# 高速のアドレス割当て機能を持つマイクロプログラミング・システム†

宮地利雄†† 榎本肇††

一般に，マイクロプログラムの開発では，ハードウェアの特徴や制約条件に強く拘束される部分が多い．ファームウェア利用の一般化に伴ない，これを自動的に処理し，利用者の負担を軽減するためのシステムの研究が広く進められている．

本論文は，ハードウェア制約条件のうちマイクロ命令の番地割当てに関する問題に焦点を置き，高速の番地割当て機能を持つシステムを実現し，その方式の検討を行ったものである．本システムでは，比較的単純な番地割当てアルゴリズムを使用しながら，十分な前処理を施すことにより，実用上十分な処理速度と出力コード密度で，機能分岐を含む自動番地割当てを実現した．本処理の高速動作を可能にするため，前処理では，番地割当てに関する拘束条件を単純なデータ構造により表現するとともに，その構造内部の順序を一定の評価値により整列する．なお，このシステム自身は，特定の機械のみを対象としているが，本論文では，我々のアルゴリズムを他の機械へ適用する場合に必要な拡張についても検討する．

## 1. まえがき

ファームウェア開発には，ハードウェアの特徴や制約条件に関する詳細な知識が必要であり，一般にソフトウェア開発に比較して困難で経験が要求される．そこで，マイクロプログラム[1]利用の一般化に伴ない，機械に強く依存した部分はシステムで処理し，利用者の負担を軽減するようなマイクロプログラミング・システムのための研究が広く進められている[2]．

筆者らは，ハードウェア制約条件のうちマイクロ命令（以降，単に命令と呼ぶ）の番地割当てに関する問題に焦点を置き，本システムにおいて，その自動化を実現し[3]方式の検討を行った．一般に，制御記憶はソフトウェアのアドレス空間に比べて小さく，命令の種類によっては分岐可能な番地の条件も複雑なため，効率のよい番地割当て技法は重要であるが，これについては，今までほとんど報告されていない．通常，アセンブラ・システムでは，手作業で番地割当てを行っており，そうでない場合やコンパイラ・システムでも，機能分岐命令については，プログラマに頼っていたり，その使用を避けている場合が多い．萩原らのMPGコンパイラは[5]番地自動割当て手続きを含んではいるが，命令数が実用レベルになった時の処理速度やコードの効率については明らかにされていない．ところが，複雑な配置可能アドレスの制約条件のもとで，手作業で番地割当てを行うことは，プログラマにとって，極めて面倒な作業であり，特に開発時における修正や機能変更，機能拡張に際して，自動的かつ効率的な番地割当て機能を持つ，我々のシステムは，非常に有効である．さらに，コンパイラ・システムの実現にあたっては，この技術は欠くことができない．

本システムは，アセンブラ・システムであり，特定のハードウェア（ACOS シリーズ 77 システム 300/400）のみを対象としてはいるが，実用上十分な処理速度と出力コード密度で，機能分岐を含む命令の自動番地割当てを行う．我々のシステムでは，比較的単純な番地割当てアルゴリズムを使用しながら，十分な前処理を行うことにより，その性能を得ていることに特色がある．

本論文では，2で一般のマイクロプログラムについて順序制御方式と番地制約条件の主要なものを掲げて関係を述べ，3でシステムの概要を，4で番地割当てアルゴリズムを述べる．5では，アルゴリズムにおける前処理の重要性を指摘して解析し，さらに，これを他の機種に適用する場合に必要な拡張についても言及する．

## 2. マイクロプログラムの順序制御と番地割当て

マイクロプログラムの実行順序の指定は，制御記憶番地レジスタ（以降，CMAR と略記する）上に，次に実行する命令の番地を作り出すことにより行われる

† A Microprogramming System Equipped with a High-speed Code-allocater by TOSHIO MIYACHI and HAJIME ENOMOTO (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

が，一般に利用されている番地生成方式は，ほぼ次のいくつかの組合せである．

（i） 一部のフィールドについて，直前の CMAR の内容をそのまま残す．

（ii） 命令の番地フィールドの値をセットする．

（iii） CMAR の一部（この場合，そのフィールドからの桁上りは無視）あるいは全体を，決められた値（多くの場合は 1）だけ増加する．

（iv） レジスタやカウンタ等の値(機能分岐)，またはテスト条件の成立に応じて1または0の値（条件分岐）を，決められたフィールドにセットする．

（v） 結線論理により決められる値をセットする．

無条件分岐命令の場合には，(i)と(ii)または(iii)との組合せが，条件付分岐命令の場合には，(i)と(ii)と(iii)（(iii)で増加する値を変える，いわゆる，スキップ，またはテスト結果によって(ii)か(iii)のいずれかを採る）または(iv)との組合せが使用される．(v)は割込み等のやや特殊な場合に利用される．

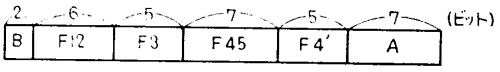
① (i)により命令の番地フィールドを削減しただけ分岐可能な範囲が狭められ，その結果，制御記憶空間が部分空間（以降，ページと呼ぶ）に細分される．

