Vol. 21 No. 2 情報処理学会論文誌 Mar. 1980

高水準マイクロプログラム記述用言語による エミュレータの記述とその最適化効果†

重松保弘†† 阿南憲子††† 有川 薫††††

仮想のスタック・コンピュータ (SC) を用いる PASCAL (P) の処理系は，利用者が SC のシミュレータを作成することにより，容易に利用者の手元の計算機に移植できる有利なシステムであるが，記憶領域と実行時間の点で効率が悪い．そこで，筆者らは，SC のシミュレータをマイクロプログラム (μP) 化することにより SC のエミュレータを作成し，効率の改善を行った．

SC のエミュレータの記述にあたって，高水準 μP 記述用言語 MPL 200/II を用いた結果，μP のコーディングとデバッグに要する労力を軽減でき，解読性のよいソース μP リストが得られた．また，MPL 200/II コンパイラにおいて，冗長なマイクロ命令の削除などの最適化に加えて，主記憶のアクセス時間における改良した最適化技法を適用した結果，エミュレータのオブジェクト μP の語数を約 20% 縮小することができた．また，いくつかの PASCAL プログラムを実行した結果，中型計算機上のシミュレータと比較して，エミュレータ化により 3.4～3.6 倍の処理速度が得られた．また，最適化によって，エミュレータの処理速度を，約 1.3 倍に向上させることができた．さらに，コンパイラによって最適化された μP は，人手によって最適化された μP にかなり近いことが確かめられた．

1. まえがき

N. Wirth によって設計されたプログラミング言語 PASCAL は，アルゴリズムやデータ構造の記述性に富んでおり，プログラミング教育に適した言語として広く利用されている．PASCAL の処理系も，1971 年に CDC 6000 を対象としたものが作成され，他機種への移植も行われている．これらの処理系におけるコンパイラは，特定の計算機の機械語で記述されており，機械語の目的コードを生成する．これに対して，仮想のスタック・コンピュータ (SC) を設計し，コンパイラによって SC のためのコード (SC コード*) を生成する処理系が作られた．そのコンパイラ自身も SC コードに変換されており，すべての処理は SC のシミュレータによって行われる．この方式による処理系は PASCAL (P) と呼ばれている[2]．

PASCAL (P) の処理系は，利用者が SC のシミュレータを作成するだけで動作させることができ，比較的，容易に利用者の手元の計算機に移植することができる有利な方法である．しかし，この処理系はシミュレータを用いていることなどから，記憶領域と実行時間の点で効率が悪いことが指摘されている[4]．そこで，筆者らは，こうした不利な点を補う方法として，SC のシミュレータをマイクロプログラム (μP) 化することによって SC のエミュレータを開発し，処理系の効率について評価を行った．

また，WCS (Writable Control Storage) をもつ μP 制御方式計算機の普及に伴って，こうしたエミュレータを利用者自身が開発する機会が増えてきたが，その記述には，アセンブリ言語形式の言語が多く用いられている．筆者らは，SC のエミュレータの記述において，高水準 μP 記述用言語 MPL 200/II[14] を用いた結果，μP のコーディングとデバックに要する労力を軽減でき，解読性のよいソース μP を得ることができた．

μP の開発に高水準言語を用いるときの問題点は，効率の良いオブジェクトコードが得られるかどうかである．本稿では，MPL 200/II コンパイラにおける，主記憶のアクセス時間の有効利用に関する改良された最適化技法について述べた後，エミュレータの語数および実行時間に関する自動最適化の効果，および人手による最適化との比較結果について述べる．

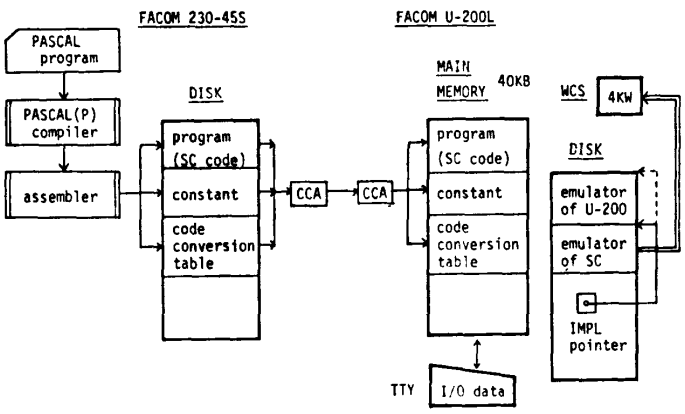
なお，実現された μP には，SC のシミュレータ以外の部分も含まれるので，実現された μP 全体を PASCAL マシンのエミュレータと呼ぶことにする．

---

† Description of an Emulator Using a High-level Microprogramming Language and its Optimization Effect by YASUHIRO SHIGEMATSU, NORIKO ANAMI, and KAORU ARIKAWA (Department of Computer Science, Kyushu Institute of Technology).

