Simon Endre: A Kalmár-féle fiktív elektronikus számítógép szimulátora MINSZK-22 gépen, Programozási rendszerek '72, Szeged, 1972; 263-268.

Az IBM PC DOS operációs rendszer alatt megvalósított Kalmár-féle fiktív gép szimulátor, Készítette: Szofi Bt., 2000.

A Kalmár-féle fiktív programozási nyelvre két szimulátor készült. Egyik a Minszk-22 számítógépre, a másik jóval később PC DOS operációs rendszer alatt működik. Az utóbbi használatához beépítettünk egy parancssoros üzemmódú szimulátort, mellyel kétcímes programokat tudunk futtatni.

A KALMÁR-FÉLE FIKTÍV ELEKTRONIKUS SZÁMÍTÓGÉP SZIMULÁTORA MINSZK-22 GÉPEN

Simon Endre

MTA Matematikai Logikai és Automataelméleti Tanszéki Kutató Csoportja

1. A Kalmár-féle fiktív elektronikus számítógépet (későbbiekben fiktív gép) Kalmár László akadémikus definiálta oktatási célokra. A programtervező matematikus szakos hallgatók gépi kódú programozási előadásain nem lett volna célszerű egy konkrét gépkonfigurációt választani és mint etalonra, erre a gépre írni fel a különböző oktató programokat. Ezek ugyanis akarva-akaratlanul kihasználják az illető géptípus speciális adottságait.

Konkrét gép esetén problémát okozott volna az a mai napig egyértelműen meg nem válaszolható kérdés, hogy az etalonnak kiválasztandó gép, egy- két- vagy háromcímes legyen.

A fiktív gépet tehát úgy kellett definiálni, hogy annak legyen egy- két- sőt háromcímes változata is.

Így egy hallgató, aki a fiktív gépen tanult programozni, konkrét gép mellé kerülne viszonylag gyorsan képes az elsajátított programozástechnikai módszerek alkalmazására a gyakorlatban is.

Nyilvánvaló hátránya a fiktív gépen való programozásnak, hogy nincs lehetőség a felírt programoknak a való kipróbálására, futtatására. Azon túl, hogy ez a tény megfosztja a hallgatókat a didaktikai szempontból semmiképpen nem lebecsülendő sikerélménytől, lehetetlenné teszi a visszacsatolást.

Mindezek a problémák adták az ötletet, hogy készítsük el a fiktív gép összes változatának szimulátorát a JATE Kibernetikai Laboratóriumban üzemelő MINSZK-22 elektronikus számítógépre, és adjunk lehetőséget a hallgatók által írt programok ezen, szimulált gépeken történő futtatására.

2. **A fiktív gép utasításrendszere**

Megállapodás szerint a gép egyes regisztereit a következőképpen jelöljük:

E: eredményregiszter (gyűjtős gépnél mindig a gyűjtőt jelenti)

R: szorzó / osztó regiszter

T: túlcsordulás regiszter

U: utasításszámláló regiszter

I k : k-adik indexregiszter

A fiktív gépnek két fő típusa definiált:

2.1. egycímes indexregiszter nélküli gép

ekkor egy gépi utasítás általános alakja a következő:

Θ a ahol

Θ - a művelet jele (mindig egy betű)

a - a címrész

például:

A a : (E) + (a) => (E)

B b : (a) => (E)

2.2. indexregiszteres gép

ennek létezik egy-, két- és háromcímes változata, az utasítások általános formája ennek megfelelően:

νΘ a, i

νΘ a, i b, j

νΘ a, i b, j c, k

Ahol

ν - a változat jele (szám, esetleg kiegészítő jelek)

Θ - a művelet jele (egy vagy két betű)

a a

b címek ab címrészek

c abc

i

j indexregiszter hivatkozások

k

νΘ együtt a gépi utasítás műveleti kódja

például:

1A a, i : (E) + (a+(I i)) => (E)

2AT a b : (a) + (b) => (E) , (a)

1OF a b c : a => (U) ha (b) >= (c)

1BX a, i : a => (I i)

A 2.2-ben leírt gép mindhárom változata esetén érvényes a következő:

- az indexregiszterek címrész hosszúságúak

- a címrész után az indexhivatkozás opcionális

- ha a hivatkozás elmarad, ez azt jelenti, hogy a 0 című indexregiszter van az illető címrészhez hozzárendelve.

- a 0 című indexregiszterben nulla van és a beleírás tiltott

2.3. Utasítási formátumok

A fiktív gép gépi kódú programjait előre adott formátumú kód űrlapra írjuk. Kétcímes gép esetén a helyes utasítás formátumokat a példa mutatja:

Címke	Cím	Műv. kód	I. címrész		II. címrész		Megjegyzés
	tart.	cím	tart.	cím			
BELÉP:	100)	2AT	PAR1	200	PAR2	201	(PAR1) + (PAR2) => (PAR1)
	+1)	2BX	EGY	1,10		0,2	1 ? (I 1) => ? (I 2)
C1:	+2)	2FX	C2:	103,10			® C2: ha (I 10) > 0
C2:	103)	1SP		0			STOP 0 => (E)
	+4)	1SP		104,0			STOP utasítás => (E)
PAR1:	200)	117					konstans1
PAR2:	+1)	77					konstans2

A címke, megjegyzés és tartalom mező kitöltése opcionális.

