Aplikace matematiky

Ladislav Koubek

Programující program Act 1c pro počítač Ural 2

Aplikace matematiky, Vol. 9 (1964), No. 2, 110–130

Persistent URL: http://dml.cz/dmlcz/102888

PROGRAMUJÍCÍ PROGRAM ACT 1c PRO POČÍTAČ URAL 2

LADISLAV KOUBEK

(Došlo dne 5. dubna 1963.)

V práci je popsán programující program ACT 1c, který má umožnit automatizaci programování pro samočinný počítač Ural 2.

Symbolický (základní) program zpracujeme na malém počítači LGP 30 nejprve ve strojový program tohoto počítače. Tento program lze odladit na LGP 30 a provést eventuální opravy v základním programu. Opravený základní program znovu zpracujeme na LGP 30, ale už ve strojový program pro počítač Ural 2.[1])

1. ÚVOD

Při práci byl využit programující program ACT 1 dodaný s počítačem LGP 30 výrobcem, užívaný (po jistých úpravách) pod označením ACT 1a resp. ACT 1b v Centru numerické matematiky Karlovy University k programování značné části úloh.

V programu ACT 1 lze totiž snadno obměňovat operační tabulku a kromě toho lze po změně některých relativních konstant vytvářet program s instrukcemi odlišnými od instrukcí LGP 30. Toho bylo využito při zpracování programu pro počítač Ural 2. Sestavuje se totiž nejprve program s instrukcemi Uralu 2, ale s adresami LGP 30 a teprve po zpracování celého programu jsou adresy LGP 30 nahrazeny adresami počítače Ural 2. Tento postup umožňuje v maximální míře využít vyzkoušeného programu ACT 1a a lze ho bez větších nesnází užít i pro kompilování programu pro libovolný jednoadresový počítač. Stačí jen vhodně pozměnit operační tabulku a algoritmus převodu adres.

Snahou při práci dále bylo využít v maximální možné míře knihovny standardních podprogramů užívané ve Výpočetní laboratoři ministerstva dopravy a spojů, kde je instalován počítač Ural 2 a s kterou autor práce při vypracování programu ACT 1c měl možnost spolupracovat.

[1]) Technická skupina Centra numerické matematiky KU pracuje na převodníku, který automaticky provede děrování filmové pásky pro Ural 2 z pásky stroje LGP 30. Po jeho dokončení odpadne i poslední zdroj chyb v programu, ruční děrování.

Protože cílem práce je program pro počítač Ural 2, budou v dalším uváděny odkazy na strojový kód, paměť atd. počítače Ural 2 a to, jak je u tohoto počítače obvyklé, v oktalové soustavě. Pomocná část, program pro LGP 30, nebude podrobně uváděna, pokud to nebude pro porozumnění textu nezbytné. Odkazy na kod a instrukce LGP 30 budou ovšem uváděny ve tvaru obvyklém pro tento počítač [1].

2. JAZYK ACT 1c

Jazyk ACT 1c, kterým programátor píše základní program, tj. program zpracovávaný programem ACT 1c na počítači LGP 30 ve strojový program ať už pro LGP 30 nebo Ural 2, vychází z pojmu aritmatické operace, jejímiž operandy mohou být konstanty, proměnné, hodnoty některých elementárních funkcí atd. Aby bylo možno popsat celý postup výpočtu na stroji, jsou zavedeny ještě nearitmetické operace, např. operace předání řízení, tisku výsledků, logických rozhodování atd., jejichž operandy mohou být opět konstanty, proměnné, ale i značky operátorů, parametry atd. Z těchto prvků se, podle níže uvedených syntaktických pravidel, skládají věty (uzavřené jednotky jazyka), operátory.[2])

Každé operaci je vzájemně jednoznačně přiřazen určitý identifikátor. Množina všech identifikátorů tvoří operační tabulku ACT 1c. Každému identifikátoru operace je dále přiřazena určitá posloupnost instrukcí, kterou na příslušné místo strojového programu zařadí programující program (viz odst. 6). Adresami v této posloupnosti instrukcí jsou konkrétní adresy, které identifikátorům operandů přiřadí opět program ACT 1c.

Operace jsou v jazyku ACT 1c obvykle dvouargumentové, tj. k vytvoření strojového programu je nutno u každé operace uvést i pravý a levý operand (na příklad $a'-'b'$; zápis $-'b'$ je nemožný), nebo jednoargumentové – identifikátor argumentu je pak vždy vpravo (na příklad $sin'\,a'$; zde *sin* je identifikátor operace – výpočtu hodnoty funkce sinus – a je identifikátorem příslušného argumentu), nebo nemají žádný argument (např. operace zastavení počítače má identifikátor *stop'*).

Z technických důvodů [1] se každý identifikátor může skládat nejvýše z 5 znaků psacího stroje LGP 30 a od ostatních identifikátorů musí být oddělen značkou konce slova, zvaným v literatuře LGP 30 ne zcela přesně podmíněný stop. V dalším budeme psát identifikátory včetně této značky konce slova, i když, z hlediska struktury vlastního jazyka ACT 1c, je bezvýznamná. Každý identifikátor z maximálně pěti znaků psacího stroje LGP 30 ukončený značkou konce slova nazveme v dalším přípustný identifikátor.

Syntaktická pravidla jazyka ACT 1 jsou jednoduchá:

1. Každý identifikátor musí být přípustným identifikátorem a musí být označením právě jednoho prvku programu (operace, proměnné, značky atd.).

[2]) Mezi aritmetické operace počítáme i operaci substituce a operace elementárních funkcí např. $sin\,\alpha$ atd., ačkoliv se zřejmě jedná vždy o obracení se k podprogramům.

2. Posloupnost identifikátorů tvoří větu jazyka, operátor, který je zpracováván ve strojový program jako jeden celek a jeho délku určuje programátor prázdným (tj. neobsahující žádný znak kromě znaku konce slova) identifikátorem konce operátoru.

3. Každý identifikátor operace má přiřazen určitý stupeň $n = 0, 1, 2, 3$. Identifikátor otevírající závorky [′ zvyšuje o 4 stupeň každého následujícího identifikátoru operace. Identifikátor uzavírající závorky]′ toto zvyšování stupně ruší. Maximálně můžeme stupeň identifikátoru zvýšit o 24 (maximálně lze tedy užít 6 do sebe uzavřených závorek).[3]) Výjimku tvoří dva identifikátory fiktivních operací *kdyz′* a *nast′*, které nemají přiřazen žádný stupeň a identifikátory *cykl2′*, *cykl′*, *pod′* a *ind′*, které musí (s příslušnými operandy) tvořit samostatný operátor.

4. Každý identifikátor operace spolu s příslušnými operandy tvoří výraz, jemuž je přiřazen stupeň rovný stupni identifikátoru operace s eventuálním zvýšením podle 3.