†† 九州工業大学情報工学科
††† 現在は大分電子計算センター
†††† 現在は日立製作所

* Pascal-Code とも呼ばれる[2]．



WCS: Writable Control Storage.
IMPL pointer: Initial MicroProgram Load pointer.
CCA: Channel to Channel Adapter.

図 1 PASCAL マシンによる PASCAL プログラムの処理過程
Fig. 1 The execution process of a PASCAL program using the PASCAL machine.

2. PASCAL マシンの処理システム

PASCAL マシンの処理システムは，**図 1** に示すように，FACOM 230-45 S (45 S) と U-200 L の複合計算機システムになっている．PASCAL プログラムは 45 S 上で，SC の機械語コード (SC コード) と定数に変換され，同時にコード変換表*が生成される．生成された SC コード等は，チャネル結合装置 (CCA) 経由で U-200 L に転送され，エミュレートされる．

U-200 L のディスクには，数種類の μP が格納されている．その主なものは，U-200 を目的計算機とするエミュレータと PASCAL マシンのエミュレータである．どの μP を WCS へ格納するかは，ディスク中の IMPL** ポインタの値によって決定される．IMPL ポインタの指す μP は，ROS*** (256 語) に格納された μP によって主記憶を経由して WCS に格納され制御が移される．

2.1 スタック・コンピュータの構成

SC を実現するための U-200 L の主記憶のレイアウトと，各領域を管理するためのポインタを**図 2** に示す．ポインタは，U-200 L の特定のレジスタに割り付けている．

図 3 に，SC 命令のフィード形式を示す．SC 命令は，基本的には，OP (Operation) 部と P, Q の両オペランド部から構成され，その語長は，通常，固定長としてインプリメントされる[2]．しかし，命令によっては，P または Q の一方，あるいは両方をもたない場合がある．そこで，筆者らは，命令に応じて語長を可変にした．大半の命令は 1〜2 語長である．CHK 命令のみ 3 語，LDC 命令のうち set を扱うもののみ 5 語とした．インプリメントした SC 命令のニモニックは，**図 6** の SSA 文のラベル部に示されている．

SC の機能については文献 2) および 5) に詳しいので，ここでは省略する．

2.2 エミュレータの構成

PASCAL マシンのエミュレータは，SC コード等の受信部と，SC のエミュレータ部に分かれる．前者および後者の μP を実行している状態を，それぞれ，受信状態およびエミュレート状態と呼ぶ．エミュレート状態の μP は，さらに，命令フェッチ，エミュレーションおよびディスプレイパネル操作の各フェイズに分かれる．各 μP の処理の概要を**図 4** に示す．

また，PASCAL マシンは，SC 命令の実行状態によって，Normal*, Step**, Display***, Store**** のモードをとる．

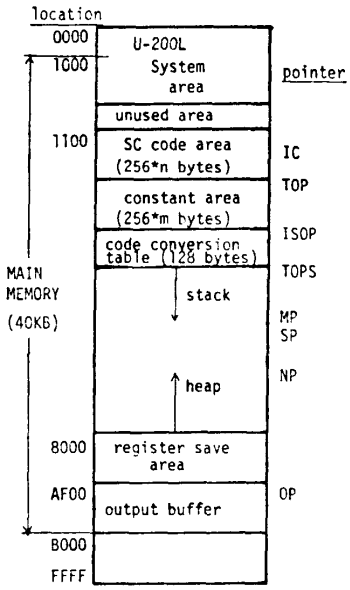


図 2 スタックコンピュータのための U-200 L の主記憶の構成
Fig. 2 Main memory layout of U-200 L for the Stack Computer.

* U-200 L のタイプライタの ISO コードと PASCAL の内部コードとのコード変換に用いる．
** Initial Microprogram Load.
*** Read Only Storage.

* SC 命令を連続実行する．
** START スイッチを押すことにより 1 命令ずつ実行する．
*** 指定されたレジスタ等の内容をディスプレイパネルに表示する．
**** エントリスイッチの値を指定されたレジスタ等に格納する．

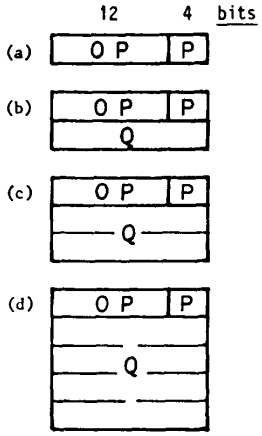


図 3 スタックコンピュータの命令形式
Fig. 3 Instruction format of the Stack Computer.