② (iii)では，命令とその次に実行される命令との制御記憶上での相対位置が決められ，たとえば，増加値が1の時には，実行順序に従って命令を配置しなければならない．

③ (iv)の場合には，分岐先の命令群において，相互間の相対位置が固定されるとともに，配置可能な絶対位置も制限される．

④ (v)またはその他の理由から，命令を配置すべき番地が指定されることがある．

本システムが対象とした ACOS 300 のマイクロ命令は，**図1**の形式を持ち，並列に実行される5種類のマイクロ操作を含んでいる．分岐用マイクロ操作には，12種類があり，番地割当てへの制限により分類すると**表1**のとおりで，上述のすべての制限の発生の可能性を持つ．なお，本システムでは，機能分岐で分岐




B：バスへデータを送出すユニットを示す
F12:主記憶アクセス・マイクロ操作とスクラッチパッド・マイクロ操作
F3:ALU マイクロ操作
F45:内部状態レジスタ・マイクロ操作と分岐用マイクロ操作
F4': F45の補助コード
A：短アドレスまたはパラメタ
F4'とA:長アドレスまたはパラメタ

**図 1** ACOS 300 のマイクロ命令形式
Fig. 1 Micro-instruction format of ACOS 300.

**表 1** ACOS 300 の分岐用マイクロ操作
Table 1 Branch micro-operation set of ACOS 300.

| 分岐タイプ＼分岐先 | | 同一ページ内 | | セグメント内任意 | もどり番地レジスタによる |
|---|---|---|---|---|---|
| | | 直後 | 任意 | | |
| 無条件分岐 | | INC | BUP | BUN | RAS<br>RIM |
| 条件付分岐 | | (BCD)<br>(RCD) | | BCD | RCD |
| 機能／条件分岐 | 2ウェイ | (BIX) | BSG<br>(4) | BIX<br>(64) | |
| | 16ウェイ | | BRM*<br>(8)<br>BID*<br>(8) | BIN<br>(32) | (BID) |

- 1ページは128語からなり，その内部では短アドレス命令で分岐できる．
- セグメントは，ベースレジスタを変更することなく分岐できる範囲で，最大32ページ，4,096語の大きさ．
- カッコで囲んだマイクロ操作は，テスト条件が不成立の時の分岐先．
- 機能／条件分岐における，カッコ内の数字は分岐先の間隔．
- BRM，BID ではテストされる4ビットのうち任意のビットをマスクできる．

先が複数ページにわたる BIN 命令は特殊な処理によっており，本論文で取扱わない．これに対する標準的手法は，**5.4**(b) で述べる拡張を採用すればよい．

## 3. システム構成の概要

本システムは，**図2**のように，アセンブラとリンケージ・エディタの二者を中心に構成される．前者は，手続きを単位としてソースプログラムを入力して番地制約条件を解析し，中間コードを生成する．後者は，

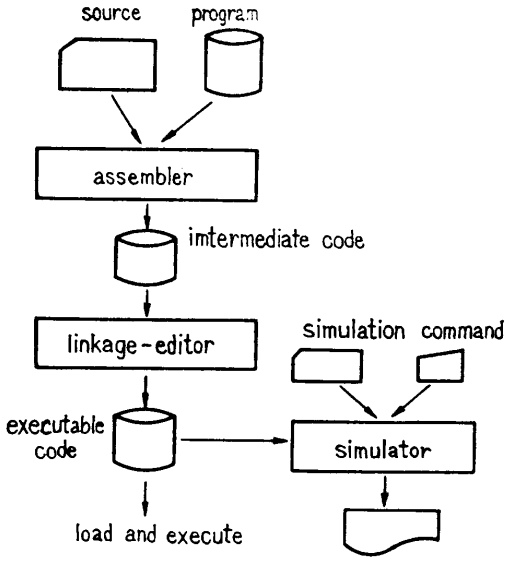



**図 2** システム構成
Fig. 2 Configuration of the micro-programming system.

必要な1個以上の手続きの中間コードを入力し，手続き間の外部参照を解決した後，各命令の番地を決定し，1セグメントの実行可能なコードを出力する．番地決定は，前節で述べた複雑な番地制約条件を考慮して，次節のアルゴリズムにより自動的に行われ，プログラマの負担を軽減するとともに制御記憶上での空間効率を向上させている．

ソースプログラムは，**図3**に例示した形式で記述され，アセンブラにより中間コードに変換される．中間コードは，基本的にはコード部と番地制約情報部とからなり，後者による制約のほかは，命令間の相対位置も含め完全に再配置可能である．コード部は，各命令ごとに作られる4つ組 (ICODE, DEST, AMASK, CMASK) の配列である．ICODE は番地フィールドを除いた命令のコーディングで，DESTは番地フィールドに対応する（通常は分岐先）命令の識別番号である．AMASK はその命令の番地フィールド位置，CMASK は ICODE の don't care でない有効部分を，それぞれ示すマスクである．最終的なコード生成は，次式により行われる．

CODE ← ICODE

**or** (〈DESTの番地〉 **and** AMASK).

この時，次式の関係の成立が必要である．

(CODE **xor** ICODE) **and** CMASK=0.