3. Szimulációs rendszer

Szimulációs rendszeren értjük a továbbiakban a szimulátort a hozzákapcsolt segédprogramokkal együtt. Mivel a szimulátor működésekor feltételezi, hogy a futtatandó program a fiktív gép memóriarekeszeiben már elhelyezésre került, szükség van egy beolvasó programra.

A rendszert aktiváló hallgatókról nem feltételezzük fel a MINSZK-22 ismeretét, tehát része egy vezérlőprogram, mely a külvilággal egy ASR-33 típusú írógépen keresztül tartja a kapcsolatot.

A programpróbák megkönnyítését szolgálja a rendszer részét képező nyomozó program.

A szimulációs rendszer így a következő modulokból épül fel:

- beolvasó program

- szimulátor (ennek része a nyomozó illetve a címkövető program)

- vezérlőprogram

- az aritmetikai egységet szimuláló szubrutinok

3.1. A beolvasó program

Lyukszalagról vagy írógépről beolvassa a programot, szintaktikus és szemantikus ellenőrzést végez: az utasításokat előre adott formában tárolja a fiktív gép adott című „rekeszében”.

Megjegyezzük, hogy az egységes kezelés miatt az egy-, két- illetve háromcímes gépek esetén is ugyanazt az utasítás reprezentációs formát alkalmaztuk (az utasítás műveleti kódja egyértelműen meghatározza a hozzátartozó címek számáz).

A fiktív gép egy memóriarekeszét a MINSZK-22 két egymás utáni szavára képeztük le. Az 1K-s fiktív memória így 37778 szót foglalt el.

A beolvasó program egy programsor karaktereit egy rekeszenként 4 byte-os pufferbe gyűjti. Minden szintaktikus kategória felismerésekor információt rak le egy stack következő szabad helyére. Ez az információ tartalmazza a felismert kategória sorszámát, valamint első pufferbeli karakterének címét (rekeszcím-bytecym). Sor vége hatására a stack elemeit generálja és a sorszám ismeretében elvégzi a hozzárendelt tevékenységet (például: címkéhez rendelt sorszám esetén veszi a stack következő elemét)

Ha a stack minden elemét generálta – a kapott utasítást vagy konstanst elhelyezte a megadott fiktív rekeszbe – eldönti, hogy van-e listázási kérelem. Ha nem olvassa a következő programsort, ha igen a stack elemeit újból generálva a szélesnyomtatón előre meghatározott formában reprodukálja a beolvasott információt, majd rátér az olvasásra. A program végét egy speciális (END) direktíva jelöli.

3.2. Szimulátor

A szimulátor megtervezésénél egy számítógép mikró-szintjén lejátszódó folyamatokat programoztuk.

A címkövetés illetve nyomozás – mely a szimulációs rendszer opcionális szolgáltatása – tulajdonképpen nem más mint a fiktív gép bizonyos regisztertartalmainak output adása. A vezérlőprogram a MINSZK-22-höz kapcsolt írógépből, megszakításos üzemben képes parancsszavakat átvenni. A jelsorozatok felismerésére a Dömölky-féle algoritmus egy általunk módosított változatát használjuk.

4. Befejezés

A szimulációs rendszer életre kelése után a matematikus hallgatók gépi gyakorlatát Kalmár professzor kérésére már e rendszerre támaszkodva vezették.

Kétéves üzemeltetési tapasztalataink alapján a rendszert hasznos oktatási segédeszköznek mondhatjuk.

Simon Endre

Az IBM PC DOS operációs rendszer alatt megvalósított

Kalmár-féle fiktív gép szimulátor

Készítette: Szofi Bt.

Történeti bevezető:

Kalmár Lászlónak elévülhetetlen érdemei vannak a felsőfokú informatikai szakemberképzés beindításában. Az egyszakos tanárképzés megszüntetésekor elérte, hogy a minisztérium engedélyezze: a harmadéves tanárjelöltek 5 százaléka az egyik szak elhagyásával a megmaradt tantárgy egy speciális területén elmélyültebb tanulmányokat folytathasson. Így 1957 őszén három hallgatójával elindult az alkalmazás-matematikus képzés

tanulmányokat folytathasson. Így 1957 őszén három hallgatóval elindult az alkalmazó-matematikus képzés, elsőként az országban.

1963-ra sikerült önálló szakként beindítani a programtervező-matematikus képzést. Az induláskor a szakon oktatott tárgyak voltak: analízis, algebra és számelmélet, geometria, általános fizika, elméleti fizika, matematikai logika, differenciálegyenletek, numerikus matematika, matematikai gépek, matematikai laboratórium, gépi programozás, halmazelmélet, valószínűségszámítás, programozás, matematikai modellezési gyakorlat, programozási nyelvek. 1963-ban alakult meg a Kibernetikai Laboratórium is, amely az oktató- és a kutatómunka számítógépes hátterét adta.