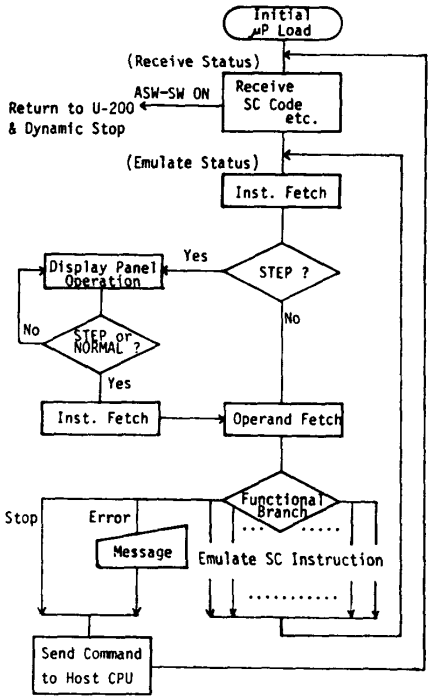


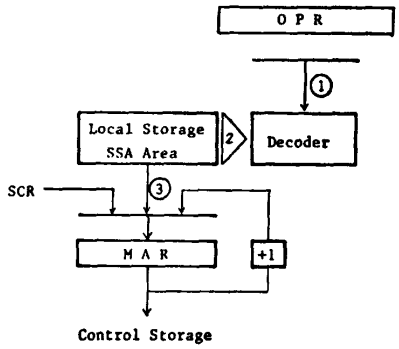
図 4 PASCAL マシンのステイタスとフェイズ
Fig. 4 The status and the phase of the PASCAL machine.

3. MPL 200/II によるエミュレータの記述

この章では，エミュレータが，MPL 200/II を用いてどのように記述されるかを示す．

まず，U-200 Lにおける機能分岐(functional branch)の機構について簡単に述べる．機能分岐が指定されると，**図 5** に示すように，OPR* の内容がデコードされ(①)，ついで，ローカルメモリのSSA** 領域 (64 語)

* Operation Register.
** Set Start Area.



OPR: Operation Register.
SCR: Sequence Control Register.
MAR: Microinstruction Address Register.

図 5 U-200 L における機能分岐の機構
Fig. 5 The mechanism of functional branch of U-200 L.

から 1 語が選択され(②)，MAR* に格納される(③)．

MPL 200/II では，この機能分岐の機構を利用するために SSA 文と SET 文を用意している．SSA 文は，目的計算機の機械コードと，そのエミュレーション・ルーチンの対応関係を保つためのマッピング・テーブルを生成し，SSA 領域に格納するために用いる．また，SET 文は，機能分岐を起こさせるために用いる．PASCAL マシンのエミュレータの SSA 文を**図 6** に示す．また，エミュレーション・ルーチンの一部を**図 7** に示す．図 6 において，各 SSA 文の右のコメントは，割り当てられた OP コードを示す．

機能分岐は，μP 制御方式特有の多分岐型分岐方式であり，いくつかの高水準 μP 記述用言語には，これを利用するための文が用意されている**．ただ，一般的には，多分岐型の文のオブジェクト μP には，(1)条件分岐型 μI を組合せた μP，(2)無条件分岐型 μI を多段に用いた間接分岐型の μP，(3)機能分岐を利用した μP，などが考えられるため，これらを考慮した文を用意することが望ましい．MPL 200/II では，(1)に対して CASE 文を，(3)に対して SSA 文と SET 文を，それぞれ用意してある．このため，たんなる多分岐か，機械コードによる機能分岐かの区別を明確化できた．

