

Kalmár László: Géptől független szemlélet kialakítása a programtervezők oktatásában, Visegrád, Számítástechnikai oktatási konferencia, 1974. 142-146.

"Magyarország számítógép múzeum."

"Mi Szegeden néhány éve úgy oldottuk meg ezt a nehézséget, hogy nem egy fiktív számítógépet definiálunk, hanem egy sereg fiktív utasítást. Jobban mondva, ezek nem is mind fiktívek, hiszen némelyek pontosan megegyeznek egy-egy gép valamely utasításával, némelyek meg csak a címek felcserélésével térnek el tőle. Voltaképpen egy univerzális jelölésrendszert definiáltunk, amelyben bármely gép bármely utasítása felírható. Ha nem egyetlen utasítással, néhány utasításból összetett makróval."

Géptől független szemlélet kialakítása 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 – számítástechnikai 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 szintű, egyetemi szintű, egyetemi képzésben részesülő programozók legnagyobb részére leendő munkahelyén programozási rutin-feladatokat fog ellátni, célszerűen 8-10-en egy-egy programtervező irányításával. Ez utóbbiak képzéséről most nem beszélek.

A számítástechnika mai helyzetét figyelembe véve nélkülözhetetlen, hogy a programtervezők megtanuljanak programot írni gépi kódban, vagy ahhoz közelálló assembly nyelven. Hiszen aki sohasem szervezett ciklust gépre irányított nyelven, annak számára egy *for_*/a FORTRAN-ban *do/* utasítást csak egy „jámbor óhaj”, ti. hogy a gép legyen szíves a következő utasításokat ismételtlen végrehajtani mindaddig, amíg valamely paraméter adott értéket el nem ér, vagy valamely más végfeltétel nem teljesül; de sejtelve sincs arról, hogy a gép hogyan teljesíti ezt az óhaját. Ez pedig nem elegendő sem fordítóprogram írására, sem kész fordítóprogram által alkalmazott algoritmus hatékonyságának megítélésére, esetleg hatékonyabbal való pótlására. Számtalan más példát is felhozhatnánk arra, hogy a gépre irányított nyelven való programozási ismeretek nélkül nem lehet ellátni rendszerprogramozási feladatokat. Igaz, vannak már egyes hazánkban is működő gépre implementált rendszerprogramíró nyelvek, ezek használata meg is könnyíti a rendszerprogramok írását és karbantartását, ezek kizárólagos használata azonban ma még nem vezet hatékony rendszerprogramokhoz.

Minthogy tehát a programtervezőknek mindenképpen meg kell tanulniuk a hazánkban üzemelő gépek gépi kódjában /vagy assembly nyelvén/ programozni, didaktikai tapasztalataink alapján úgy döntöttünk, hogy kezdjük mindjárt ezzel. Ugyanis tapasztalataink szerint így értik meg legkönnyebben azt, hogyan futnak le a gépen a magasabb szintű nyelveken írt programok.

Csak hogy, mint előző előadásomban már mondtam, Magyarország számítógép múzeum. Előre nem tudhatjuk, hogy végző hallgatóink milyen számítógépet üzemeltető központba vagy vállalathoz, kutatóintézetbe stb. kerülnek majd. Általános tapasztalat, hogy egyetlen gép utasítás-választékához való kötődés megnehezíti más rendszerű gépekre irányított nyelven való programozást. Ezért több tankönyvíró kitalált, ún. fiktív belső nyelven tanít programozni. /Gondolok pl. Flores kitűnő könyveiben használt FLAP, vagy Knuth könyvsorozatában használt MIX fiktív gépre irányított nyelvekre./ De ekkor meg olyan gépre irányított 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 úgy oldottuk meg ezt a nehézséget, hogy nem egy fiktív számítógépet definiálunk.

hanem egy sereg fiktív utasítást. Jobban mondva, ezek nem is mind fiktívek, hiszen némelyek pontosan megegyeznek egy-egy gép valamely utasításával, némelyek meg csak a címek felcserélésével térnek el tőle. Voltaképpen egy univerzális jelölésrendszert definiáltunk, amelyben bármely gép bármely utasítása felírható. Ha nem egyetlen utasítással, néhány utasításból összetett makróval. /Az ilyen makrókat tréfásan „mini-makrónak” nevezzük./

Vannak köztük egycímes, kétcímes és háromcímes utasítások is. Az utóbbiak nem azért, mintha arra számítanánk, hogy valamely hallgatónk háromcímes géphez kerül majd dolgozni. Hanem azért, hogy ezeket kevesebb című gépekre írt program esetén mini-makróként használhassuk.