Kalmár-féle fiktív gép:

A programtervező-matematikus képzés sokáig erősen elméleti jellegű volt. Ez három okra vezethető vissza. Egyrészt a szak a matematikus, sőt a matematika-fizika tanárszakból nőtt ki, másrészt sokáig az volt a nézet, hogy a számítástechnikát (informatikát) a szaktudományokon, elsősorban a fizikán keresztül lehet a gyakorlatban alkalmazni. Fontos ok volt továbbá, hogy a képzéshez nem volt megfelelő számítástechnikai infrastruktúra. Mai szemmel nézve is kiemelkedő az a módszer, amelyet Kalmár László alkalmazott a technikai, technológiai korlátok leküzdésére: olyan, a valóságban nem létező (fiktív) gépet (gépeket) konstruált, amely az összes lényeges számítógépes utasítástípust végrehajtotta, ezt használta az általános gépi programozási fogalmak, módszerek oktatására. A hallgatók sokáig ilyen fiktív gépeken szereztek programozási gyakorlatot.

A fiktív gép létrehozását az is indokolta, hogy az abban az időben létező számítógépek eltérő architektúrával rendelkeztek. A gépi kódú programozás előadáson nem lett volna célszerű egy konkrét gépkonfigurációt választani, és mint etalonra, arra a gépre írni a különböző oktató programokat. Ezek ugyanis akarva-akaratlanul kihasználják az illető géptípus adottságait. Konkrét gép esetén problémát okozott volna az az abban az időben egyértelműen meg nem válaszolható kérdés, hogy az etalonnak kiválasztandó gép, egy- két- vagy háromcímes legyen. A fiktív gépet tehát úgy kellett definiálni, hogy annak legyen egy-, két-, sőt háromcímes változata is. Így egy hallgató, aki a fiktív gépen tanult programozni, konkrét gép mellé kerülve viszonylag gyorsan képes az elsajátított programozástechnikai módszerek alkalmazására a gyakorlatban is.

A Kalmár gép felépítéséről és programozásáról eredeti előadás jegyzetekből (1957-ből és 1966-ból) nyertünk információt. Ezek digitalizálva megtekinthetőek. A most következő összefoglaló is ezekre a jegyzetekre épül.

A Kalmár gépnek a konkrét géptípustól elvonatkoztatott felépítése miatt sok utasítása van. Vannak egy- két- és háromcímes utasításai is. Ez attól függ, hogy a gép utasításai hány megadott memóriahellyel képesek dolgozni. Az utasítások nevében az első szám jelzi, hogy hány címes, tehát rendre 1, 2 vagy 3. A gép legfontosabb regisztere az eredményregiszter (E). Ez egy kiemelt memória hely a művelet eredményének tárolására. Ennek a regiszternek legnagyobb szerepe az egycímes gépeknél van. A gép rendelkezhet indexregiszterrel is (I). Számuk általában egy, kettő vagy három és segítségükkel relatív címzést valósíthatunk meg. A gép fontosabb utasításai:

Utasítás Jelentés

1A a (E)+(a) -> (E)

2A a b (a)+(b) -> (E)

3A a b c (a)+(b) -> (c),(E)

1S a (E)-(a) -> (E)

2S a b (a)-(b) -> (E)

3S a b c (a)-(b) -> (c),(E)

1'S a (a)-(E) -> (E)

1M a (E)(a) -> (E)*

2M a b (a)(b) -> (E)*

3M a b c (a)(b) -> (c),(E)*

1D a (E)/(a) -> (E)

2D a b (a)/(b) -> (E)

2D a b (a)/(b) -> (E)

3D a b c (a)/(b) -> (c),(E)

1'D a (a)/(E) -> (E)

1B a (a) -> (E)

1T a (E) -> (a) ; (E) = 0

2T a b (a) -> (b),(E)

1ST Megáll a gép

Vannak még ugró- és átugrató utasítások, szalagról olvasó- és oda író utasítások, indexregiszteres utasítások és egyéb speciális utasítások is.

A Kalmár gép programozását a SZOFI által készített emulátoron szeretnénk bemutatni. Ez a Kalmár gép kétcímes, indexregiszter nélküli megvalósítása. Ennek az un. SZOFI Kalmár gépnek az utasításai egy kicsit különböznek az eredeti Kalmár gépétől, ezért áttekintjük a különbségeket (a SZOFI Kalmár gép részletes leírása megtalálható a következő fejezetben):

SZOFI gép Kalmár gép

ADD a b *2A a b*

SUB a b *2S b a*

MULT a b *2M a b*

DIV a b *2D b a*

LOAD a *1T a*

STORE a b *2T a b*

STOP *1ST*

JUMP a *1U a*

J>=0 a b *4F a b*

J=0 a b *6F a b*

J<=E a b *7F a b*

J>E a b *7 F a b*

A SZOFI gépben a műveletek eredményei a második címre és az eredményregiszterbe is bekerülnek, az eredeti Kalmár gépben viszont csak az eredményregiszterbe! Nézzünk meg egy egyszerű hatványozó programot először kétcímes, indexregiszter nélküli Kalmár gépen, majd SZOFI Kalmár gépen:

Kalmár gép:

Cím Műveleti kód Cím1 Cím2 Megjegyzés

201 2T 0 214 0 -> (i)

202 2T 1 213 1 -> (y)

203 6F 210 212 ha n=0 akkor ugrik a végére

204 2M 211 213 (a)(y) -> (E)*

205 1T 213 (E) -> (y)

206 2A 214 1 (i)+1 -> (E)

207 1T 214 (E) -> (i) ; (E)=0

208 1B 214 (i) -> (E)

209 7 F 204 212 ha (E)<(n), akkor ugrik a ciklus elejére

210 1S Megáll a gép

211 2 a

212 12 n

213 0 y

214 0 i

SZOFI Kalmár gép:

STORE 0 i

STORE 1 y

J=0 KI: n

M: MULT a y

ADD 1 i

J>E M: n

KI: WRITE n y

STOP - -

a: #2

n: #12

y: -

i: -

Vegyük észre, hogy a Kalmár gépen konkrét szimbolikus címeket használtunk (indexregiszterrel használhattunk volna relatív címeket is), mert így szerepelne a gép memóriájában is. Ezzel szemben a SZOFI Kalmár gépben írt programot –fordítás után- majd az emulátor helyezi el a memóriában, tehát nem kell címekkel bajlódnunk, kényelmesen programozhatunk egy assembly szintű nyelven.

SZOFI Kalmár gép:

Egyszerűsége miatt a Kalmár-féle fiktív gép még ma is felhasználható gépi kódú programozás oktatására. A SZOFI Magyar-Amerikai Informatikai Oktató- és Vizsgaközpont elkészítette a fiktív gép kétcímes, indexregiszter nélküli változatának emulátorát. A program az eredeti Kalmár László-féle számítógép modellen és assembly nyelven alapszik, Dr. Pávó Imre átdolgozásában. A SZOFI Kalmár-gép szabadon letölthető, terjeszthető és módosítható a Free Software Foundation által készített GNU Public License 2 alapján (

http://www.szofi.hu/gnu/kalmar/index.html).

Általános felépítés: A SZOFI Kalmár gép 32 bites memóriarekeszekkel rendelkezik. Címzésük 0-4095-ig történik, tehát a memória mérete 4 kiloszó (4096*4 bájt = 16 kilobájt). Ebből a **ROM** fél kiloszó (0-511 címen). Előnyös a számok oktális formában történő megjelenítése. Így a rekeszek címzése 0000-7777-ig történik. **RAM** 1000-től. A programok a **RAM** elejéről, tehát az 1000-es című rekeszen indulnak. Az **ALU** egyszerűsített, csak

aritmetikai és relációs műveletek elvégzésére képes. Csak egyetlen regisztert használhatunk, az akkumulátort (jele: **E**). Az utasítások kétcímesek, közvetlen címzésűek. A rekeszek tartalma négybájtos lebegőpontos (tehát tört) számként kezelt a műveleti utasításoknál. Van pár rejtett utasítás, ami lehetővé teszi, hogy a rekeszek tartalmával, mint egész számokkal számoljunk, de ez csak az utasításmódosításnál játszik szerepet a modellgépnél. A gépi kódnál valamivel magasabb szintű nyelv, a Kalmár Assembly elrejti ezek használatát az egyszerűség kedvéért.

Utasítások gépi ábrázolása: A 32 bites rekeszen csak 30 bitet foglalnak el:

6 bit	2 bit	12 bit	12 bit
Műveleti kód	Üres	1. cím	2. cím

Az utasítások felfoghatók egy 30 jegyű bináris számként is, ennek kitüntetett szerepe lesz majd az utasításmódosítás esetében. Látható, hogy a 4 oktális számjegyű címek 4*3=12 bitesként 2^12=4096 rekesz címzését teszik lehetővé. Innen a 16K memóriakorlát.

Utasításrendszer:

- Műveleti utasítások: A négy aritmetikai alapművelet elvégzésére 8 utasítás. Az első négy utasítás két megadott rekesz tartalmát, a másik négy az eredményregiszter és egy megadott rekesz tartalmát használja fel.

ADD a b ..... (E), (b) <- (b) + (a)

SUB a b ..... (E), (b) <- (b) – (a)

MULT a b ..... (E), (b) <- (b) * (a)

DIV a b ..... (E), (b) <- (b) / (a)

ADD' a b ..... (E), (b) <- (E) + (a)

SUB' a b ..... (E), (b) <- (E) – (a)

MULT' a b ..... (E), (b) <- (E) * (a)

DIV' a b ..... (E), (b) <- (E) / (a)

- Segéd utasítások: 5 + 2 db.