なお，付録に，命令フェッチ・フェイズとディスプレイパネル操作フェイズの μP リストを示す．

4. エミュレータの最適化

この章では，MPL 200/II コンパイラにおける最適

* Microinstruction Address Register.
** たとえば，Datasaab FCPU μP には DO CASE 文，MPGL[9] には CASE 文がある．

```
SSA   0   ERROR;
SSA   1   ERROR;
SSA   2   ERROR;
SSA   3   ERROR;
SSA   4   ERROR;
SSA   5   ERROR;
SSA   6   ERROR;
SSA   7   UNI ;     /* 0B0 */
SSA   8   STP ;     /* 1C0 */
SSA   9   STQ ;     /* 1C4 */
SSA  10   SQI ;     /* 1C8 */
SSA  11   SGS ;     /* 1CC */
SSA  12   SBI ;     /* 1D0 */
SSA  13   ODD ;     /* 1D4 */
SSA  14   NOT ;     /* 1D8 */
SSA  15   NGI ;     /* 1DC */
SSA  16   ERROR;
SSA  17   ERROR;
SSA  18   ERROR;
SSA  19   ERROR;
SSA  20   ERROR;
SSA  21   ERROR;
SSA  22   ERROR;
SSA  23   ERROR;
SSA  24   MPI ;     /* 1E0 */
SSA  25   MOD ;     /* 1E1 */
SSA  26   IOR ;     /* 1E2 */
SSA  27   INT ;     /* 1E3 */
SSA  28   INN ;     /* 1F0 */
SSA  29   EOF ;     /* 1F1 */
SSA  30   DVI ;     /* 1F2 */
SSA  31   DIF ;     /* 1F3 */
SSA  32   ANDP ;    /* 0C0 */
SSA  33   ADI ;     /* 0C1 */
SSA  34   ABI ;     /* 0C2 */
SSA  35   XJP ;     /* 0C3 */
SSA  36   UJP ;     /* 0C4 */
SSA  37   SRO ;     /* 0C5 */
SSA  38   MOV ;     /* 0C6 */
SSA  39   LDO ;     /* 0C7 */
SSA  40   LCA ;     /* 0C8 */
SSA  41   LAO ;     /* 0C9 */
SSA  42   IXA ;     /* 0CA */
SSA  43   IND ;     /* 0CB */
SSA  44   INC ;     /* 0CC */
SSA  45   FJP ;     /* 0CD */
SSA  46   ENTP ;    /* 0CE */
SSA  47   DECP ;    /* 0CF */
SSA  48   CSP ;     /* 0D0 */
SSA  49   CHK ;     /* 0D1 */
SSA  50   NEQ ;     /* 0D2 */
SSA  51   LES ;     /* 0D3 */
SSA  52   LEQ ;     /* 0D4 */
SSA  53   GRT ;     /* 0D5 */
SSA  54   GEQ ;     /* 0D6 */
SSA  55   EQUP ;    /* 0D7 */
SSA  56   LCI ;     /* 0D8 */
SSA  57   STRP ;    /* 0D9 */
SSA  58   LOD ;     /* 0DA */
SSA  59   LDC ;     /* 0DB */
SSA  60   LDA ;     /* 0DC */
SSA  61   CUP ;     /* 0DD */
SSA  62   RETP ;    /* 0DE */
SSA  63   MST ;     /* 0DF */
```

図 6 SC 命令のエミュレーションルーチンの先頭番地を，U-200 L のローカル・ストレジの SSA 領域に格納するための SSA 文 (MPL 200/II で記述)

Fig. 6 SSA statements (written in MPL 200/II) to store the start addresses of emulation routines for each SC instruction into the SSA (Set Start Address) area of the local storage of U-200 L.

化手法の改良点，および，各種最適化手法の実効性について述べる．

4.1 主記憶のアクセス時間の有効利用

一般に，μP 制御方式の計算機では，主記憶のアクセス時間は，制御記憶のそれと比較すると，はるかに長い．そのため，主記憶のアクセス中に，他の有効なマイクロ命令 (μI) を実行できれば，μP の実行時間の短縮が期待できる．

U-200 L では，主記憶のアクセスを行うために，次の手順を必要とする．すなわち，AREQ (Access

```
/* LOAD CONTENTS OF BASE-LEVEL ADDRESS */
LDO :  CALL @GET ;
       SP = SP + 8 ;
       AB = @ SHL 3 ;
       W1 = TOPS + AB ;
       W2 = SP ;
       CALL READWRITE ;
       GO TO NEXT0 ;

/* STORE AT ADDRESS */
STRP : CALL @GET ;
       CALL BASE@ ;
       W1 = SP ;
       W2 = AB ;
       CALL READWRITE ;
       SP = SP - 8 ;
       GO TO NEXT0 ;

/* STORE */
SRO :  CALL @GET ;
       AB = @ SHL 3 ;
       W1 = SP ;
       W2 = TOPS + AB ;
       CALL READWRITE ;
       SP = SP - 8 ;
       GO TO NEXT0 ;

/* LOAD ADDRESS */
LDA :  CALL @GET ;
       CALL BASE@ ;
       SP = SP + 8 ;
       W1=AB SHR 3;
       WRITE W1 INTO SP+2 WORD;
       GO TO NEXT0 ;
```

図 7 MPL 200/II で記述した SC 命令のエミュレーションルーチンの例

Fig. 7 Examples of emulation routines of SC instruction (written in MPL 200/II).

Request) μI によって主記憶のアクセスを開始し，WAIT μI によって，その完了の同期をとる．したがって，この両 μI の間に他の有効な μI を移動することにより μP の実行時間を短縮できる．ただし，次の場合には移動ができない．第 1 の場合は，リソースの競合を招く場合である．U-200 L のいくつかのリソース*は，主記憶のアクセス中に他の μI がこれを使用すると，リソースの競合を招く．したがって，これらのリソースを使用する μI は，AREQ と WAIT の両 μI の間に移動できない．これを，リソースの競合による移動禁止条件が成立する，と呼ぶ．第 2 の場合は，μI の移動によって，情報の流れ (Information flow) の構造が破壊される場合である．これについては，最適化の手順中で述べる．

次に，この最適化の手順を示す．ただし，i は節 (node) に付けられた番号を示す．M は，AREQ, WAIT の両 μI の節の間に移動された節の番号の集合を示す．また，節 i の μI が参照または定義するリソースの集合を，それぞれ $r_u(n_i)$, $r_d(n_i)$ と表わす．また，ある μI の節の列が制御の流れ (Control flow) においてシーケンシャルであり，かつ，入口と出口を各 1 つしかもたないとき，これを基本ブロック (basic block) と呼ぶ．

(手順)

* MB (Memory Buffer), SR (System status Register) 等.