5. Operandem aritmetického výrazu je buď identifikátor proměnné, konstanty nebo aritmetický výraz stupně vyššího nebo rovného, a to podle těchto zásad:

V jednom operátoru vytváříme výrazy rekurentně sestupně podle stupňů a v případě rovnosti stupňů zleva napravo. Operandy výrazu nejvyššího stupně (nebo prvního z výrazů nejvyššího stupně zleva, je-li jich několik) jsou vždy identifikátory.

Oba identifikátory *kdyz′* a *nast′* fiktivních operací jsou při tom ignorovány a mohou být zařazeny na libovolné místo posloupnosti identifikátorů tvořících operátor.

6. Aritmetický operátor musí být aritmetickým výrazem stupně $4n$, $n = 0, \ldots, 6$.

7. Operandy nearitmetického výrazu mohou být buď aritmetické výrazy nebo identifikátory konstant, proměnných a značek.

8. Nearitmetický výraz nemůže být operandem žádného výrazu.

9. Nearitmetický operátor se skládá buď z jediného nearitmetického výrazu, nebo z aritmetického výrazu stupně $4n$, $n = 0, \ldots, 6$ a jednoho nearitmetického výrazu stupně $4n'$, $n' \leqq n$, nebo z posloupnosti nearitmetických výrazů, jejichž stupně zleva napravo nerostou.

10. Každý identifikátor proměnné s indexem musí být operandem jednoho pseudooperátoru dimense v záhlaví (viz odst. 4).

11. Každý identifikátor indexu musí být uveden v soupisu identifikátorů indexů v záhlaví (viz odst. 4).

12. Identifikátor proměnné s indexem může být operandem, jen je-li specifikován indexem.

13. Indexem může být buď identifikátor indexu uvedený v soupisu v záhlaví, nebo identifikátor konstanty.[4])

[3]) Každá závorka musí být ovšem uzavřena, tj. v každém operátoru musí být stejný počet identifikátorů [′ a]′.

[4]) V jazyce ACT 1c nelze užít jako indexu výraz, ani jedna proměnná nemůže mít dva indexy. Při práci s více indexy (na příklad při řešení úloh lineární algebry) je musíme vždy linearizovat. Lze tedy užívat jen lineární masivy a hodnota indexu je v podstatě jen relativní adresou prvku v masivu.

14. Každý operátor lze označit jednou (a jen jednou) značkou, jejíž identifikátor má tvar $s\alpha'$, kde $\alpha = 00, \ldots, 191$.

15. Operandem nearitmetické operace může být jen ta značka, kterou je označen některý z operátorů.

16. Konec základního programu je vyznačen prázdným operátorem.

3. ARITMETICKÉ OPERACE

Jak už bylo řečeno, pokládáme v jazyce ACT Ic za aritmetické operace nejen 4 základní aritmetické operace, ale také operaci substituce (dosazení hodnoty levého operandu do pravého) a operace výpočtu hodnot elementárních funkcí (seznam identifikátorů, které sem patří, je v odstavci 6), ačkoliv v tomto případě se vždy jedná vlastně o přechod na podprogram výpočtu hodnoty příslušné funkce. To můžeme dělat proto, že příslušné obrácení se k podprogramu je naprogramováno přímo programem ACT Ic a při tom podle syntaktických pravidel jazyka.

Protože počítač Ural 2 může pracovat jak v pohyblivé čárce (rozsah $|x| \leqq 2^{63}$) tak i v pevné čárce (rozsah $|x| < 1$) a kromě toho musíme pracovat také s indexy, jejichž hodnoty jsou vždy celá čísla ($|x| \leqq 99999$) vyjádřená v adresních jednotkách (přesněji řečeno dvojnásobcích adresních jednotek – viz odst. 3), musíme pro každou aritmetickou operaci mít 3 různé identifikátory (s výjimkou operací výpočtu elementárních funkcí, neboť knihovna podprogramů užívaná ve Výpočetní laboratoři min. dopravy obsahuje jen podprogramy v pohyblivé čárce).

Pro pohyblivou čárku jsou to identifikátory

$$+' \; -' \; x' \; /' \; .$$

Pro pevnou čárku

$$p+' \; p-' \; px' \; p/'$$

a pro práci s indexy

$$i+' \; i-' \; ix' \; .$$

Všechny tyto identifikátory mají zřejmý význam.

Užijeme-li identifikátory pro práci s indexy, předpokládáme ovšem, že operandy jsou celá čísla $|x| \leqq 99999$ a ovšem i výsledek takové operace musí být menší než 10^5 a dostaneme ho (podle posloupností instrukcí odpovídajících těmto identifikátorům – viz odst. 6) jako celé číslo.

Protože programy vytvořené programem ACT Ic pro Ural 2 umísťují všechny veličiny do úplných buněk, jsou všechny hodnoty indexů a celých konstant zobrazeny ve stroji ve dvojnásobcích adresních jednotek, neboť je užíváme hlavně pro modifikaci adres, které musí být vždy sudé [3]. Dělení indexů není zavedeno, protože při práci s indexy se mu lze snadno vyhnout a vytváření celé části podílu by bylo v důsledku zaokrouhlování na stroji Ural 2 značně obtížné.

Identifikátory substituce jsou opět 3, a to

$$;' p;' i;' .^{5})$$

Identifikátory

$$+' p+' i+' -' p-' i-'$$

mají stupeň 1.

Identifikátory

$$x' px' ix' /' p/'$$

mají stupeň 2.

Identifikátory

$$;' p;' i;'$$

mají stupeň 0.

Všechny identifikátory výpočtu elementárních funkcí, na příklad *sin'*, *cos'*, *tg'*, *log'*, ..., *abs'* a *pabs'* (k výpočtu absolutní hodnoty neužíváme podprogram a proto musíme mít dva identifikátory; pro výpočet absolutní hodnoty indexu lze užít identifikátor *pabs'*) mají stupeň 3.

Podle syntaktických pravidel 1÷15 jazyka ACT 1c je tedy nutno aritmetické výpočty

$$m = \frac{a+b}{c}\,; \quad m = ab - cd\,; \quad m = \frac{a}{bc}\,; \quad m = \sin(a-b)$$

programovat takto:

$$['a'+'b']'/'c';'m''\,,$$
$$a'x'b'-'c'x'd';'m''\,,$$
$$a'/'b'/'c';'m''\ \text{nebo}\ a'/'['b'x'c']';'m''\,,$$
$$sin'['a'-'b']';'m''\,,$$
atd.

Při programování v jazyce ACT 1c se snažíme, není-li to na závadu přehlednosti, aby stupně identifikátorů zleva napravo nevzrůstaly, neboť pak každý výraz je ihned operandem následujícího výrazu a tím odstraníme zapisování mezivýsledků do pomocných buněk paměti stroje, a tím přirozeně zkrátíme strojový program a zrychlíme výpočet.