番地制約情報部は，前節の①～④の分類により整理された4種類の表からなっている．

```
$PROC( BRANCH , ID=BR );                    $FIX( OP(0)=0 );
$GLOBAL( OP(0-15) , OP_B(0-15) , IDX_ADDR , IND_ADDR , ADDR_ILLEGAL );
OP(11):  EIC,       LSA,        BRM 1F      / OP_B(0-15) /
OP_B(0):      RESB, LSA, SRA,   BIX         / INDEX:IDX_ADDR, INDIRECT:IND_ADDR /
              ARI,              BSG BRVA    / ELSE:EXIT /
         IRP,       USA,        BSG AC31    / THEN:ADDR_ILLEGAL /
                                BSG AC30    / ELSE:EXIT /
         EPR4,                  JMP
EXIT:    EJPL,WID3,LOP2,ROP0,BIN            / OP(0-15) /
$END;
```

- $PROC と $END は手続きの開始と終了，$FIX は特定の命令の番地の指定，$GLOBAL は手続き間で広域的なラベルの宣言のためのアセンブラに対する指令である．
- 第3～9行の各行は命令に対応し次の形式である．
  [(ラベル):][(マイクロ操作),]*(分岐マイクロ操作)[/(分岐先並らび/]
  分岐用マイクロ操作の指定が JMP の場合には，アセンブラが適当な無条件分岐操作を選択する．必要な分岐先並らびの省略は，次の行の命令の参照と解釈される．

**図3** マイクロアセンブラ言語の手続きの例
Fig. 3 Sample procedure in micro-assembler language.

## 4. 番地割当てアルゴリズム

各命令への番地割当ては，リンケージ・エディタで行われ，本システムの処理の中心をなすものであるが，番地制約条件と制御記憶の大きさの制限のために決められた大きさの箱の中へ種々の形状の積木を入れていく「ペントミノ・パズル」に似た組合せ的な問題となっており，特殊な制限された問題の場合を除いて解に到達するまで体系的に試行錯誤を繰返すようなアルゴリズムによらざるを得ない．

我々の番地割当てアルゴリズムは，前処理と本処理からなる．前処理では，中間コードから作業用テーブルを構成し，エラーの検査を行う．本処理は，作業用テーブル上で動作し，試行の繰返しにより条件を満足する命令配置を捜すもので，単純なアルゴリズムによっているが，解を得るまでの平均的な実行時間は，次節で述べるように前処理を十分に行うことにより著しく短縮されている．

### 4.1 作業用テーブルの構造

作業用テーブルは，ロケーション表，ページ表，命令リストからなる．ロケーション表には，各ロケーションに対する命令の割当て状態が記録され，ページ表には，各ページの空き領域の語数や特定のページに割当てる必要がある命令のリスト（ページ集合）等が，記録される．命令リストは，命令エントリを要素とするリスト構造で，論理的構造は**図4**のとおりである．

ページ集合は，割当てページが同一である必要のある命令の集まりで，グループは，相互間の相対位置が与えられた命令群である．

命令エントリは，4つ組 (ID, ICODE, ADDR, MASK, DISP) で，ID は命令の識別情報，ADDR は番地割当て作業中に記録される一時的な番地で，ページ番号 P-N とページ内番地 P-ADDR の組で表わされる．ICODE は，中間コードのそれと同じである．MASK と DISP は，グループ内でのみ有効で，同一グループに属する命令間の相対的位置関係を示すもので，前処理中に計算されセットされる．DISP は，同一グループ内の命令間の相対番地を規定し，さらに MASK の0でないビット位置に対しては，その命令が割当てられてよい番地のビットをも規定している．**図5**に，DISP と MASK がセットされる一例を示す．

同一のページ集合に属す命令は，同一のページ番号を持つように，同一のグループに属する命令は，共通

```
〈命令リスト〉: sequence of 〈ページ集合〉;
〈ページ集合〉: record
                 g-seq: sequence of 〈グループ〉;
                 i-set: set of 〈命令エントリ〉;
                 end
  〈グループ〉: sequence of 〈命令エントリ〉;
```

**図4** 命令リストの論理的構造
Fig. 4 Logical structure of a micro-instruction list.

```
A: …
B: …
C: …, INC /A/
   …, BSG … /A,B/
```

（a） ソースプログラム

| 命令ラベル | MASK（2進数） | DISP（2進数） |
|---|---|---|
| A | 00100 | 00001 |
| B | 00000 | 00101 |
| C | 00000 | 00000 |

（b） （a）に対する MASK, DISP の設定

- A, B, C は同一グループに属す
- AはCの直後の番地へ……条件②
- Aの番地は下から4ビットめが0で，その4語後がBの番地……条件③

図 5 MASK, DISP の設定例
Fig. 5 Sample value of MASK and DISP.