(1) WAIT μI の節を含む基本ブロックにおいて，WAIT μI の節に続く節に，1から順次，番号付けを行う．このとき，付与された番号の最大値を m とする．

(2) $i \leftarrow 1$, $M \leftarrow \phi$.

(3) 節 i で，リソースの競合による移動禁止条件が成立すれば(5)へ行く．

(4) 次の条件を調べ，そのすべてを満足するとき，節 i を WAIT μI の節の直前に移動し，M に i を入れる．

条件(a) $r_u(n_i) \cap \left\{ \bigcup_{j=1,\ j \notin M}^{i-1} r_d(n_j) \right\} = \phi.$

条件(b) $r_d(n_i) \cap \left\{ \bigcup_{j=1,\ j \notin M}^{i-1} r_u(n_j) \right\} = \phi.$

条件(c) $r_d(n_i) \cap \left\{ \bigcup_{j=1,\ j \notin M}^{i-1} r_d(n_j) \right\} = \phi.$

(5) $i=m$ なら終る．

(6) $i \leftarrow i+1$, (3)へ行く．

WAIT μI の直接後行節から，節 i の直接先行節までのすべての節を中間節と呼ぶことにすると，上記の条件(a)は，節 i の μI が参照するリソースを，中間節の μI が定義していないことを示す．条件(b)は，節 i の μI の定義するリソースを，中間節の μI が参照していないことを示す．条件(c)は，節 i の μI の定義するリソースを，中間節の μI が再定義していないことを示す．

図8は，主記憶のアクセスを含む μP の例であり，**図9**，**図10**は，それぞれ，従来の最適化および改良後の最適化を適用したときのオブジェクト μP である．図10の μP は，図9の μP と比較すると，実行時間が 6T (70 ns/T) 減少している．

4.2 最適化効果

表1に，命令フェッチ・フェイズの μP の語数と実行時間における最適化効果を示す．なお，(a)は，最適化を行わないとき，(b)は，前節の最適化を行わないとき，(c)は，すべての最適化を行ったとき，(d)は，人手による最適化を行ったとき，の各 μP である．表1からわかるように，主記憶のアクセス時間の有効利用に関する最適化が，実行時間の短縮に有効に作用している．

人手による最適化は，(c)の μP に対して，次の4種類を適用した．第1は，前節で述べた最適化を基本ブロック外に拡張して適用する．第2は，リソースの競合を避けるために無条件で挿入した無効 μI を，他の有効な μI で置きかえる．第3は，文献 13) の最適化手法を拡張して適用する*．第4は，U-200L 特有

```
539
                /* BOOLEAN AND */
                ANDP : AB=SP+2;
                       SP=SP-8;
540                    READ #1 WORD ;
541                    READ FROM SP+2 WORD;
542                    WRITE MB AND #1 WORD;
543                    GO TO NEXTO ;
544
```

図8 主記憶のアクセスを含む μP の例 (MPL 200/II で記述)

Fig. 8 An example of μP (written in MPL 200/II) which includes main memory access.

```
1237  ANDP      MV          GR3,AB
1238            A           C0002,AB
1239            LP          100110,ER
1240            S           ER,GR3
1241            (AREQ,INST)
1242            (WAIT)
1243            MV          MB,FW1
1244            MV          GR3,AB
1245            A           C0002,AB
1246            NOP
1247            (AREQ,INST)
1248            (WAIT)
1249            AND         FW1,MB
1250            NOP
1251            (AREQ,W,INST)
1252            B           NEXTO
1253            (WAIT)
```

図9 図8のオブジェクト μP (従来の最適化技法を適用)

Fig. 9 Object μP of Fig. 8 (old optimization technique is applied).

```
1234  ANDP      MV          GR3,AB
1235            A           C0002,AB
1236            LP          100110,ER
1237            S           ER,GR3
1238            (AREQ,INST)
1239            MV          GR3,AB
1240            A           C0002,AB
1241            (WAIT)
1242            MV          MB,FW1
1243            (AREQ,INST)
1244            (WAIT)
1245            AND         FW1,MB
1246            NOP
1247            (AREQ,W,INST)
1248            B           NEXTO
1249            (WAIT)
```

図10 図8のオブジェクト μP (新しい最適化技法を適用)

Fig. 10 Object μP of Fig. 8 (new optimization technique is applied).

表1 PASCAL マシンの命令フェッチフェイズの μP に対する最適化効果

Table 1 The effect of the optimization techniques for the μP of the instruction fetch phase of the PASCAL machine.

size (words)				execution time (T)			
(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
17	16	16	16	52	50	42	42

(a): not optimized.
(b): optimized.
(optimization for main-memory access is not executed)
(c): optimized.
(d): optimized by hand.
70 ns/T

* 無効 μI を置き換える有効な μI の選択範囲を拡大する．

表 2 PASCAL マシンのエミュレーションフェイズの μP に対する最適化効果
Table 2 The effect of the optimization techniques for the μP of the emulation phase of the PASCAL machine.