Vannak köztük segédutasítások regiszterekbe töltő, regiszterekből a memóriába, vagy pl. egyik memóriarekeszből a másikba tároló utasítások, sőt még megállító utasítások is, az utóbbiak főleg azért, hogy elmondhassuk, hogy a mai gépeken ilyen utasítások nincsenek, hanem külön e célra írt programmal gondoskodunk arról, hogy egy számítás befejezése után a gép automatikusan térjen át a következő számításra, ahol még annak megadására is rengeteg mód van, hogy melyiket tekintse a várakozók közül következőnek /művelet utasítások, olyanok is, amelyeknél a gép mindkét operanduszt egy-egy memóriarekeszből veszi; olyanok is, ahol egyik vagy akár mindkét operanduszt 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 utasításban előírt műveletet hajt végre; e műveletek lehetnek aritmetikai alpműveletek, az operandusztok abszolút értékén végzett műveletek, olyan műveletek, amelyek eredményét csak abszolút értékét számítja ki a gép, bitenkénti logikai műveletek stb./.

De legfőképpen nagyon sokféle ugró utasítást definiálunk. A feltétlen ugráson kívül az aritmetikai alaprelációnak memóriarekeszek és/vagy regiszterek tartalmán, esetleg egyik operanduszként a 0-n való ellenőrzését kívánó feltételes ugrások is. Olyanok is, amelyekben csak egy ugráscím van megadva és az ugrásfeltétel nem teljesülése esetén nem történik ugrás, olyanok is, amelyekben külön-külön ugráscím van megadva az ugrásfeltétel teljesülése esetén is és nem teljesülése esetén is, végül olyanok is, amelyen az ugrásfeltétel teljesülése esetén csak a következő utasítást ugorja át a gép. Definiáltunk túlcsoportulást-jelöltől vagy más egy bites regiszter tartalmától függő feltétel ugrásokat is. Természetesen ciklus-szervezés, vagy szubrutin-készítés megkönnyítésére szolgáló speciális ugró utasításokat is definiálunk.

Mindezeket az utasításokat indexregiszter nélküli és indexregiszter/ek/ tartalmától függően automatikusan módosított változatban is definiáljuk. Természetesen definiálunk /közvetlen vagy közvetett operandusú/ indexmanipulációs utasításokat is. Egy-egy feladat megoldásakor megadjuk, hogy a definiált utasítások közül melyeket szabad használni. Ha azt akarjuk megtanítani, hogy egyes utasításokat mely másokkal lehet pótolni, ha a kérdéses utasítás nincs meg a gépen, akkor a felhasználható utasítás-készletet szűkre szabjuk. Ha nem valami más ismeret elsajátításában /pl. a ciklus-szervezés, a szubrutin-készítés különböző módszerein/ van a hangsúly, akkor meg bőven megengedjük a már tanult utasítások használatát, pl. minden egycímes utasítást megengedünk.

A hazánkban üzemelő legfontosabb gépek belső nyelvét rendszerint a gyakorlatokon úgy ismerik meg a hallgatók, hogy a kérdéses nyelv utasításait a mi jelölésrendszerünkben írjuk fel, ha egy utasításban nem lehetséges, mini-makróként. Aztán megtanítjuk a hallgatóinknak, hogy a kérdéses gépet használó programozók milyen mnemonikus kóddal nevezik meg ezt az utasítást.

Kezdetben az utasításokat az általunk definiált jelölésrendszer szerinti mnemonikus kóddal és oktális valódi címekkel adjuk meg. A program-úrlapon azonban van egy-egy oszlop /annyi, ahány címes a gép/ a címek tartalmának jelölése számára is. Miután megtanítjuk a memória-felosztás legegyszerűbb módjait, nem töltjük ki az abszolút címeknek szánt oszlopot, és máris megtanulják a hallgatók a legegyszerűbb /szimbolikus címes, és minthogy már a makró fogalmát ismerik, makrókkal bővíthető/ assembly-nyelvet. Ebben hasonló módon felírjuk a hazánkban üzemelő legfontosabb gépek alacsonyabb-magasabb szintű assembly-nyelveit is. Minthogy ugrás-címek esetén nem sok értelme van a cím tartalmát adni meg szimbolikus címként, ehelyett bevezetjük a címek fogalmát; program-úrlapunkon már eleve van hely a címkéknek. A címkéket természetesen a módosítandó /pl. átcímezendő/ utasítások megnevezésére is alkalmazzuk. A módosító-konstansokra szintén van géptől független jelölés a jelölés-rendszerünkben. Minthogy sok volna minden olyan utasításnak címkét adni, amelyre hivatkozunk /pl. ahova ugrunk, vagy amelyet módosítunk/, bevezetjük a címke + 1, címke - 3 és hasonló „címke-kifejezéseket” is, köztük az önrelatív címeket tartalmazókat is. Ha már címkékkel végzünk műveleteket /hacsak nagyon primitíveket is/, miért ne végezhetnénk szimbolikus címeken is? Így eljutunk az aritmetikai kifejezéseket, zárójeleket is megengedő programozási nyelveken át végül a magasabb szintű nyelvekig is.

Tapasztalatunk szerint ezen a módon sikerül a hallgatók géptől független szemléletét kialakítani, úgy, hogy munkahelyükön nem jelent nagyobb megrázkódtatást rendszerprogramozó munkájukban sem, ha új gépet állítanak be, mint egy jó gépiró számára az, ha az addig megszokott helyett új írógépen kell gépelnie.