の gbase (0≦gbase＜ページ長) に対して次式を満足するように割りつけられる.

$$\begin{cases} \text{P-ADDR}=\text{DISP}+\text{gbase } \textbf{mod } \text{ページ長}, \\ (\text{P-ADDR } \textbf{xor } \text{DISP}) \textbf{ and } \text{CMAR}=0. \end{cases}$$

### 4.2 前 処 理

前処理では，**図6**のアルゴリズムに従って，入力した中間コードから本処理のための作業用テーブルを構成し，命令リストをソートする．作業用テーブルは，各命令がグループに属さず，それぞれ異なるページ集合を形成するように初期化され，続いて，ページ内分岐命令により一方から他方へ分岐する命令の対すべてについて，それぞれ対になっている2命令が同一ページ集合に属するよう，各々が属するページ集合が1つのページ集合にマージされ，また，CMAR 加算型分岐命令により一方から他方へ分岐する命令対と，機能／条件分岐命令からの分岐先の命令群のすべてについて，それぞれグループが構成され，MASK, DISP が調整される．なお，番地が指定された命令およびそれと同じグループに属す命令に対しては（後者については，割当て番地を算出し），必要なチェックの後，これらを含むページ集合を命令リストから除いてページ表にリンクし，これらの命令をロケーションの表の各々の番地に記入し，ページ集合から削除する．

最後では，命令リスト内でのページ集合，ページ集合の $q$-seq 内でのグループ，およびグループ内での命令エントリを，番地割当ての困難な順序に並らべかえる．このソートは，本処理の処理時間短縮のうえで，次節に述べるような大きな効果を持っている．なお，番地割当て難易度の評価は，次の考え方による．

① 命令数の多いグループやページは割当て困難.
② 間隔が大きく分岐数の多い機能／条件分岐の分岐先になっている命令，または，これを含むグループやページは割当て困難.

本システムでの割当て難易度の評価値は，命令エントリに対しては，その MASK を2進数として見た値を，グループでは，命令エントリの難易度の和に命令エントリ数の2倍を加えた値を，ページでは，グループの難易度の和を用いた.