(1) internal procedures.

procedure name	size (words)				execution time (T)			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
QGET	7	7	7	7	32	32	32	28
BASE	17	12	12	13	20+40P	14+34P	14+34P	12+34P
BASEQ	21	16	16	16	32+40P	24+34P	24+34P	22+34P
CMP	19	18	17	17	72	68	56	50
READWRIT	15	12	11	12	212	194	146	144

(2) emulators of SC instructions.

instruction	size (words)				execution time (T)				internal procedure (used)
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)	
LOD	13	12	12	12	28	24	24	24	QGET, BASEQ READWRIT
LDO	13	12	12	12	34	28	28	28	QGET, READWRIT
STR	12	10	10	10	26	22	22	22	QGET, BASEQ READWRIT
SRO	14	13	13	12	32	26	26	26	QGET, READWRIT
LDA	16	15	16	16	52	48	48	48*1	QGET, BASEQ
STO	16	15	16	16	52	48	42	42	READWRIT
LDC (I)	84	66	66	63	52	40	40	40*1	QGET
LDC (B)					66	54	54	54*1	QGET
LDC (A)					76	64	64	64*1	QGET
LDC (S)					186	152	152	152*2	QGET*4
LCA	13	12	12	12	46	40	40	40*1	QGET
MST	20	19	18	18	76	74	54	54*1	BASE
CUP	19	18	18	17	66	62	58	58	QGET
ENT	8	7	7	7	20	16	16	16	QGET
RET (F)	26	22	21	21	78	70	60	56	
RET (P)					72	70	60	56	
FJP	20	15	15	15	44 46	40	40	40	QGET
UJP	5	4	4	4	10	6	6	6	QGET

*1……次に実行される SC 命令の命令フェッチフェイズの実行時間が 8 T 短縮される.
*2……*1 に加えて, QGET の実行時間が 8 T 短縮される.

のタイミングを利用した最適化*を行う. 以上である.

表2は, エミュレーション・フェイズにおけるいくつかの μP の語数と実行時間の最適化効果を示している. ここで, (1) および (2) は, それぞれ, 内部手続きおよび SC 命令のエミュレータ部に関する語数と実行時間を示している. (1)において, 内部手続き BASE と BASEQ の実行時間は, SC コードのオペランドP部の値によって決定される.

表2の (2) において, LOD～LCA の各命令は, PASCAL コンパイラのオブジェクトとして比較的多く生成されるデータ転送用命令である. この場合の最適化効果は, 主としてタイミングの同期用に挿入した無効 μI の削除によるものである. また, (d)では, 前節の最適化を基本ブロック外に拡張した効果が見られる. MST～RET の各命令は, 関数または手続きの呼出しに関するものであるが, この場合, (b), (c)ともに, 有効に最適化されている. FJP と UJP は分岐命令である. 各命令の実行時間は, これに, 内部手続きの実行時間を加えたものとなる.

* (1)AREQ μI のすぐ次のステップに限り, MB の値は, AREQ μI 以前の値が保証されることを利用した最適化. (2)サブルーチン・リターンのための μP のステップ数の短縮.

```
VAR I : INTEGER ;
PROCEDURE HANOI(N:INTEGER; X,Y,Z:CHAR);
    BEGIN
      IF N>0 THEN
        BEGIN
          HANOI(N-1,X,Z,Y);
          WRITE('MOVE PIECE ',N:1,' FROM ',X,' TO ',Z,';',EOL);
          HANOI(N-1,Y,X,Z);
        END
    END;
BEGIN
  FOR I:=1 TO 30000 DO
    HANOI(4,'A','B','C') ;
END.
```

(a) Example 1. Tower of HANOI.

```
VAR  I,J: INTEGER ;
     FUNCTION FIB(N: INTEGER): INTEGER ;
          BEGIN IF N=0
                THEN FIB:=0
                ELSE IF N=1
                     THEN FIB:=1
                      ELSE FIB:=FIB(N-1)+FIB(N-2)
          END ;
BEGIN
     FOR I :=1 TO 23 DO
          J := FIB(I)
END.
```

(b) Example 2. Fibonacci numbers.

```
 VAR  I,J,K : INTEGER ;
    FUNCTION ACK(M,N: INTEGER): INTEGER ;
          BEGIN  IF M=0
                 THEN ACK:=N+1
                 ELSE IF N=0
                      THEN  ACK:=ACK(M-1,1)
                      ELSE  ACK:=ACK(M-1,ACK(M,N-1))
          END ;
BEGIN
     FOR I:=0 TO 3   DO
          FOR J:=0 TO 5 DO
             K:=ACK(I,J)
END.
```

(c) Example 3. Ackerman's function.