V algoritmu ACT 1c není totiž záměna argumentů operace. Při tom jsou programovány výrazy v pořadí podle stupně sestupně a v případě výrazů téhož stupně zleva. Výsledek každé operace zůstává ve střadači a není-li levým operandem výrazu, který

[5]) Ve skutečnosti ovšem *p;'* a *i;'* jsou ekvivalentní (tj. program jim odpovídající je týž) a byly zavedeny jen pro úplnost.

bude programován v následujícím kroku, zasílá se do pomocné buňky paměti, kde je chráněn až do té doby, kdy se stane operandem některého výrazu.

Programujeme-li na příklad

$$c'x'['a'+'b']';'d'' ,$$

jsou stupně identifikátorů operací zleva 2, 5, 0, takže nejprve bude programován výraz $a'+'b'$. Ten má být pravým operandem výrazu $c'x'['a'+'b']$, nemůže být proto přímo použit a zasílá se do pomocné buňky, takže v dalším kroku programujeme výraz s levým operandem touto pomocnou buňkou. Výraz $c'x'['a'+'b']'$ je levým operandem dalšího programovaného výrazu $c'x'['a'+'b']';'d'$ a není proto přesílán do pomocné buňky.

Programujeme-li týž výpočet ve tvaru

$$['a'+'b']'x'c';'d'' ,$$

mají identifikátory operací stupně 5, 2, 0, takže každý výraz vyššího stupně je levým operandem následujícího výrazu a odpadá tedy přesílání do pomocných buněk. Tím ušetříme ve strojovém programu dvě instrukce, což může znamenat značnou úsporu času, je-li takový výpočet v mnohonásobně opakovaném cyklu. (Z tohoto hlediska je výhodnější první z uvedených možností programování výpočtu $m = a/bc$.)

Pro práci s indexy je výhodné některé konstanty nezavádět jako údaje až při práci programu, ale uvést je přímo v základním programu. (Na příklad krok a počet průběhů cyklu atd.) V tomto případě uvedeme v základním programu identifikátor příslušné konstanty (0', 1', ..., 99999') a píšeme na příklad

$$i'i+'1'i;'i'' .$$

Konstantám uvedeným v základním programu přiřadí ACT 1c určitou buňku (viz odst. 5), současně převede jejich dvojnásobek do dvojkové soustavy (jako celá čísla v adresních jednotkách) a toto dvojkové vyjádření umístí do přiřaděné buňky.

Chceme-li některou konstantu v pohyblivé čárce uvést v základním programu (např. π nebo některé fyzikální konstanty), musíme použít identifikátoru *pohyb'* ve spojení

$$n'pohyb'm';'a''$$

(také operace *pohyb'* je v jazyce ACT 1c aritmetickou operací), kde m je celé a $0 \leqq n \leqq 7$ udává počet desetinných míst konstanty m.

Na příklad zápisem

$$3'pohyb'7216';'a''$$

umístíme do buňky a číslo 7,216 ve standardním tvaru pohyblivé čárky (ovšem až při práci strojového programu; ACT 1c jen přiřadí buňky konstantám 3 a 7216 a jako celá čísla je převede do dvojkové soustavy).

4. ZNAČENÍ OPERÁTORŮ, PŘEDÁVÁNÍ ŘÍZENÍ

Každý operátor v základním programu lze označit značkou. Pro zjednodušení algoritmů programujícího programu má v jazyku ACT 1c identifikátor každé značky operátoru tvar $s\alpha'$, kde α je celé číslo $\alpha = 0, \ldots, 191$, takže v jednom základním programu lze označit maximálně 192 operátorů.

Jak už bylo řečeno, můžeme označit každý operátor, musíme však označit jen ty operátory, kterým chceme v průběhu práce strojového programu předávat řízení, neboť značka tohoto operátoru bude operandem alespoň jedné nearitmetické operace.

Operace předání řízení jsou v jazyce ACT 1c čtyři. Jejich identifikátory jsou:

jdi' pro nepodmíněné předání řízení,

pak' pro podmíněné předání řízení,

vrat' pro nepodmíněné předání řízení s návratem,

pod' pro nepodmíněné předání řízení programu psanému ve strojovém kódu.

První tři jsou identifikátory nearitmetických jednooperandových výrazů, jejichž operandem je vždy značka. Např. *jdi'sα'* atd. Ve strojovém kódu odpovídá tomuto výrazu jediná instrukce $22x_1x_2x_3x_4\ 0$, kde $x_1x_2x_3x_4$ je první adresa operátoru vyznačeného $s\alpha$.

Výraz *pak'sα'* je programován rovněž jedinou instrukcí, a to $21\ x_1 \ldots x_4\ 4$.[6]) To znamená, že operátor $s\alpha$ dostane řízení, právě když $\omega = 0$. Je-li $\omega = 1$, převezme řízení následující operátor programu. U LGP 30 odpovídá výrazu *pak'sα'* instrukce $T x_1x_2x_3x_4$, tj. operátor $s\alpha$ dostane řízení, právě když $\omega = 1$.

Signál ω, podle kterého se řídí, zda dojde nebo nedojde k předání řízení operátoru $s\alpha'$ vytváříme operacemi srovnávání veličin. V základním programu toto srovnání zapisujeme posloupností identifikátorů

kdyz'a'vetsi'b'pak'sα''
kdyz'a'mensi'b'pak'sα''
kdyz'a'rovno'b'pak'sα''

pro práci v pohyblivé čárce,

kdyz'a'vet'b'pak'sα''
kdyz'a'men'b'pak'sα''
kdyz'a'rov'b'pak'sα''

pro práci s indexy a v pevné čárce.

Identifikátor *kdyz'* je pomocný a slouží jen k tomu, aby zápis byl přehlednější. Ve strojovém kódu bezoperandovému výrazu *kdyz'* neodpovídá žádná instrukce.[7])

[6]) Instrukce $21\ x_1 \ldots x_4\ 0$ není při programování v jazyce ACT 1c použita.

[7]) Zápisy *kdyz'a'vetsi'b'pak'sx''* a *a'vetsi'b'pak'sx''* jsou ekvivalentní — vytvoří se podle nich týž program.

Identifikátory *vetsi'*... mají stupeň 3. Operandy příslušných výrazů mohou být buď identifikátory nebo aritmetické výrazy (stupně většího než 3), takže při srovnávání aritmetických výrazů musíme ovšem užít závorek. Na příklad

$$kdyz'['a'+'b']'vetsi'c'pak's\alpha''\ .$$

Identifikátory *vetsi'* a *vet'* programujeme *ostrou* nerovnost (signál $\omega = 0$ pro Ural 2 resp. $\omega = 1$ pro LGP 30 se vytvoří právě když hodnota levého operandu je větší než hodnota pravého).

