GÉPTŐL FÜGGETLEN SZEMLÉLET KIALAKITÁSA A PROGRAMTERVEZŐK OKTATÁSÁBAN

Kalmár László

A József Attila Tudományegyetem Természettudományi Karán az 1957-58 tanévtől kezdve foglalkozunk - eleinte alacsony, az utóbbi időben egyre inkább növekvő létszámban - számitástechnikai szakemberek képzésével. Jelenleg kétféle képzési forma van nálunk. Az ötéves egyetemi képzésben részesülő programtervezőket elsősorban a software-fejlesztés, software-adaptálás és software-karbantartás feladatának ellátására képezzük ki. A hároméves, főiskolai szintü, egyetemi képzésben részesülő programozók legnagyobb része leendő munkahelyén programozási rutin-feladatokat fog ellátni, célszerüen 8-10-en egy-egy programtervező irányitásával. Ez utóbbiak képzéséről most nem beszélek.

A számitástechnika mai helyzetét figyelembe véve nélkülözhetetlen, hogy a programtervezők megtanuljanak programot irni gépi kódban, vagy ahhoz közelálló assembly-nyelven. Hiszen aki sohasem szervezett ciklust gépre irányitott nyelven, annak számára egy for /a FORTRANban do/ utasitás csak egy "jámbor óhaj", ti. hogy a gép legyen szives a következő utasitásokat ismételten végrehajtani mindaddig, amig valamely paraméter adott értéket el nem ér, vagy valamely más végfeltétel nem teljesül; de sejtelme sincs arról, hogy a gép hogyan teljesiti ezt az óhajt. Ez pedig nem elegendő sem forditóprogram irására, sem kész forditóprogram által alkalmazott algoritmus hatékonyságának megitélésére, esetleg hatékonyabbal való pótlására. Számtalan más példát is felhozhatnék arra, hogy gépre irányitott nyelven való programozási ismeretek nélkül nem lehet ellátni rendszerprogramozási feladatokat. Igaz, vannak már egyes hazánkban is működő gépre implementált rendszerprogramiró nyelvek, ezek hasz-

nálata meg is könnyiti a rendszerprogramok irását és karbantartását, ezek kizárólagos használata azonban ma még nem vezet hatékony rendszerprogramokhoz.

Minthogy tehát a programtervezőknek mindenképpen meg kell tanulniok a hazánkban üzemelő gépek gépi kódjában /vagy assembly nyelvén/ programozni, didaktikai tapasztalataink alapján ugy döntöttünk, hogy kezdjük mindjárt ezzel. Ugyanis tapasztalataink szerint igy értik meg legkönnyebben azt, hogyan futnak le a gépen a magasabb szintü nyelveken irt programok.

Csakhogy, mint előző előadásomban már mondtam, Magyarország számitógép-muzeum. Előre nem tudhatjuk, hogy végző hallgatóink milyen számitógépet üzemeltető központba vagy vállalathoz, kutatóintézetbe stb. kerülnek majd. Általános tapasztalat, hogy egyetlen gép utasitás-választékához való kötődés megnehezíti más rendszerü gépekre irányitott nyelven való programozást. Ezért több tankönyviró kitalált, un. fiktiv gép belső nyelvén tanit programozni. /Gondolok pl. Flores kitünő könyveiben használt FLAP, vagy Knuth könyvsorozatában használt MIX fiktiv gépre irányitott nyelvekre./ De ekkor meg olyan gépre irányitott nyelven tanul programozni a hallgató, amely épp azért lehet elegáns, mert maga a gép nem is létezik, tehát minden létező gép különbözik tőle.

Mi Szegeden néhány éve ugy oldottuk meg ezt a nehézséget, hogy nem egy fiktiv számitógépet definiálunk, hanem egy sereg fiktiv utasitást. Jobban mondva, ezek nem is mind fiktivek, hiszen némelyek pontosan megegyeznek egy-egy gép valamely utasitásával, némelyek meg csak a cimek felcserélésében térnek el tőle. Voltaképpen egy univerzális jelölésrendszert definiálunk, amelyben bármely gép bármely utasitása felirható. Ha nem egyetlen utasitással, néhány utasitásból összetett makróval. /Az ilyen makrókat tréfásan "mini-makróknak" nevezzük./

Vannak közöttük egycimes, kétcimes és háromcimes utasitások is. Az utóbbiak nem azért, mintha arra számitanánk, hogy valamely hallgatónk háromcimes géphez kerül majd dolgozni. Hanem azért, hogy ezeket kevesebb cimü gépekre irt program esetén mini-makróként használhassuk.