図 11 PASCAL で書いた例題プログラム
Fig. 11 Example programs written in PASCAL.

表 3 PASCAL マシンのエミュレータの語数
Table 3 The size (words) of the emulator of the PASCAL machine.

emulator	(a)	(b) (100(a-b)/a)	(c) (100(a-c)/a)	(d) (100(a-d)/a)
size (words)	2,573	2,094 (18.6%)	2,061 (19.9%)	2,045 (20.5%)

表 4 F 230-45 S と U-200 L のメモリサイクル時間の比較
Table 4 The comparison of memory cycle time between F 230-45 S and U-200 L.

F 230-45 S	Main Memory	700 ns
U-200 L	Main Memory	650 ns
	Main Memory (for microprogram)	980 ns (Read) 1,120 ns (Write)
	Control Storage	140 ns

エミュレータ全体の語数の比較結果を**表 3** に示す．最適化によって，約 20% の μI が削除された．一般に制御記憶は，主記憶に比べて容量が極めて小さいので最適化によるエミュレータの語数の縮小は有用である．

次に，SC のシミュレータ*のエミュレータ化効果および，エミュレータの最適化効果について述べる．

図 11 は，実行時間を比較するために用いた例題プログラムである．**表 4** に，45 S と U-200 L のメモリサイクル時間の比較を示す．**表 5** は，実測結果である．45 S は，ハードワイヤード方式の中型計算機であり，SC のシミュレータでも，ある程度の速度が得られる．しかし，エミュレータ化によって 2.6～2.8 倍の処理速度が得られ，最適化により 3.4～3.6 倍に処理速度を向上させることができた．

また，エミュレータ(a)を最適化して得られたエミュレータ(b)，(c)では，(a)に比べて，それぞれ，1.1，1.3 倍の処理速度が得られた．表 3 と表 5 を比較すると，タイミングの同期用無効命令の削除などは，主として μP の語数の縮小に極めて有効であり，主記憶のアクセス時間の有効利用は，主として μP の実行時間の短縮に極めて有効に作用していることがわかる．こうした，各種の最適化技法の適用により，語数と実行時間のうえで，ともに，人手による最適化に近い，極めて有効な最適化効果を得ることができた．

5. あ と が き

SC のシミュレータを μP 化することにより，ミニコンピュータ上のエミュレータでも中型計算機上のシミュレータを充分上回る処理効率が得られることがわ

表 5 シミュレータ，エミュレータおよび最適化されたエミュレータの実行時間の比較
Table 5 The comparison of execution time between simulator, emulator and optimized emulators.

	example 1	example 2	example 3
simulator (s)	523 sec	146 sec	34 sec
emulator (a) (s/a)	195 (2.7)	53 (2.8)	13 (2.6)
emulator (b) (s/b), (a/b)	179 (2.9), (1.1)	48 (3.0), (1.1)	12 (2.8), (1.1)
emulator (c) (s/c), (a/c)	154 (3.4), (1.3)	41 (3.6), (1.3)	10 (3.4), (1.3)
emulator (d) (s/d), (a/d)	148 (3.5), (1.3)	40 (3.7), (1.3)	9 (3.8), (1.4)

* F 230-45 S 上にインプリメントされたもの[4].

かった．また，MPL 200/II を用いた結果，短期間に解読性の良い μP を作成することができ，高水準 μP 記述用言語の有用性を確認することができた．また，μP の最適化についても，対象機種を固定し，きめ細かく最適化を行うことにより，人手による最適化に近い効率の良い μP を得ることができた．

なお，最適化のうち，主記憶のアクセス時間の有効利用が効果的であったのは，SC が，主記憶上に設定されたスタックの操作を基礎としていることによると思われる．また，最適化に要した CPU 時間は，合計約500秒であり，その大半は Dead アルゴリズム[13]の処理に費された．

謝辞 PASCAL (P) コンパイラとシミュレータは，東京大学で開発されたものを使用させていただいたので，ここに謝意を表します．また，本研究の機会を与えていただいた本学・情報工学科の安在弘幸助教授，および，日頃，御指導を載く九州大学の吉田将教授に感謝します．