LOAD a - ..... Jelentése: (E) <- (a)

WRITE1 a - ..... Jelentése: (a) kinyomtatódik

WRITE2 a b ..... Jelentése: (a) és (b) kinyomtatódik

STORE a b ..... Jelentése: (E), (b) <- (a)

STOP - - ..... Jelentése: a program futása megáll

READ1 a - ..... Jelentése: (a) <- input (bekért szám)

READ2 a b ..... Jelentése: (a), (b) <- input (bekért számok)

- Ugró utasítások: 5 db.

JUMP a - ..... Jelentése: Ugrás az 'a' címre.

J>=0 a b ..... Jelentése: Ugrás az 'a' címre, ha (b) >= 0.

J=0 a b ..... Jelentése: Ugrás az 'a' címre, ha (b) = 0.

J<=E a b ..... Jelentése: Ugrás az 'a' címre, ha (b) <= (E)

J>E a b ..... Jelentése: Ugrás az 'a' címre, ha (b) > (E)

- Rejtett utasítások: A Kalmár-gép valós számokkal végez műveleteket, az utasítás módosítás (statikus átcímzés) viszont az utasítás + 2 címmel, mint egész számmal operál. Ezért az utasításmódosítás kivitelezésére szükség van olyan aritmetikai utasításokra is, amelyek egész számként kezelik a 32 bites rekesz tartalmát.

Ezekből 3 darab van.

ADDD a b ..... Jelentése: (E), (b) <- |(b)| + |(a)|

SUBB a b ..... Jelentése: (E), (b) <- |(b)| - |(a)|

MULTT a b ..... Jelentése: (E), (b) <- |(b)| * |(a)|

Ahol az |(a)| az a rekesz tartalmát jelöli, de a szokásostól eltérően nem négybájtos lebegőpontos számként, hanem négybájtos egészként értelmeződik Ezek az utasítások „rejtettek", mivel a program ill. a léptető konstansok használata esetén a Kalmár Assembly megengedi a sima *ADD, SUB, MULT* utasítások használatát. Ha olyan aritmetikai utasítást helyezünk el az assembly forráskódban, ami a megfelelő konstansokkal (0011, 0021, 0121) operál, akkor az *ADD, SUB, MULT* helyett az *ADDD, SUBB* , illetve *MULTT* utasítások kerülnek a tárgykódba.

Konstansok: A konstansok pár kiemelt fontosságú, meghatározott tartalmú rekesz a **ROM** -ban. A szabvány konstansok részben a 0, 1, 2, -1 számokat tartalmazzák, amiket általában az aritmetikai műveletekhez használunk. A szabvány konstansok másik része, illetve a program konstans speciális számértékek, amelyeket egy utasításhoz adva (vagy kivonva) értelmesen változik meg az utasítás. Ez az un. *utasításmódosítás* , a Neumann-elvű számítógépek újdonsága. A 11-es rekesz tartalmával az utasítás első címét módosíthatjuk 1-gyel. A 21-es rekesz tartalmával a második címet. A 121-es rekesz tartalmával mindkettőt egyszerre. Ezt a műveletet *statikus átcímzésnek* nevezzük (már jó ideje háttérbe szorult a statikus átcímzések szerepe a könnyebben kezelhető relatív címzések miatt.) A 4-es rekesz tartalma eljárások szervezésére szolgál, az un. *csatlakozó utasítások* kialakításával. (Segítségével egy *ADD* utasításból *JUMP* utasítás kehet, kettővel növelve az 1. címet. Ezt az eredményregiszterbe kerülő utasítást fogja egy meghívott eljárás elhelyezni saját végébe, hogy így a vezérlésátadás visszafelé is megtörténjen, s a program folytatódjon az eljárás meghívása utáni címen. Lásd: Eljárásszervezés.)

- Szabvány konstansok:

(0000) = 0 (0011) = 00 0001 0000

(0001) = 1 (0021) = 00 0000 0001

(0002) = 2 (0121) = 00 0001 0001

(0003) = -1

- Program konstans:

(0004) = JUMP-ADD 0002 0000

A Kalmár Assembly fő jellemzői:

- Jelek: A magyar ABC, továbbá a „_", „:", „'", „#", „$", „.", „-„ jelek használhatók. A kis- és nagybetűk meg vannak különböztetve, kivéve az utasításokat. (A hexadecimális jelek pillanatnyilag csak nagybetűvel írhatók.). A megjegyzések a „{„, „}" és a „/*" „*/" jelpárok között történhetnek. Ez a két jelpár a Pascal-hoz hasonlóan egymást felülbírálja, így egyiket a másikba ágyazhatjuk. A megjegyzéseken belül akármilyen más ASCII karakter előfordulhat.

- Címkék: Rekeszeket jelölnek. Változók, és ugrások egyszerű leírását teszik lehetővé. Lehet konstans, ebben az esetben a kód erre a memóriacímre kerül. Lehet azonosítóval jelzett, ebben az esetben a fordító a soron következő memóriacímet adja neki értékül.