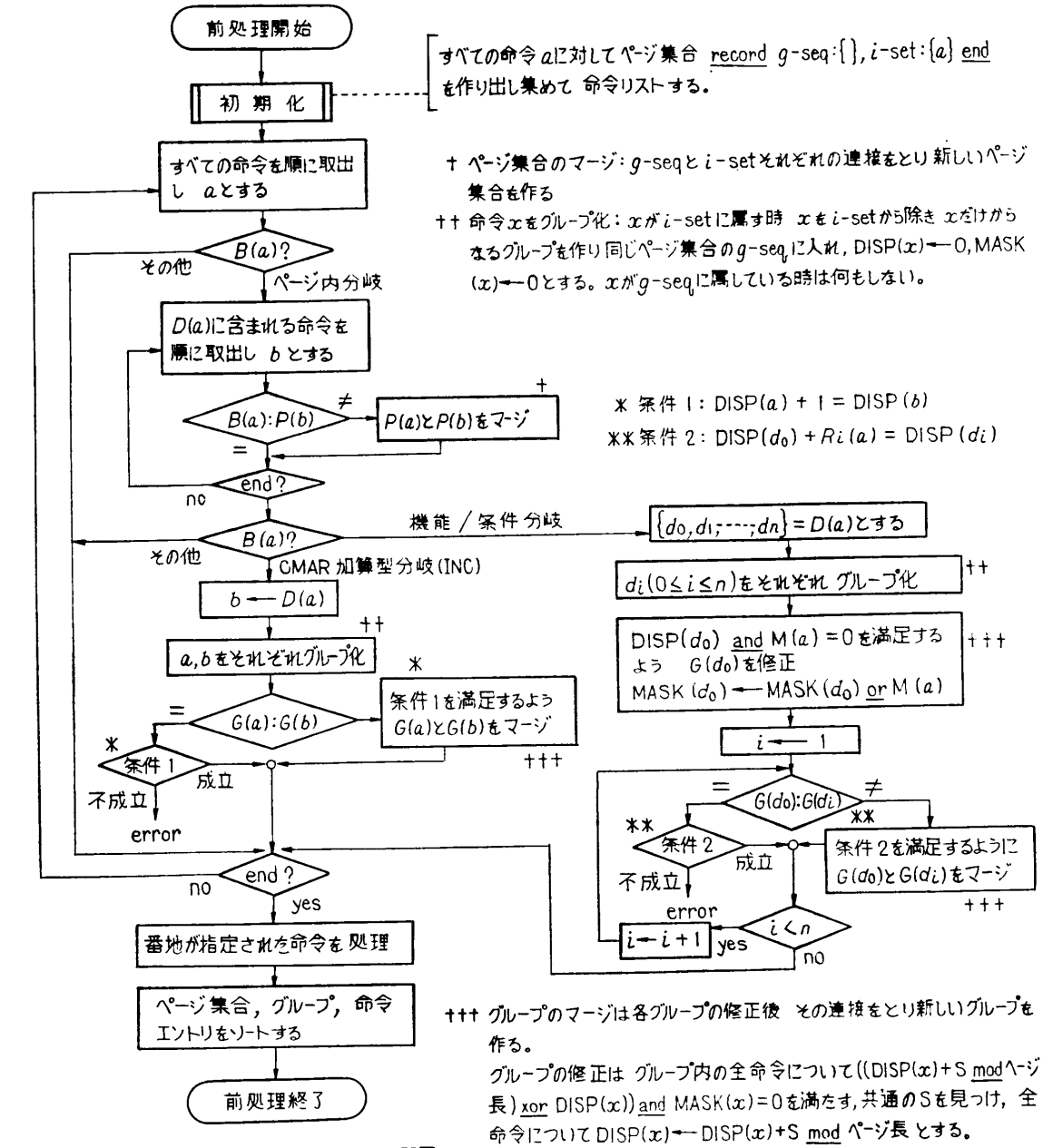
### 4.3 本 処 理

本処理は，作業用テーブル上で動作し，試行とバックトラックを中心とする比較的単純だが高速の動作を行う構成とした．その主要部のアルゴリズムを**図7**に示す．基本的には，ページ集合が同一ページに割当てなければならない命令群を表わしているので，順に，これにページを割当てるとともに，一方，グループが相対位置が決められた命令群を表わしているので，各ページに含まれるグループについて順に，それに含まれる命令をそのページの空き領域の先頭から番地条件を満足するように配置していく．いきづまれば，適当なところまでバックトラックし，別の試行に入る.

ページ集合に対してページを割当てるためのページ選択アルゴリズムについては，ページ集合の命令数とページ表中に記録されている各ページの空き領域の大きさを比較して，最初に入れることができるページを選ぶ方法 (first-fit) と，最も空き領域の多いページを選ぶ方法 (most-sparse) のいずれかをコマンドにより指定している．前者では，バックトラック時に割当てページを次に入るページに変更して試行を再開するため，解が存在すれば必ず見つかり，後者では，解を得るまでの平均処理時間が短いことが特徴である．なお，ページ表にリンクされたページ集合は，対応するページが初めて選択された時に，命令リスト内のページ集合とほぼ同様のアルゴリズムで番地割当てされる.

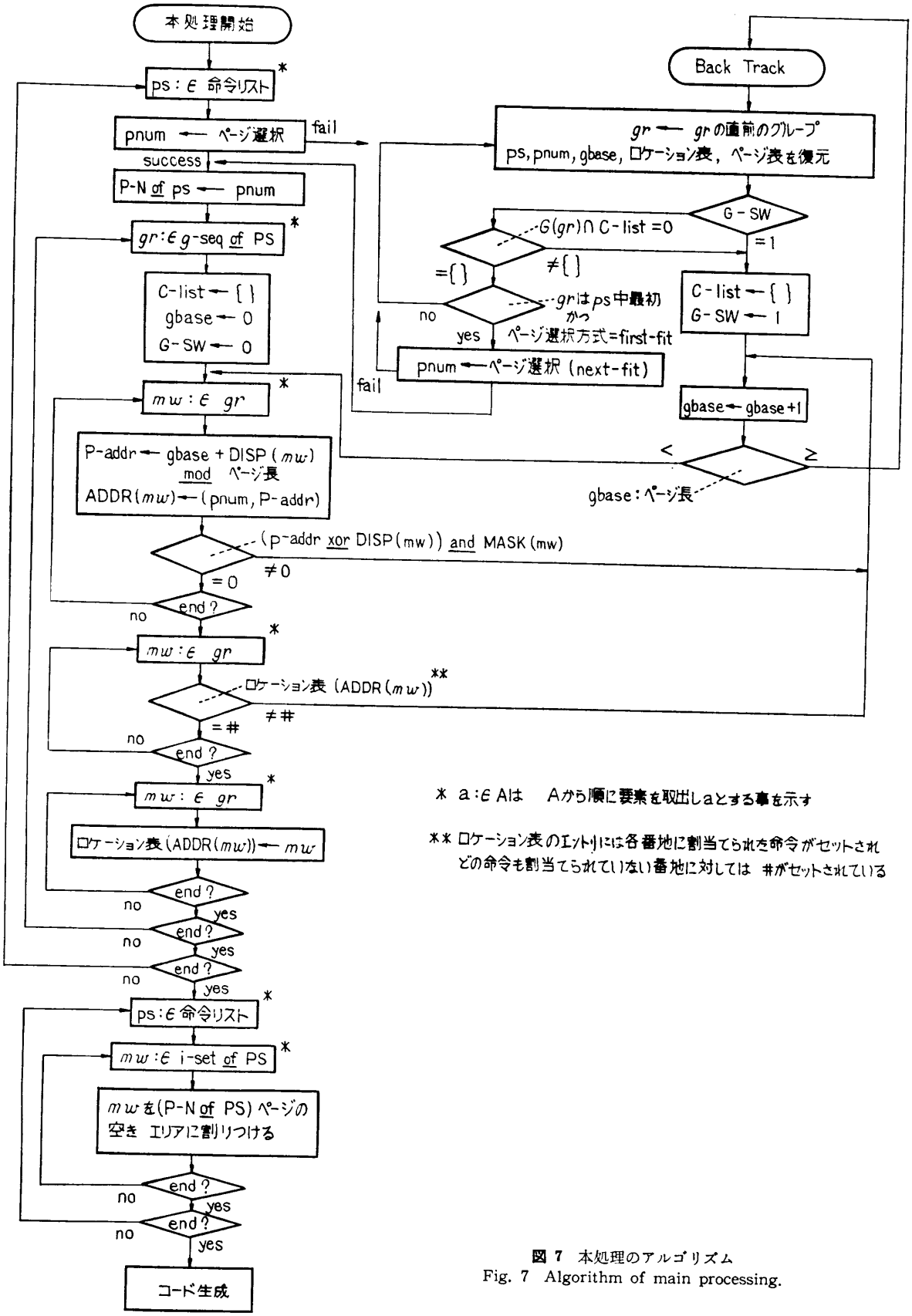
すべてのページ集合のページ割当て，および，すべてのグループの番地割当てが終了すると，グループに属していない命令，これらは，それが属しているページ集合に割当てられたページ内ならば，他の命令とは無関係にどの空領域に配置してもよく，また必要な領域数もページ集合の割当て時に確保されているので，それぞれに決められたページ内の空き領域に割当てるだけでよく，これにより，全命令の完全な番地割当てが得られる.