参 考 文 献

1) Jensen, K. and Wirth, N.: PASCAL User Manual and Report 2nd edition, p. 167, Springer-Verlag, New York (1975).
2) Nori, K. V., et al.: The Pascal (P) compiler Implementation Notes, Berichte der Fachgruppe Computer-Wisseuschaften, Nr. 10, ETH Zürich (1974).
3) 武市：Pascal コンパイラの portability について，第 16 回情報処理学会プログラミングシンポジウム報告集，pp. 90-96 (1975).
4) 武市，疋田，安村：東京大学の PASCAL コンパイラについて，情報処理学会第 16 回全国大会予稿集，pp. 123-124 (1975).
5) 疋田：コンパイラのキットを用いた PASCAL の移植，日経エレクトロニクス，pp. 100-131 (1976. 12. 13).
6) 馬場，萩原，藤本：マイクロプログラム記述言語 MPGL，情報処理，Vol. 18, No. 6, pp. 558-565 (1977).
7) 萩原：マイクロプログラミング，p. 343，産業図書，東京 (1977).
8) 島田，山口，坂村：LISP マシンとその評価，信学論(D)，Vol. 59-D, No. 6, pp. 406-413 (1976).
9) Griss, M. L. and Swanson, M. R.: MBALM/1700: A Microprogrammed LISP Machine for the Burroughs B 1726, MICRO 10 proc., pp. 15-25 (1977).
10) 宮脇，木下，渡辺，萩原：APL インタプリンタのファームウェア化とその効果について，情報処理学会論文誌，Vol. 20, No. 2, pp. 172-178 (1979).
11) 馬場，藤本，萩原：MPG マイクロプログラムコンパイラ，情報処理，Vol. 19, No. 1, pp. 16-25 (1978).
12) 松崎：実用化の試みが始まったマイクロプログラムの最適化，日経エレクトロニクス，第 185 号，pp. 76-91 (1978).
13) 重松，有川，安在：マイクロプログラミング言語 MPL 200 とその最適化技法，情報処理，Vol. 19, No. 8, pp. 749-757 (1978).
14) 重松，飯野，安在：マイクロプログラミング言語 MPL 200/II，九州工業大学研究報告（工学），No. 37, pp. 115-122 (1978).
15) 重松，阿南：MPL 200/II による PASCAL マシンの記述について，情報処理学会・計算機アーキテクチャ研究会資料 32-2 (1978).
16) 重松，飯野，安在：MPL 200/II Recursive Descent Compiler とその簡単なコンパイラ記述用語，情報処理学会論文誌，Vol. 20, No. 4, pp. 346-354 (1979).
17) 有川，重松，安在：マイクロプログラムの最適化について，電気四学会九州支部大会論文集，pp. 334 (1978).
18) 馬場：ファームウェア工学，情報処理，Vol. 20, No. 7, pp. 622-646 (1979).
19) 十山，柴山，富田，萩原：QA-1 による Pascal 用スタック計算機のエミュレーション，情報処理学会第 20 回全国大会論文集，pp. 25-26 (1979).

付　録

```
115
          /* INSTRUCTION FETCH PHASE */
          NEXT0 : IC = IC + 2 ;
116       NEXT : READ OPR FROM IC WORD ;
117       w1=KEY1;
118       w1=w1 AND X'00F0';
119       IF w1=STEP THEN
           BEGIN   /* DISPLAY PANEL OPERATION PHASE */
            LOOP
              SR=0;
120           REPEAT
                w2=SR AND START;
121           UNTIL w2=START;
122           w1=KEY1;
123           w1=w1 AND X'00F0';
124           w2=0;
125           IF w1=STEP   THEN w2=1; FI
127           IF w1=NORMAL THEN w2=1; FI
129         EXIT IF w2=1;    /* STEP OR NORMAL MODE */
130           w2=KEY1;
131           w2=w2 AND X'000F';
132           CASE w1 OF
                /* DISPLAY MODE */
                X'0050': IF w2<8 THEN
                           BEGIN
                             OPR=w2;
133                          w2=GRS;
134                          OPR=w2;
135                        END ELSE
                           CASE w2 OF
                                 8: OPR=ADR;
136                              9: READ OPR WORD;
137                             10: OPR=STR;
138                             11: OPR=IRC;
139                             12: OPR=FR0;
140                             13: OPR=FR1;
141                             14: OPR=FR2;
142                             15: OPR=FR3;
143                        ESAC
144                     FI
145             /* STORE MODE */
                X'0060': IF w2<8 THEN
                           BEGIN
                             OPR=w2;
146                          GRS=ENT;
147                        END ELSE
                           CASE w2 OF
                                 8: ADR=ENT;
148                              9: WRITE ENT WORD;
149                             10: STR=ENT;
150                             11: IRC=ENT;
151                             12: FR0=ENT;
152                             13: FR1=ENT;
153                             14: FR2=ENT;
154                             15: FR3=ENT;
155                        ESAC
156                     FI
157             ELSE: ;
158           ESAC
159         POOL
160         READ OPR FROM IC WORD;
161       END
          FI
162           CT = OPR ;
163           P = CT ;        /* GET  OPERAND P */
164           SET SA EX ;     /* FUNCTIONAL BRANCH */
```

MPL 200/II で記述した PASCAL マシンの命令フェッチ・フェイズとディスプレイパネル・オペレーション・フェイズの μP

The μP (written in MPL 200/II) for the instruction fetch phase and the display panel operation phase of the PASCAL machine.

(昭和 54 年 4 月 23 日受付)
(昭和 54 年 12 月 20 日採録)