Identifikátory *mensi'* a *men'* programujeme *neostrou* nerovnost (signál $\omega = 0$ pro Ural 2 resp. $\omega = 1$ pro LGP 30 se vytvoří, právě když hodnota levého operandu je menší nebo rovna hodnotě pravého operandu).

Zápis

$$vrat's\alpha''$$

bude zaprogramován instrukcí

$$22\ x_1x_2x_3x_4\ 4\ .$$

Operace *vrat'sα'* slouží tedy k předání řízení na podprogram s návratem. Podprogram, kterému tímto způsobem předáváme řízení, musí mít ovšem tvar

$$s\alpha'jdi's\alpha''\ldots''jdi's\alpha''\ ,$$

neboť do jeho první buňky bude instrukcí $22\ x_1 \ldots x_4\ 4$ zapsána instrukce návratu a řízení bude předáno až jeho druhé buňce. Po ukončení práce podprogramu pak musíme předat řízení první buňce, což provedeme právě tím, že podprogram zakončíme operátorem *jdi'sα''*.

Identifikátor *pod'* užíváme, chceme-li užívat podprogramy psané ve strojovém kódu. Pro podprogram psaný ve strojovém kódu zvolíme libovolný název z přípustných identifikátorů, na příklad *progr'*. V záhlaví rezervujeme masiv pro umístění n buněk strojového kódu podprogramu do paměti stroje zápisem:

$$dim'progr'm''\ ,$$

kde $m = [(n + 1)/2]$ pro stroj Ural 2 a $m = n$ pro LGP 30. (Obsahuje-li užitý program pro Ural 2 jiný počet instrukcí než odpovídající program pro LGP 30, musíme ovšem buď rezervovat větší masiv a nebo po odladění na LGP 30 změnit záhlaví při kompilování definitivního programu pro Ural 2.)

V místě, odkud se chceme k podprogramu obrátit, uvedeme pak nearitmetický operátor (viz bod 3 odst. 1)

$$pod'progr'a'b'\ldots''\ .$$

Druhý identifikátor je vždy identifikátorem zvoleného názvu podprogramu, další pak identifikátory faktických parametrů, s nimiž má podprogram pracovat. První z faktických parametrů je vždy proměnná, další buď proměnná nebo značky operátorů.

Ze strojového programu (odst. 6) vytvořeného podle těchto instrukcí je ovšem zřejmo, že podprogramy ve strojovém kódu musí být sestaveny speciálně tak, aby správně aplikovaly uvedené faktické parametry.

Mezi operace předání řízení můžeme ještě zařadit výhybku, kterou programujeme zápisem

$$nast's\alpha'na's\beta''\ .$$

Při práci strojového programu pak jen měníme první instrukci operátoru $s\alpha$ na instrukci bezpodmínečného předání řízení na operátor $s\beta$. Operátor $s\alpha$ má ovšem obvykle tvar

$$s\alpha'jdi's\alpha''\ .$$

Výhody takto jednoduchého programování výhybky jsou zřejmé.

5. PROMĚNNÉ S INDEXEM, CYKLY, ZÁHLAVÍ ZÁKLADNÍHO PROGRAMU

Při programování je velmi důležitým problémem modifikace adres při práci s množinami proměnných.

V jazyku ACT 1c jsou k účelům modifikace adres zavedeny proměnné s indexem. Identifikátorem proměnné s indexem může být libovolný přípustný identifikátor (ovšem s výjimkou identifikátorů operací, proměnných a značek) např. a'.

Masiv buněk paměti, potřebný pro umístění množiny hodnot označených společným identifikátorem proměnné s indexem, rezervujeme tím, že v záhlaví základního programu uvedeme pseudooperátor dimense

$$dim'a'n''\ .$$

ACT 1c pro každou proměnnou s indexem rezervuje n úplných buněk a zapamatuje i adresy a_0 a a_n.

Každou proměnnou s indexem musíme v základním programu specifikovat právě jedním indexem. Stroj totiž v každém výpočtu může pracovat jen s jedinou hodnotou. Index pak určuje tuto jedinou hodnotu z množiny umístěné v paměti.

Je-li indexem identifikátor konstanty m (přirozeně $m \leqq n$), budeme pracovat stále s určitou hodnotou a_m a ve strojovém programu bude uvedena konkrétní adresa $a_m = a_0 + m$.

Je-li indexem identifikátor indexu, musí být především tento identifikátor uveden v soupise identifikátorů indexů v záhlaví základního programu v pseudooperátoru

$$index'i'\ldots j''$$

(počet identifikátorů indexů není omezen) a ve strojovém programu bude uvedena buď konkrétní adresa a_0 nebo a_n a instrukce, která provede modifikaci podle okamžité hodnoty indexu.

Počítač Ural 2 má jeden index-registr [3] a modifikace adres se provádí v záporných instrukcích tak, že od adresy uvedené v programu odečítáme okamžitý obsah index-registru.

Kromě toho lze provádět modifikaci adres instrukcí 30. Modifikaci pak počítač provádí tak, že k instrukci následující za instrukcí 30 $x_1 \ldots x_4$ 0 přičte obsah buňky $x_1 \ldots x_4$.

Počítač LGP 30 nemá ani index-registr, ani instrukci obdobnou instrukci 30 a modifikace adres je vždy programována – při programování v jazyce ACT 1c ovšem automaticky, jakmile použijeme proměnnou s indexem, a to způsobem odpovídajícím buď modifikaci užitím index-registru nebo instrukce 30 počítače Ural 2.

První z identifikátorů uvedených v soupise identifikátorů indexů v záhlaví je vždy určen k modifikaci užitím index-registru. Je-li tedy indexem první identifikátor soupisu, dostaneme ve strojovém programu zápornou instrukci s nejvyšší adresou masivu.

Je-li indexem některý z dalších identifikátorů soupisu, modifikujeme adresu operací 30, tj. ve strojovém programu bude dvojice instrukcí 30⟨*index*⟩0 Φ⟨*nejnižší adresa*⟩4 kde ⟨*index*⟩ znamená buňku přiřazenou příslušnému identifikátoru indexu (v ní je vždy okamžitá hodnota indexu), Φ je některá z instrukcí Uralu 2 a ⟨*nejnižší adresa*⟩ je počáteční adresa masivu.

Dosazení počáteční hodnoty do index-registru programujeme nearitmetickým operátorem

$$ind'm's\alpha'' ,$$

kde druhý identifikátor m je buď identifikátorem konstanty nebo proměnné. Značka $s\alpha'$ určuje, kam má být instrukcí 27 přeslán dosavadní obsah index-registru. Ze soustavy instrukcí počítače Ural 2 plyne, že má být přeslán do druhé buňky operátoru $s\alpha'$, který ovšem bude operátorem cyklu řízeného index-registrem (je to operátor složený z jediného nearitmetického výrazu)

$$s\alpha'cykl\ 2's\beta'' ,\ ^{8})$$

kde $s\beta'$ je značka prvního operátoru cyklu.