**表2**にリンケージ・エディタの処理時間の一例を示す．命令数が記憶領域語数に近い場合に実行時間の増




命令 $x$ に対して, $P(x)$ は所属するページ集合を, $G(x)$ は所属するグループを, $B(x)$ は分岐操作, $D(x)$ は分岐先を示す. $B(x)$ が機能分岐の時, $M(x)$ はレジスタやテスト条件からセットされるビットを1, それ以外のビットを0とするマスク. $R_i(x)$ は第 $i$ 番目の分岐先の 第0番目の分岐先に対する相対番地を示す。

**図 6** 前処理のアルゴリズム
Fig. 6 Algorithm of preprocessing.




* a : ∈ A は　Aから順に要素を取出しaとする事を示す

** ロケーション表のエントリには各番地に割当てられた命令がセットされ どの命令も割当てられていない番地に対しては ##がセットされている

**図 7** 本処理のアルゴリズム
Fig. 7 Algorithm of main processing.

**表 2** リンケージ・エディタの実行時間の例
Table 2 Sample CPU time of linkage-editor.

(単位: 秒)

| ページ選択方式 | | first-fit | | most-sparse | |
|---|---|---|---|---|---|
| 処理時間 | | 本処理 | 全処理時間 | 本処理 | 全処理時間 |
| 命令数 | 制御記憶語数 | | | | |
| 249 | 256 | 69.84 | 78.90 | 10.40 | 15.25 |
| | 384 | 0.52 | 4.84 | 0.26 | 4.32 |
| | 2,048 | 0.54 | 5.11 | 0 19 | 4.07 |
| | 4,096 | 0.54 | 6.07 | 0.48 | 4.88 |
| 964 | 1,024 | 10.36 | 20.42 | 7,200 以上 | 7,200 以上 |
| | 2,048 | 10.35 | 21.96 | 1.15 | 11.92 |
| | 4,096 | 10.50 | 23.29 | 0.94 | 13.65 |
| 2,372 | 2,432 | 31.21 | 53.46 | 15.88 | 38.74 |
| | 4,096 | 31.23 | 55.23 | 7.59 | 30.11 |

大が見られるものの，全体として実用上十分な処理速度を有している．

## 5. 番地割当てアルゴリズムの検討

### 5.1 前処理について

本論文で報告したリンケージ・エディタは第2版のものである．実用プログラムに対して，命令数が 200 程度以下では，第1版でもほぼ満足できる時間で処理できていたが，それ以上の命令数では，例えば**表 2** と同じ 964 命令の入力に対し，数時間以上の計算によっても処理が終了しないといった著しい性能低下が見られた．第2版の開発時には，この経験をもとに，次の3点を中心とする前処理の完全化をはかり，期待された性能を実現した．

① 複雑な番地制約条件を，ページ集合とグループを基本とした単純な構造として再構成する．

② 明らかに実現不可能な制約条件を，前処理の段階で見つけてエラーとする．

③ ページ集合やグループのソートを行う．

本システムでは，アセンブラ・システムとしての性格から②の場合をエラーとしているが，コンパイラ・システムではダミー命令の挿入を行いエラーを回避すべきである．この検査は，ページ集合やグループの構成中，および DISP, MASK の計算中に行われ，本処理における無駄な試行を省く．

また，①は本処理のアルゴリズムの単純化に，③は本処理に対する負担の軽減に大きな効果を持っており，後者については次節でさらに検討する．

### 5.2 前処理におけるソートの効果

前処理における，グループ内での命令エントリのソートは，本処理での gbase 選択の高速化の点から，また，ページ集合とグループのソートは次の各点から有効と認められる．

(a) 番地制約条件が**2**で掲げた①と②のみからなる時，番地割当て問題は**ビンづめ問題**の特殊な場合，すなわち容量と各要素の大きさが整数で与えられた問題になっている．ビンづめ問題と，それとについて知られている結果のみを次に掲げる[6]．