Vannak köztük segédutasitások /regiszterekbe töltő, regiszterekből a memóriába, vagy pl. egyik memóriarekeszből a másikba tároló utasitások, sőt még megállitó utasitások is, az utóbbiak főleg azért, hogy elmondhassuk, hogy a mai gépeken ilyen utasitások nincsenek, hanem külön e célra irt programmal gondoskodunk arról, hogy egy számitás befejezése után a gép automatikusan térjen át a következő számitásra, ahol még annak megadására is rengeteg mód van, hogy melyiket tekintse a várakozók közül következőnek/; müveleti utasitások /olyanok is, amelyeknél a gép mindkét operandusz egy-egy memóriarekeszből veszi, olyanok is, ahol egyik vagy akár mindkét operandusz valamely regiszterből veszi; olyanok is, amelyeknél a művelet eredményét mindjárt tárolja is a gép valamely memóriarekeszbe, olyanok is, amelyeknél a művelet eredményén további, szintén az utasitásban előirt műveletet hajt végre; e műveletek lehetnek aritmetikai alapműveletek, az operanduszok abszolut értékén végzett műveletek, olyan műveletek, amelyek eredményének csak abszolut értékét számitja ki a gép, bitenkénti logikai műveletek stb./.

De legfőképpen nagyon sokféle ugró utasitást definiálunk. A feltétlen ugráson kivül az aritmetikai alaprelációknak memóriarekeszek és/vagy regiszterek tartalmán, esetleg egyik relanduszként a 0-n való ellenőrzését kivánó feltételes ugrások is. Olyanok is, amelyekben csak egy ugráscim van megadva és az ugrásfeltétel nem teljesülése esetén nem történik ugrás, olyanok is, amelyekben külön-külön ugráscim van megadva az ugrásfeltétel teljesülése esetén is és nem teljesülése esetén is, végül olyanok is, amelyekben az ugrásfeltétel teljesülése esetén csak a következő utasitást ugorja át a gép. Definiálunk tulcsordulás-jeltől vagy más egyhites regiszter tartalmától függő feltételes ugrásokat is. Természetesen ciklus-szervezés, vagy szubrutin-készités megkönnyitésére szolgáló speciális ugró utasitásokat is definiálunk.

Mindezeket az utasitásokat indexregiszter nélküli és indexregiszter/ek/ tartalmától függően automatikusan módositott változatban is definiáljuk. Természetesen definiálunk

/közvetlen vagy közvetett operanduszu/ indexmanipulációs utasitásokat is.

Egy-egy feladat megoldásakor megadjuk, hogy a definiált utasitások közül melyeket szabad használni. Ha azt akarjuk megtanitani, hogy egyes utasitásokat mely másokkal lehet pótolni, ha a kérdéses utasitás nincs meg a gépen, akkor a felhasználható utasitás-készletet szükre szabjuk. Ha meg valami más ismeret elsajátittatásán /pl. a ciklus-szervezés, a szubrutin-készitás különböző módszerein/ van a hangsuly, akkor meg bőven megengedjük a már tanult utasitások használatát, pl. minden egycimes utasitást megengedünk.

A hazánkban üzemelő legfontosabb gépek belső nyelvét /rendszerint a gyakorlatokon/ ugy ismerik meg a hallgatók, hogy a kérdéses nyelv utasitásait a mi jelölésrendszerünkben irjuk fel, ha egy utasitásban nem lehetséges, mini-makróként. Azután megtanitjuk a hallgatóinknak, hogy a kérdéses gépet használó programozók milyen mnemonikus kóddal nevezik meg ezt az utasitást.

Kezdetben az utasitásokat az általunk definiált jelölésrendszer szerinti mnemonikus kóddal és oktális valódi cimekkel adjuk meg. A program-ürlapon azonban van egy-egy oszlop /annyi, ahány cimes a gép/ a cimek tartalmának jelölése számára is. Miután megtanitjuk a memória-felosztás legegyszerübb módjait, nem töltjük ki az abszolut cimeknek szánt oszlopot és máris megtanulták a hallgatók a legegyszerübb /szimbolikus cimes, és minthogy már a makró fogalmát ismerik, makrókkal bővithető/ assembly-nyelvet. Ebben hasonló módon felirjuk a hazánkban üzemelő legfontosabb gépek alacsonyabb-magasabb szintü assembly-nyelveit is. Minthogy ugrás-cimek esetén nem sok értelme van a cim tartalmát adni meg szimbólikus cimként, ehelyett bevezetjük a cimkék fogalmát; program-ürlapunkon már eleve van hely a cimkéknek. A cimkéket természetesen a módositandó /pl. átcimzendő/ utasitások megnevezésére is alkalmazzuk. A módositó-konstansokra szintén van géptől független jelölés a jelölés-rendszerünkben. Minthogy sok volna minden olyan utasitásnak cimkét adni, amelyre hivatkozunk /pl. ahova ugrunk, vagy amelyet módositunk/, bevezetjük a cimke + 1,

cimke - 3 és hasonló "cimke-kifejezéseket" is, köztük az önrelativ cimeket tartalmazókat is. Ha már cimkékkel végzünk müveleteket /hacsak nagyon primitiveket is/, miért ne végezhetnénk szimbolikus cimeken is? Igy eljutunk az aritmetikai kifejezéseket, zárójeleket is megengedő programozási nyelveken át végül a magasabb szintü nyelvekig is.

Tapasztalatunk szerint ezen a módon sikerül a hallgatók géptől független szemléletét kialakitani, ugy, hogy munkahelyükön nem jelent nagyobb megrázkódtatást rendszerprogramozó munkájukban sem, ha uj gépet állitanak be, mint egy jó gépiró számára az, ha az addig megszokott helyett uj irógépen kell gépelnie.