Cyklus programovaný právě uvedeným způsobem ovšem probíhá $m + 1$-krát [4] a po jeho ukončení (při práci strojového programu) se obsah indexregistru obnoví na stav, který měl před užitím operace $ind'm's\alpha''$, takže lze snadno programovat libovolný počet cyklů v cyklu.

Chceme-li programovat cyklus řízený jinak než index-registrem, užijeme nearitmetického operátoru tvaru

$$cykl'j'k'n's\alpha'' ,\ ^{8})$$

[8]) Oba tyto zápisy mají v jazyce ACT 1c do jisté míry výjimečné postavení a musí být vždy samostatnými operátory (viz pravidlo 3, odst. 1).

kde j' je identifikátorem proměnné řídící cyklus (může, ale nemusí to být identifikátor indexu), k' a n' jsou identifikátory proměnných nebo konstant. Cyklus pak probíhá tak, že při každém průběhu se dosadí $j + k \Rightarrow j$ (předpokládáme ovšem, že jde o celá čísla a proto i toto sčítání je sčítáním podle identifikátoru $i+'$) a srovnává j s n. Pokud $j \leqq n$ dostává řízení operátor $s\alpha'$, pro $j > n$ cyklus končí. Počáteční hodnotu j musíme ovšem určit před vstupem do cyklu.

Jak už bylo řečeno, chceme-li používat proměnné s indexem a indexy v základním programu, musíme sestavit záhlaví základního programu, které se skládá z pseudooperátorů dimense (maximálně 11) a jednoho soupisu identifikátorů indexů.

Protože na počítači LGP 30 je možno pracovat v pohyblivé čárce jen užitím podprogramů, je nutno rezervovat pro ně masiv 640 buněk. V tomto masivu jsou všechny operace pohyblivé čárky, převody 2→10 a 10→2 v pevné i pohyblivé čárce a pomocný program pro ladění programů (viz odst. 7). Nejsou sem však zahrnuty podprogramy elementárních funkcí. Chceme-li proto pracovat také s elementárními funkcemi, musíme tento masiv zvětšit o buňky potřebné k umístění programů jejich výpočtu. Potřebné údaje jsou uvedeny v práci [1].

U počítače Ural 2 tvoří všechny podprogramy převodů 2→10 a 10→2 jakož i výpočtu všech elementárních funkcí jeden masiv, který vždy umísťujeme do paměti (viz odst. 5), takže obdobný masiv rezervovat nemusíme. Abychom nemusili měnit záhlaví, rezervujeme místo na podprogramy LGP 30 pseudooperátorem dimense

$$dim'comp'm'' ,$$

podle kterého se rezervuje masiv o m buňkách jen při zpracovávání programu pro LGP 30. Při zpracovávání programu pro Ural 2 tento masiv (s identifikátorem $comp'$) není rezervován. Všemi ostatními pseudooperátory dimense rezervujeme masiv m buněk jak pro LGP 30, tak pro Ural 2 (pro tento počítač jsou to ovšem vždy úplné buňky).

6. ROZDĚLENÍ PAMĚTI, OPERACE TYPU *čti*

Rozdělení paměti je pevné jak pro počítač Ural 2, tak pro počítač LGP 30.

U Uralu 2 je v buňce 0000 vždy umístěna 0; buňky 0002 – 0034 jsou pracovními buňkami strojového programu. V buňkách 0036 – 0067 je umístěna část standardního programu a některé konstanty nutné pro práci strojového programu sestaveného ACT 1c (viz odst. 6). V buňkách 6700 – 7777 je knihovna standardních programů počítače Ural 2 (užívaná ve Výpočetní laboratoři ministerstva dopravy a spojů v Praze).

Strojový program začíná v buňce 0070. Ihned za poslední buňkou strojového programu (poslední buňka je vždy lichá, v případě, že by vyšla poslední instrukce do sudé

buňky, zařadí ACT 1c do další liché buňky nulovou instrukci) jsou v úplných buňkách dvojkové ekvivalenty celých nezáporných konstant, jejichž identifikátory byly užity v základním programu, takže program a tyto relativní konstanty lze do počítače zavést jako jeden masiv. Za relativními konstantami jsou buňky přiřazené jednotlivým identifikátorům proměnných a indexů uvedených v soupise identifikátorů v záhlaví, a to v pořadí, ve kterém se jejich identifikátory poprvé vyskytují v základním programu.

Masivy přiřazené identifikátorům proměnných s indexem jsou řazeny sestupně počínaje buňkou 6676 včetně v pořadí příslušných pseudooperátorů dimense. Před poslední masiv je zařazen ještě pomocný masiv tolika pomocných úplných buněk, kolikrát bylo použito v základním programu operací typu *čti* (tj. operací s identifikátory *cti'pcti'icti'*), jejichž operandem je proměnná bez indexu.

Do stroje Ural 2 lze totiž vstupní údaje zavádět jen skupinovou instrukcí a je prakticky vyloučeno programovat zavádění jen po jedné hodnotě (velká spotřeba pásky, obtížná kontrola zavádění i obtížná příprava pásky). Na druhé straně by požadavek, aby všechny buňky přiřazené jednotlivým zaváděným hodnotám tvořily lineární masiv, příliš komplikoval algoritmus programu ACT 1c. Proto byl volen postup, při kterém všechny vstupní údaje zavedeme do stroje jako jeden masiv. Při tom dvojkodesítkový kód konstant proměnných s indexem zavedeme do příslušných buněk (počátek každého masivu udá na konci práce ACT 1c) a proměnných bez indexu do pomocného masivu (jeho počátek udá opět ACT 1c), a to sestupně v pořadí, v němž jsou operace typu *čti* uvedeny v základním programu (tj. první vstupní hodnota přijde do poslední buňky pomocného masivu atd.). Instrukcí typu *čti*, jejímž operandem je proměnná s indexem, pak jen převedeme příslušný kód z dvojkodesítkové soustavy do dvojkové (ať už v pevné nebo pohyblivé čárce nebo jako celé číslo) a umístíme zpět do téže buňky. U k-té operace typu *čti*, jejímž identifikátorem je proměnná bez indexu, bereme dvojkodesítkový kód z $(n-k+1)$-té buňky pomocného masivu a teprve po převodu do dvojkové soustavy umisťujeme do buňky přiřazené identifikátoru proměnné. (Upozorňujeme znovu, že počítáme jen ty operace typu *čti*, kde operandem je proměnná bez indexu.)