- Utasítások: A már bemutatott szintaxissal használhatók. Ne feledkezzünk meg a két cím kötelező jelöléséről.

- Konstansok: A null konstans („-„) konkrét érték nélküli konstansot és címet jelöl. A számkonstansok lehetnek oktálisak, decimálisak, illetve hexadecimálisak. A decimális számok „#" jellel, a hexadecimálisak „$" jellel kell, hogy kezdődjenek. Törtszámok mindhárom számrendszerben felírhatók, tizedespontot használva.

Eljárásszervezés: Az eljárásszervezés a *program konstans* segítségével történik a Kalmár-gépben. Egy eljárás felépítése:
Eljárás: MULT' 1 Csat { Csatoló utasítás elmentése }

. { Eljárástörzs }

Csat: - { Csatoló utasítás mentési helye }

Egy eljárás hívása a főprogramból:

Ön: ADD Ön 0004 { Csatoló utasítás kialakítása }

Ön: ADD Ön 0004 { Csatoló utasítás kialakítása }

JUMP Eljárás - { Eljárás meghívása }

Magyarázata: A csatoló utasítás kialakításánál a következő összeadás zajlik le:

ADD Ön 0004

+ JUMP-ADD 0002 0000 (Program konstans)

JUMP Ön + 2 0004 (Csatlakozó utasítás)

Látható, hogy az eredményregiszterben kettővel nagyobb címre való feltétel nélküli ugró utasítás alakul ki. (A *00004* -es program konstans értéke nem változik, mivel a **ROM** nem írható.) Az eljárás meghívásánál az eredményregiszter értéke nem vész el, sőt a *MULT' 1* Csat segítségével a *Csat* címkével jelölt eljárásvégre helyezzük. Így példánkban a vezérlés az eljárásvégre került csatlakozóutasítás segítségével visszakerül az Ön címkével jelölt utasítást követő második címre, vagyis az eljáráshívást követő utasításra.

Példaprogramok: Nézzük meg néhány példaprogramon keresztül, hogyan is működik a SZOFI Kalmár-gép (További példaprogramok találhatók a KALMAR\PELDAK könyvtárban.). A programok könnyen kipróbálhatók a SZOFI Kalmár-gép emulátor, assembler, és fejlesztői környezet segítségével.

Hatványozó program

```
/*********************************************************************

(c) Szofi Magyar-Amerikai Informatikai Oktató- és Továbbképző Központ, 2000

PROGRAMOZÁS ALAPJAI SEGÉDLET (176a/DEV_098/1.0) honlap: http://www.szofi.hu

*********************************************************************

*

* 3. jegyzetpélda. Ciklusos programok I.

*

* Írjunk természetes kitevőre hatványozó programot:

* y <- a^n, ahol ,,n'' természetes szám és ,,a'' nem egyenlő 0-val

* bemenő adatok: a, n

* kimenő adat: y

* munka- (v. számláló) rekesz: i

*

*/


STORE 0 i

STORE 1 y

J=0 KI: n

M: MULT a y

ADD 1 i
```

J>E M: n

KI: WRITE n y

STOP - -

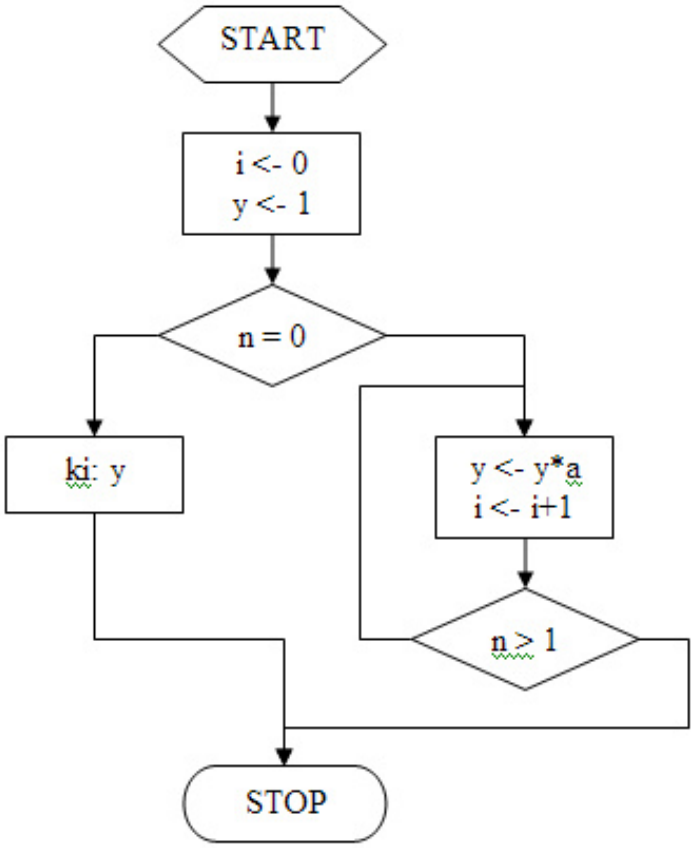
a: #2

n: #12

y: -

i: -

A program folyamatábrája:



Maximumkiválasztó program

/***

(c) Szofi Magyar-Amerikai Informatikai Oktató- és Továbbképző Központ, 2000

PROGRAMOZÁS ALAPJAI SEGÉDLET (176a/DEV_098/1.0) honlap: http://www.szofi.hu

****

*

* 8. jegyzetpélda. Összetett ciklusos programok

* 8. jegyzetpélda: Összetett ciklusos programok

*

* Írjunk programot n szám maximumának kiválasztására.

