímére mutat) értéke kettővel csökken, ha valami kikerül onnan, kettővel nő. Ez a fajta memóriaszervezés lehetővé teszi szubrutinok hívását, tetszőleges mélységig.

A CALL szubrutin hívás elmenti a stack tetejére a következő (azaz a CALL utáni) utasítás címét, és ezt a címet tölti be az IP-be a RET utasítás. A verem tetején tárolt adat teljes címe az SS:SP regiszterpárban található.

Assemblyben a PUSH utasítással lehet adatot rakni a verembe, ez kettővel csökkenti az SP veremmutató értékét. A POP utasítással lehet adatot kivenni, ez kettővel növeli a mutató értékét.

b. Direktívák csoportosítása, működésük

Olyan parancsok, amelyek vagy a fordítást, vagy pedig a futtatást befolyásolják. A direktívákat és az operandusaikban lévő szimbólumokat is a lexikális elemző határozza meg, ez általában előfeldolgozó modul segítségével történik. Assembly példák:

- .DOSSEG (program szegmenseket szigorú sorrendben akarjuk betölteni)
- .MODEL SMALL/MEDIUM (mennyi kódszegmens van, vagyis az eljáráshívások NEAR, illetve FAR típusúak-e)
- .STACK, .DATA, .CODE (a program megfelelő szegmenseire utalnak)
- END (programvég)

Pascal példák:

- kapcsoló direktívák (kétféle lehetőségük van, + jelre bekapcsolnak, - jelre ki). Pl.: {\$B-} csak addig vizsgálja a kifejezést (and és or), amíg nem dönthető el egyértelműen; {\$D+} debug infókat fűz a fordító a programhoz; {\$I+} I/O műveletek automatikus ellenőrzése - kikapcsolva IOResult-tal ellenőrizhető és kezelhető

- paraméteres direktívák (a direktíva után SPACE-el elválasztva paraméter következnek). Pl.: $\{ \$M \}$ segítségével a memóriaméretetek adhatók meg
- feltételes fordítás: csak akkor fordítja le, ha megadott feltétel teljesül. $\{ \$IF \text{ feltétel} \} \dots \{ \$ENDIF \}$

c. LR(1) grammatikák és elemzések

Alulról felfelé elemzéseknél a terminális szimbólumokból kiindulva haladnak a kezdőszimbólum felé. Ezek az elemzések egy vermet használnak, és a vizsgált módszerek léptetés-redukálás típusúak, azaz az elemző mindig egy redukciót próbál végezni, ha ez nem sikerül, akkor az input szimbólumsorozat következő elemét lépteti. Ezek az elemzések a legjobboldalibb levezetés "inverzét" adják. Az elemzés alapproblémája a mondatformák nyelének a meghatározása.

Első lépésben a grammatika szabályaiból egy táblagenerátorral elkészítünk egy elemző táblázatot, majd az elemző program a táblázattal és egy veremmel tudja elemezni a szöveget. Egy G' kiegészített grammatikát LR(k) grammatikának nevezünk, ha

$$S' \rightarrow \alpha A w \rightarrow \alpha \beta w$$

$$S' \rightarrow \gamma B x \rightarrow \gamma \delta x = \alpha \beta y$$

$\text{FIRST}_k(w) = \text{FIRST}_k(y)$ esetén $\alpha = \gamma$, $A = B$, $x = y$. Ekkor $\alpha \beta w$ mondatformában, k szimbólumot előreolvasva a w első szimbólumától egyértelműen meghatározható, hogy β a nyél, és az $A \rightarrow \beta$ szabály szerint kell redukálni. Minden LR(k) grammatikához létezik egy vele ekvivalens LR(1) grammatika. Ha egy $\alpha \beta x$ mondatforma nyele β , akkor az $\alpha \beta$ prefixeit az $\alpha \beta x$ járható prefixeinek nevezzük.

Működés

A szimbólumokat olvasási sorrendben egy verembe helyezzük el, közben vizsgáljuk, hogy nem alakult-e ki egy olyan csoport, amely a levezetési szabály jobboldala, vagyis egy nyél. Amennyiben a csoport nyél, akkor helyébe a szabály baloldalán álló nemterminálist írjuk.

Az elemzés menete:

A verem szimbólumpárokat tartalmaz, az első elemben egy terminális vagy nemterminális szimbólumot tárolunk, a másodikban az automata állapotának a sorszámát. A verem kezdeti tartalma legyen (#,0). Ha az automata egy léptetést hajtott végre, akkor a beolvasott szimbólum és az új állapot sorszáma kerül a verembe. A redukció állapotban, ha az $A \rightarrow \alpha$ szabály szerint kell redukálni, akkor töröljük a verem $|\alpha|$ db. sorát, azaz $2^{*|\alpha|}$ elemét, írjuk a verem tetejére, az első elembe az A szimbólumot, lépünk vissza az automatában a járható prefixszel bejárt úton $|\alpha|$ lépést. Határozzuk meg, hogy ebből az állapotból az A hatására melyik állapotba kerül az automata, és ezt az állapotsorszámot írjuk a verem tetején levő második elembe. Ha egy állapotátmenet az $S' \rightarrow S$ szabály redukciójához jutunk, akkor az elemző az input szöveget elfogadta és megáll. Ha egy állapotból olyan input szimbólum hatására kell továbblépni, amelyhez a táblázatban nincs megadva állapotsorszám, akkor az elemzés szintaktikus hibát detektál, és megáll.