7. STROJOVÝ KÓD PŘIŘAZENÝ IDENTIFIKÁTORŮM OPERACÍ

Pro přehled uvedeme v tomto odstavci všechny identifikátory operací jazyka ACT 1c s příslušným stupněm a odpovídajícím úsekem strojového programu, které podle nich vytvoří ACT 1c. Strojový kód ovšem uvádíme jen v tom tvaru, jako by každý výraz tvořil samostatný operátor, tj. bez uvádění eventuálních pomocných buněk, které se v programu objeví při programování složitějších operátorů v důsledku postupného programování výrazů podle syntaktických pravidel jazyka ACT 1c. Adresy při tom budeme psát symbolicky, např. $\langle a \rangle$ znamená adresu přiřazenou identifikátoru a'.

Aritmetické operace:

Identifikátor	Stupeň	Výraz	Strojový kód	Poznámka
;′	0	a′;′b″	42 ⟨a⟩ 4 56 ⟨b⟩ 4	
i;′, p;′	0	a′i;′b″ a′p;′b″	02 ⟨a⟩ 4 16 ⟨b⟩ 4	
+′	1	a′+′b′	42 ⟨a⟩ 4 41 ⟨b⟩ 4	
−′	1	a′−′b′	42 ⟨a⟩ 4 43 ⟨b⟩ 4	
i+′, p+′	1	a′i+′b′ a′p+′b′	02 ⟨a⟩ 4 01 ⟨b⟩ 4	
i−′, p−′	1	a′i−′b′ a′p−′b′	02 ⟨a⟩ 4 03 ⟨b⟩ 4	
x′	2	a′x′b′	42 ⟨a⟩ 4 46 ⟨b⟩ 4	
/′	2	a′/′b′	42 ⟨a⟩ 4 47 ⟨b⟩ 4	
px′	2	a′px′b′	02 ⟨a⟩ 4 06 ⟨b⟩ 4	
p/′	2	a′p/′b′	02 ⟨a⟩ 4 07 ⟨b⟩ 4	
ix′	2	a′ix′b′	02 ⟨a⟩ 4 06 ⟨b⟩ 4 11 0021 4	
pohyb′	3	m′pohyb′n′	02 ⟨n⟩ 4 56 0002 4 30 ⟨m⟩ 4 46 0040 4	
abs′	3	abs′a′	42 ⟨a⟩ 4 10 0060 4	
pabs′	3	pabs′a′	02 ⟨a⟩ 4 10 0060 4	

Strojový program vytvořený podle všech identifikátorů výpočtu hodnot elementárních funkcí má tvar

42 ⟨*a*⟩ 4 ,

22 α 4 ,

kde α je vstupní adresou příslušného podprogramu [4]. Jsou to identifikátory (všechny stupně 3):

odmoc', *sin'*, *cos'*, *tg'*, *ctg'*, *asin'*, *acos'*, *atg'*, *actg'*, *exp'*, *log'*

(všechny tyto identifikátory mají zřejmý význam; exponenciela má základ *e*, log je přirozený).

Nearitmetické operace:

jdi'	0	*jdi'sα''*	22 ⟨*sα*⟩ 0	⟨*sα*⟩ znamená první instrukci operátoru označeného *sα*
vrat'	0	*vrat'sα''*	22 ⟨*sα*⟩ 4	
pak'	0	*pak'sα''*	21 ⟨*sα*⟩ 4	
kdyz'	nemá			jen pro přehlednější zápis
vetsi'	3	*kdyz'a'vetsi'b'pak'sα''*	42 ⟨*a*⟩ 4 43 ⟨*b*⟩ 4 03 0036 4 21 ⟨*sα*⟩ 4	ostrá nerovnost (pohyblivá čárka)
mensi'	3	*kdyz'a'mensi'b'pak'sα''*	42 ⟨*b*⟩ 4 43 ⟨*a*⟩ 4 21 ⟨*a*⟩ 4	neostrá nerovnost (pohyblivá čárka)
rovno' *rov'*	3 3	*kdyz'a'rovno'b'pak'sα''* *kdyz'a'rov'b'pak'sα''*	02 ⟨*a*⟩ 4 14 ⟨*b*⟩ 4 21 ⟨*s*⟩ 4	v obou případech jde o srovnání po *bitech* (pevná i pohyblivá čárka)
vet'	3	*kdyz'a'vet'b'pak'sα''*	02 ⟨*a*⟩ 4 03 ⟨*b*⟩ 4 03 0036 4 21 ⟨*sα*⟩ 4	ostrá nerovnost (pevná čárka)
men'	3	*kdyz'a'men'b'pak'sα''*	02 ⟨*b*⟩ 4 03 ⟨*a*⟩ 4 21 ⟨*sα*⟩ 4	neostrá nerovnost (pevná čárka)
nast'	nemá			jen pro přehlednější zápis

na′	3	*nast′sα′na′sβ″*	ϱ: 02 $\varrho+3$ 0 $\varrho+1$: 16 $\langle s\alpha\rangle$ 0 $\varrho+2$: 22 $\varrho+4$ 0 $\varrho+3$: 22 $\langle s\beta\rangle$ 0	adresy závisí na umístění programu v paměti, které proto uvádíme
cti′	3	*cti′a″*	02 ⟨*m*-tá buňka pomoc. masivu⟩ 4 22 7536 4 56 ⟨*a*⟩ 4	
		cti′a′i″	−02 $\langle a_m\rangle$ 4 22 7536 4 −56 $\langle a_m\rangle$ 4	*i′* je identifikátor prvního indexu; a_m je poslední adresa masivu
		cti′a′j″	30 ⟨*i*⟩ 0 02 $\langle a_0\rangle$ 4 22 7536 4 30 ⟨*i*⟩ 0 56 $\langle a_0\rangle$ 4	*j′* není identifikátorem prvního indexu; a_0 je první adresa masivu
icti′	3	*icti′a″*	02 ⟨*m*-tá buňka pomoc. masivu⟩ 4 22 0060 4 16 ⟨*a*⟩ 4	
pcti′	3	*pcti′a″*	02 ⟨*m*-tá buňka pom. masivu⟩ 4 22 7566 4 16 ⟨*a*⟩ 4	
pis′	3	*pis′a″*	42 ⟨*a*⟩ 4 22 7611 4 32 0001 0	
ppis′	3	*ppis′a″*	02 ⟨*a*⟩ 4 22 7677 4 32 0001 0	
ipis′	3	*ipis′a″*	02 ⟨*a*⟩ 4 22 0064 4 32 0001 4	
radek′	0	*radek″*	33 0001 0	mezera v tisku
stop′	0	*stop″*	37 0000 0	zastavení stroje
stop 4′	0	*stop* 4″	23 0001 0 37 0000 0	zastavení podle 1. klíče
stop 8′	0	*stop* 8″	23 0002 0 37 0000 0	zastavení podle 2. klíče