*

* y <- max(a[1] + a[2] + ... + a[n-1] + a[n])

*

* bemenő adatok: n, a[1] .. a[n]

* kimenő adat: y

* munkarekesz: i

*/

STORE J1 MK:

STORE J2 EK:

STORE 0 i

MK: STOP - - / J1: tartalma kerül ide */*

KB: ADD 1 i

EB: J<=E KI: n

LOAD y -

EK: STOP - - / J2: tartalma kerül ide */*

MÓDK: ADD 11 MK:

ADD 21 EK:

JUMP MK: -

MÓDB: ADD 21 EK:

ADD 11 MK:

JUMP KB: -

KI: WRITE y -

STOP - -

y: -

i: -

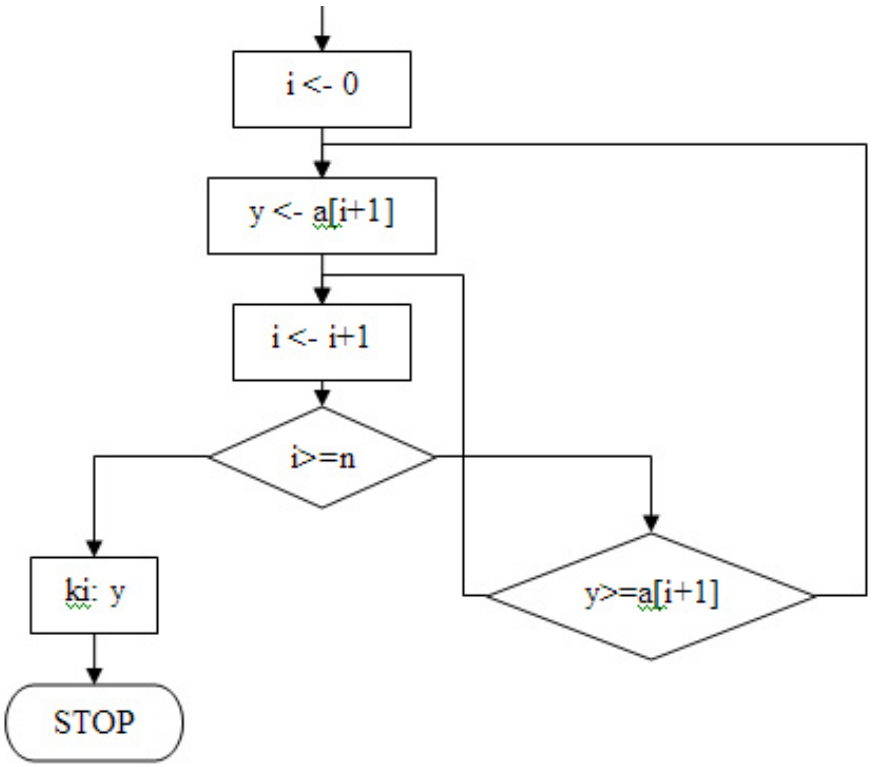
J1: STORE a1 y

J2: J<=E MÓDB a2

n: #10

a1: 5 a2: 4 6 4 2 3 1 5 2

A program folyamatábrája:



Prímszámkiíró program

```
/*************************************************************************
(c) Szofi Magyar-Amerikai Informatikai Oktató- és Továbbképző Központ, 2000
PROGRAMOZÁS ALAPJAI SEGÉDLET (176a/DEV_098/1.0) honlap: http://www.szofi.hu
*************************************************************************
*
* Prímszámok kiiratása N1-től és N2-ig
*
*/
STORE N1 i
P1: STORE i Prím_IN/
C: ADD C 0004
JUMP Prím -
J=0 P2 Prím_OUT
WRITE1 i -
P2: ADD 1 i
J>E P1 N2
STOP - -
N1: #1
N2: #100
```

```
N2: #100
i: -
/*
* Prím: Ha IN prím, OUT = 1, különben OUT = 0
*
* Algoritmus: A szám prím, ha kisebb, vagy egyenlő kettővel,
* illetve ha nem osztója 2-től a szám gyökéig egy természetes szám sem.
*/
Prím: MULT' 1 Prím_END
LOAD 2 -
J<=E Prím_TRUE Prím_IN
STORE Prím_IN SQR_IN
Prím_F1: ADD Prím_F1 0004
JUMP SQR -
STORE SQR_OUT Gyok
STORE 0 Prím_OUT
STORE 2 Prím_i
Prím_L1: STORE Prím_IN Prím_y
DIV Prím_i Prím_y
STORE Prím_y Egész_IN
Prím_F2: ADD Prím_F2 0004
JUMP Egész -
SUB Egész_OUT Prím_y
J=0 Prím_END Prím_y
ADD 1 Prím_i
J>E Prím_L1 Gyok
Prím_TRUE: STORE 1 Prím_OUT
Prím_END: -
Prím_IN: -
Prím_OUT: -
Prím_i: -
Prím_y: -
Gyok: -
/*
* SQR: OUT <- négyzetgyök(IN)
*/
SQR: MULT' 1 SQR_END
```

```
STORE 1 SQR_y
SQR_L: LOAD SQR_IN -
DIV' SQR_y SQR_x
ADD SQR_y SQR_x
DIV 2 SQR_x
SUB' SQR_y SQR_s
STORE SQR_x SQR_y
MULT SQR_s SQR_s
J<=E SQR_L SQR_e2
STORE SQR_x SQR_OUT
SQR_END: -
SQR_IN: -
SQR_OUT: -
SQR_e2: #0.000001
SQR_x: -
SQR_y: -
SQR_s: -
/* Egész: OUT <- Egészresz(IN) */
Egész: MULT' 1 Egész_END
STORE 0 Egész_MIN
STORE MaxEgész Egész_MAX
SUB Egész_MAX Egész_MIN
STORE 0 Egész_i
Egész_L1: STORE Egész_MAX Egész_y
SUB Egész_MIN Egész_y
DIV 2 Egész_y
ADD Egész_MIN Egész_y
LOAD Egész_IN -
J>E Egész_L2 Egész_y
STORE Egész_y Egész_MIN
JUMP Egész_L3 -
Egész_L2: STORE Egész_y Egész_MAX
Egész_L3: ADD 1 Egész_i
J>E Egész_L1 Egész_N
STORE Egész_MIN Egész_OUT
Egész_END: -
Egész_IN
```

Egész_IN: -

Egész_OUT: -

Egész_MIN: -

Egész_MAX: -

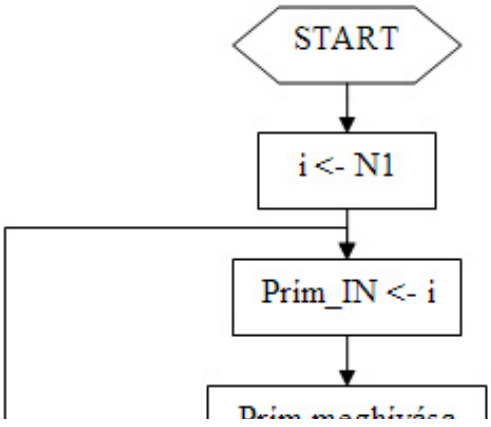
Egész_i: -

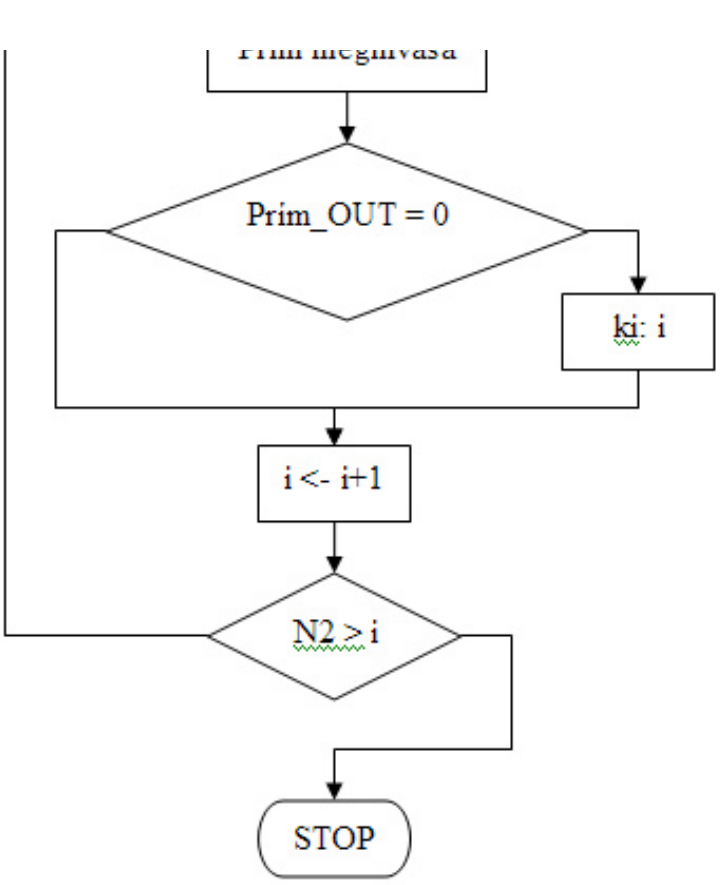
Egész_y: -

Egész_N: #24

MaxEgész: #8388608

A program folyamatábrája:




Prim_OUT = 0
ki: i
i <- i+1
N2 > i
STOP