**ビンづめ問題**: $(0,1]$ に属する実数のリスト $L=(a_1, a_2, \cdots, a_n)$ が与えられた時，$L$ の要素を最小個数 $L^*$ のビンにつめよ．ただし，どのビンでも入れた要素の和が，ビンの容量である1を越えてはならない．

**定理 1** ビンづめ問題は，NP 完全である．

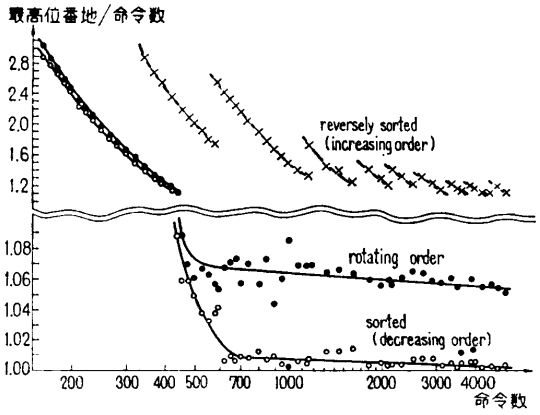
**定理 2** リスト $L$ に対して，first-fit および best-fit によりつめた時のビンの個数をそれぞれ $FF(L)$, $BF(L)$ とし，さらに $L$ が非増加順にソートされた時のそれを $FFD(L)$, $BFD(L)$ とする時，次が成立する．

$$FF(L)\leqq(17/10)L^*+2. \quad BF(L)\leqq(17/10)L^*+2.$$
$$FFD(L)\leqq(11/9)L^*+4. \quad BFD(L)\leqq(11/9)L^*+4.$$
$$\lim_{L^*\to\infty}\frac{FF(L)}{L^*}=\lim_{L^*\to\infty}\frac{BF(L)}{L^*}=\frac{17}{10}.$$
$$\lim_{L^*\to\infty}\frac{FFD(L)}{L^*}=\lim_{L^*\to\infty}\frac{BFD(L)}{L^*}=\frac{11}{9}.$$

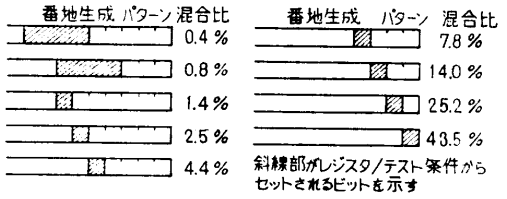
これらの結果から，命令数が多い場合にはページ集合やグループのソートにより解が改善されると考えてよい．

(b) 番地制約条件が**2**で掲げた③のみからなる場合についても，上と同様のソートの効果が見られる．first-fit アルゴリズムによる解の大きさ（命令が割当てられた最高位番地）の命令数に対する比が，入力をソートすることにより減少する模様を試算した結果を**図 8** に示す．ソートは，レジスタやテスト条件からセットされるビット位置を示すマスクを二進数として見た値について非増大順に行う．実用上重要な 500～4,000 語の範囲において，解の大きさ／命令数の比が整順の入力では 1.00～1.05 であるのに対し，乱順では 1.05～2.80 となっており，first-fit アルゴリズムのもとで入力のソートが解の改善に大きな効果を持つことがわかる．ここに掲げた結果では図 8(b)に示すビットのセットを行う機能／条件分岐と混合比を仮定したが，他の場合にも同様の傾向が認められた．

(c) 一般的な定量評価は難しいが，記憶領域に対して命令数が比較的多い条件下では，最初の解を得る




(a) first-fit アルゴリズムのもとでの解の大きさ/命令数の試算




(b) 上記の試算時に仮定した機能/条件分岐混合比

**図 8** 機能/条件分岐に対するグループのソートの効果
Fig. 8 Effect of sorting groups with functional/conditional branches.

**表 3** ソートによる本処理速度向上 (割当て命令数: 249)
Table 3 Reduction of main-processing CPU time by sorting page-sets and groups.

(単位: 秒)

| ページ選択方式 | first-fit | | most-sparse | |
|---|---|---|---|---|
| 制御記憶語数 | 256 | 384 | 256 | 384 |
| ソートしない | 7,200 以上 | 0.43 | 2,853.78 | 58.23 |
| グループのみソート | 7,200 以上 | 0.71 | 7,200 以上 | 36.25 |
| ページ集合のみソート | 4.37 | 0.70 | 1.72 | 0.54 |
| ページ集合とグループともにソート | 69.84 | 0.52 | 10.40 | 0.26 |

までのバックトラック回数が，入力のソートにより著しく減少することがある．**表 3** は，我々のシステムでグループやページ集合のソートを抑制した時の本処理の実行時間を示す．命令数が少ないため，グループのソートにより実行時間が延びたりするなど一般的傾向をそのまま反映していないと考えられる面もあるが，ソートの効果が大きい事が明らかになっている．

### 5.3 本処理について

本処理は比較的単純な構造とし，それにより試行とバックトラックを中心とする処理の実行速度を向上させた．一旦番地割当てを行った命令の再配置は，バックトラックにより割当て時の状態を再現してから，その次に条件を満足する位置へ再配置することにより行っている．