−pos′	0	*−pos′a″*	16 0002 4 02 ⟨*a*⟩ 4 11 0101 4 16 0004 4 02 0002 4 30 0004 0 11 0100 4	posuv obsahu střadače vpravo o *a* míst; pomocný posuv konstanty *a* je nutný, neboť jinak bychom mohli posouvat jen o sudý počet míst
+pos′	0	*+pos′a″*	16 0002 4 02 ⟨*a*⟩ 4 11 0101 4 16 0004 4 02 0002 4 30 0004 0 11 0000 4	posuv obsahu střadače vpravo o *a* míst
extr′	0	*extr′a″*	12 ⟨*a*⟩ 4	logický průnik
ctihx′	0	*ctihx′a″*	02 ⟨*m*-tá adresa pom. masivu⟩ 4 16 ⟨*a*⟩ 4	zavádění konstant psaných přímo v dvojkové soustavě
ber′	0	*ber′a″*	02 ⟨*a*⟩ 4	
dej′	0	*dej′a″*	16 ⟨*a*⟩ 4	

Následující operace musí být programovány jako samostatný operátor; proto jim není přiřazen žádný stupeň:

ind′	*ind′m′sα′*	27 ⟨*sα*⟩ +1 0 30 ⟨*m*⟩ 0 25 0000 4	*m′* je identifikátor proměnné
ind′	*ind′k′sα′*	27 ⟨*sα*⟩ +1 0 25 k_8 4	*k′* je identifikátor konstanty, k_8 je její oktalové vyjádření, které ovšem provede ACT 1c
cykl 2′	*cykl 2′sα″*	24 ⟨*sα*⟩ 0 00 0000 0	
cykl′	*cykl′i′k′m′sα*	02 ⟨*i*⟩ 4 01 ⟨*k*⟩ 4 16 ⟨*i*⟩ 4 02 ⟨*m*⟩ 4 03 ⟨*i*⟩ 4 21 ⟨*s*⟩ 4	

pod' *pod'název'a'b'...c''* 42 ⟨*a*⟩ 4
22 ⟨*název*⟩ 4
42 ⟨*b*⟩ 4
⋮
42 ⟨*c*⟩ 4

Jak už bylo řečeno, v buňkách 0036–0067 je pomocný program a relativní konstanty používán při práci strojového programu. Je to následující program:

0036	00	0000	0	I 39 (pro operace *vetsi* a *vet*)
37	00	0000	4	
40	40	0000	0	2^{17}
41	00	0011	0	
42	63	1463	0	$2^{17} . 10^{-1}$
43	63	1507	0	
44	50	7534	0	$2^{17} . 10^{-2}$
45	50	7505	4	
46	40	6111	4	$2^{17} . 10^{-3}$
47	72	2704	0	
50	64	3334	0	$2^{17} . 10^{-4}$
51	−35	3002	0	
52	51	7426	4	$2^{17} . 10^{-5}$
53	61	0700	4	
54	41	4336	4	$2^{17} . 10^{-6}$
55	−64	0541	0	
56	65	5376	0	$2^{17} . 10^{-7}$
57	−23	2543	0	
60	00	0000	0	program pro *icti'*
61	22	7572	4	
62	11	0017	4	
63	22	0060	0	
64	00	0000	0	program pro *ipis'*
65	11	0117	0	
66	22	7666	4	
67	22	0064	0	

Význam konstanty v úplné buňce 0036 je zřejmý z popisů úseků programů přiřazených identifikátorům *vetsi'* a *vet'*. Konstanty v buňkách 0040÷0056 užíváme pro převod celočíselných konstant (eventuálně i hodnot indexů vypočítaných v průběhu práce strojového programu) na standardní tvar pohyblivé čárky. Tyto celočíselné hodnoty jsou totiž ve stroji zobrazeny ve tvaru $a . 2^{-17}$ a užitím normalisační instrukce 56 0002 4 dostáváme tedy jen číslo $a . 2^{-17}$, které je proto nutno vynásobit

jednou z uvedených konstant (závislou na počtu myšlených desetinných míst – viz odst. 2), abychom dostali žádané číslo.

Úseky programu v buňkách 0060÷0063 a 0064÷0067 musíme zařadit, protože podprogramy pro převod celých čísel do dvojkové soustavy a naopak, obsažené v knihovně standardních programů Výpočetní laboratoře ministerstva dopravy a spojů, umisťují jednotky přivedeného čísla do 32. bitu a je tedy nutný posuv, neboť v programech vytvořených užitím ACT 1c předpokládáme, že jednotky jsou umístěny v 17. bitu.

8. POSTUP PRÁCE PŘI POUŽITÍ ACT 1C

Po sestavení základního programu můžeme přikročit ke kompilaci na počítači LGP 30 programem ACT 1c. Obvykle, jak už bylo řečeno, vytváříme nejprve strojový program pro počítač LGP 30. Postup práce je stejný jako při práci s programem ACT 1a resp. ACT 1b. Podrobný návod nalezne čtenář v práci [2], kde je popsána i možnost použití kontrolního programu usnadňujícího odlaďování strojového programu.

Programující program při tom kontroluje syntaktickou správnost základního programu. Chybu signalizuje tak, že vypíše *e* a číslo chyby. Seznam těchto kontrol a možnosti odstranění chyb nalezneme opět v práci [2].

Při kompilování programu pro Ural 2 by se už žádné formální chyby neměly objevit. Protože však při eventuálních opravách základního programu mohou vzniknout písařské chyby, je kontrola syntaktické správnosti základního programu prováděna znovu a i chyby jsou signalizovány znovu.

Po ukončení vlastní kompilace se počítač LGP 30 zastaví a po novém startu vypisuje strojový program pro počítač Ural 2. Při tom můžeme volit čtvrý formát zápisu programu:

1. Píšeme 8 instrukcí na jeden řádek. Na začátku každého řádku píšeme (červeně) číslo buňky, v níž je umístěna první instrukce řádku.
2. Píšeme na každý řádek jen jednu instrukci a (červeně) její umístění.
3. Píšeme 8 instrukcí na jeden řádek, ale nevypisujeme jejich umístění.
4. Píšeme bez uvádění umístění každou instrukci na zvláštní řádek.

Změnu formátu ovládáme tlačítky 6-BIT EINGABE a SPRUNG počítače LGP 30. Je-li zapnuto tlačítko 6-BIT EINGABE, píšeme každou instrukci na nový řádek. Je-li zapnuto tlačítko SPRUNG, nevypisujeme umístění programu.

Po vypsání poslední instrukce napíše ACT 1c ještě relativní konstanty (bez ohledu na tlačítko 6-BIT a SPRUNG každou na zvláštní řádek a bez uvedení umístění) a kontrolní součet celého programu (a tvaru dvou instrukcí) včetně relativních konstant (počínaje buňkou 0070). Po té se počítač znovu zastaví.

Po opětném startu napíše některé údaje o strojovém programu. Nejprve je to *i* 0070 udávající (pevnou) první buňku programu; pak *f* a adresu poslední instrukce programu a konečně *k* a adresu poslední buňky obsazené relativními konstantami. Dále napíše

počáteční adresy všech masivů (v pořadí, v němž jsou v záhlaví příslušné pseudooperátory dimense) a počáteční adresu pomocného masivu pro instrukci typu *čti*. Konečně vypíše všechny použité značky a první adresy operátorů jimi označených.

Po opětném startu provede počítač skok na čpoáteční adresu programu ACT 1c, takže můžeme ihned zpracovávat další program (ovšem pro Ural 2).

9. PŘÍKLAD PROGRAMU SESTAVENÉHO ACT 1C

```
dim'comp'640''
dim'a'901''
dim'b'31''
index'i'j''
s00'icti'n''n'ix'n'i;'n2''
ind'n2's02''jdi's02''
s01'cti'a'i''
s02'cykl2's01''
1'i;'j''
s03'cti'b'j''
cykl'j'1'n's03''
radek''
ind'n2's06''
s04'1'i;'j''0';'suma''jdi's06''
s05'a'i'x'b'j'+'suma';'suma''
cykl'j'1'n's06''
pis'suma''jdi's04''
s06'cykl2's05''
stop''jdi's00'''
0070  02 3166 4  22 0060 4  16 0172 4   02 0172 4  06 0172 4   11 0021 4   16 0174 4  27 0107 0
0100  30 0174 0  25 0000 4  22 0106 0  −02 6676 4  22 7536 4  −56 6676 4   24 0103 0  00 0000 0
0110  02 0162 4  16 0170 4  30 0170 0   02 3170 4  22 7536 4   30 0170 0   56 3170 4  02 0170 4
0120  01 0162 4  16 0170 4  02 0172 4   03 0170 4  21 0112 4   34 0001 0   27 0156 0  30 0174 0
0130  25 0000 4  02 0162 4  16 0170 4   42 0164 4  56 0176 4   22 0155 0  −42 6676 4  30 0170 0
0140  46 3170 4  41 0176 4  56 0176 4   02 0170 4  01 0162 4   16 0170 4   02 0172 4  03 0170 4
0150  21 0155 4  42 0176 4  22 7611 4   32 0001 0  22 0131 0   24 0136 0   00 0000 0  37 0000 0
0160  22 0070 0  00 0000 0
00 0002 0 00 0000 0
00 0000 0 00 0000 0
Σ00  0004 4 53 1045 0
i 0070 f 0161 k 0165
3266
3170
 3166
s00 0070
s01 0103
s02 0106
s03 0112
s04 0131
s05 0136
s06 0155
```

Jako příklad je uveden jen velmi jednoduchý program součinu matice s vektorem. Domníváme se, že uvedený program nepotřebuje žádný komentář.

Poznamenejme jen, že program nultých zón pro zavedení programu a číselného materiálu je ovšem nutno napsat ručně.

10. ZÁVĚR

Program ACT 1c byl v praxi vyzkoušen na počítačích LGP 30 v Centru numerické matematiky Karlovy university a Ural 2 ve Výpočetní laboratoři ministerstva dopravy a spojů.

Sestaven byl program výpočtu vlastních čísel a vektorů reálné symetrické matice 22. stupně Jacobiho metodou. Strojový program končil v buňce 1147 a relativní konstanty v buňce 1155.

K vytvoření obou programů pro LGP 30 i Ural 2 a odladění stroj. programu na LGP 30 na matici 3. stupně se spotřebovalo 2 hodin 37 minut strojového času. Vlastní výpočet (včetně tisku výsledné matice a matice vlastních vektorů) trval asi 10 minut.

Podotkněme, že na počítači LGP 30 by týž výpočet trval asi 75 strojových hodin.

Výsledky odpovídaly předepsané přesnosti rovné normě matice násobené 10^{-8}. Největší vlastní číslo bylo asi 6,6, nejmenší asi $4 \cdot 10^{-4}$.

Předložená práce je pouze ukončením první etapy pokusu o automatizaci programování na počítači Ural 2 užitím (pomocného) počítače LGP 30. Programem ACT 1c můžeme totiž zatím zpracovávat jen programy, které, včetně všeho číselného materiálu, lze najednou umístit do operativní paměti počítače Ural 2. Další etapou práce v Centru numerické matematiky KU bude vypracování programujícího programu umožňujícího užití pomocných pamětí Uralu 2 a zdokonalení soustavy symbolických operací i vlastního algoritmu programujícího programu.

Literatura

[1] *Jiří Raichl:* Programování a obsluha LGP 30, CNM 1961.

[2] *Ladislav Koubek:* Programující program samočinného počítače LGP 30, CNM 1961.

[3] Programování pro počítač Ural 2. Výpočetní laboratoř ministerstva dopravy a spojů 1961,

[4] *Koubek, Kudláček, Raichl:* Programování pro samočinný počítač LGP 30, SPN 1963.

Резюме

ПРОГРАММИРУЮЩАЯ ПРОГРАММА АСТ 1с ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЫ УРАЛ 2

ЛАДИСЛАВ КОУБЕК (Ladislav Koubek)

Программирующая программа АСТ 1с является попыткой автоматизировать программирование для быстродействующей вычислительной машины Урал 2 с применением малой вспомогательной машины LGP 30. Символическая программа сначала преобразована в машинную программу машины LGP 30, отлажена и затем на LGP 30 снова преобразована в машинную программу вычислительной машины Урал 2.

В работе описан язык, на котором записываются программы, и соответствующая машинная программа вычислительной машины Урал 2; затем описан полученный опыт в деле практического использования этого метода, который, конечно, можно применить и при сотрудничестве других вычислительных машин, а не только LGP 30 и Урал 2.

Summary

THE ACT 1c COMPILER FOR THE URAL 2 COMPUTER

Ladislav Koubek

The ACT 1c compiler is an attempt at automatisation of programming of the medium computer Ural 2 using the small computer LGP 30 as auxiliary. The symbolic programme is first transformed into a machine programme for the LGP 30 and tested out; and then transformed, again on the LGP 30, into a machine programme for the Ural 2.

This paper contains a description of the input symbolic language, the corresponding Ural 2 machine programme, and some remarks on experience with this compiling system; it may obviously be modified for the cooperation of a different pair of computers.

Adresa autora: Ladislav Koubek C.Sc., Centrum numerické matematiky Karlovy university, Malostranské nám. 25, Praha 1 - Malá Strana.