番地割当てアルゴリズム自身の改良については，発見的技法の導入も考えられ，また番地制約条件の種類を限定した特殊な問題に対しては最適解を与える決定的な手続きも存在するが，前者には作業用記憶量の増大に後者には一般化の困難さに問題があり，一般的な情況のもとでの大きな改善は困難だろう．

### 5.4 他機種のための拡張

我々のシステムは ACOS 300 のみを対象としたものだったが，他の機種のためのシステムで我々の番地割当てアルゴリズムを利用する場合に必要となることがある拡張のいくつかについて述べる．

(a) 命令の番地フィールド幅が3種類以上ある機種では，例えば**図 9** のようにページ集合を階層的に構成し，ページ割当ても階層的に行うようにすればよい．

(b) **2** で述べた(iii)と(iv)の方式において，加算されるフィールドまたはセットされるフィールドで最小の番地フィールドに完全に含まれないものがある機種では，同一のグループに属する命令が異なるページ集合に属する場合が発生する．この場合には，作業用テーブルの命令リストに次のような構造を選べばよい．

命令リストは，同様に，ページ集合とグループからなる構造を持ち，グループは，それに属す1命令の番地を決めると他のすべての命令の番地が決まるような命令群である．ページ集合は，我々のシステムと異なり，リスト構造上ページ集合に属す命令でも論理的には他のページ集合に属すもの存在し，またその逆もありうる．構造の一例を**図 10** に示す．命令エントリは，6つ組 (ID, ADDR, MASK, DISP, BASE, MOD, PSET) で与えられ，ID と ADDR は我々のシステムと同様である．MASK から PSET までは，グループに属す命令エントリについてだけ有効である．PSET は論理的に所属するページ集合を示し，リスト構造上所属するページ集合に一致する場合には null とする．本処理では次の条件を満足するように番地割当てを行う．

```
〈大ページ集合〉: record
                  p-seq : sequence of 〈小ページ集合〉;
                  i-set 1: set of 〈命令エントリ〉;
                  end
〈小ページ集合〉: record
                  g-seq : sequence of 〈グループ〉;
                  i-set 2: set of 〈命令エントリ〉;
                  end
```

**図 9** 階層的ページ集合の例
Fig. 9 Sample hierachey of page-sets.

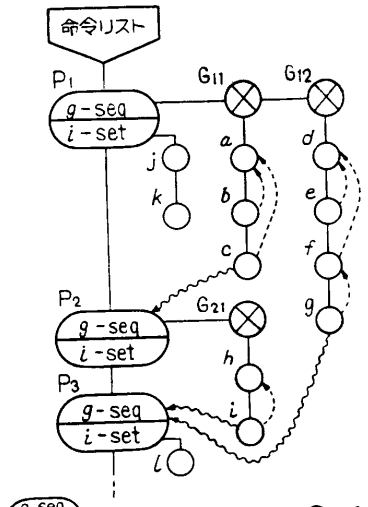



**図 10** 拡張された命令リストの構造
Fig. 10 Extented structure of instruction-list.

$$\begin{cases} \text{BASE}=\text{null} \rightarrow \text{ADDR} \\ \quad =(\text{P\_N}, \text{GBASE}+\text{DISP}\ \textbf{mod}\ \text{ページ長}). \\ \text{BASE}\neq\text{null} \rightarrow \textbf{int}\ (\text{ADDR/MOD}) \\ \quad =\textbf{int}\ (\text{ADDR}'/\text{MOD})\ \text{かつ} \\ \quad\ \ \text{ADDR}-\text{DISP} \\ \quad =\text{ADDR}'-\text{DISP}\ \textbf{mod}\ \text{MOD}. \\ (\text{ADDR}\ \textbf{xor}\ \text{DISP})\ \textbf{and}\ \text{MASK}=0. \end{cases}$$

ただし，ADDR′, DISP′ は，BASE で示された命令エントリの ADDR, DISP を表わす．GBASE は同一グループ内で共通の値 (0≦GBASE＜ページ長) とし，P_N はページ集合に割当てられたページ番号である．

前処理時には，複数のページ集合にまたがるグループは，リスト構造上最初に現われるページ集合だけに含まれるようにする．ここでも，ページ集合とグループのソートが重要である．本処理は，細部を除き我々のシステムとほぼ同様に進めることができる．ただし，PSET により参照されたページ集合は，参照された時点でページ割当てまたは条件のテストを行わなければならない．

(c) 命令の先取り方式を採用している機種については特別な考慮をしていないので不利だが，グループに属さない命令については本処理の最終段階で番地割当てする際に実行順序を配慮することにより，いくらかのコードの改善をはかることができる．

## 6. まとめ

すでに，本システムを利用しリスト処理用を中心とする[7]いくつかのマイクロプログラムの作成を行ってきた．これらはタグ・フィールドの解析等のため機能分岐を多数利用しており，番地制約条件が少なくない．しかしながら，これらに対しても制御記憶を効率よく利用する番地割当てを行い，また複数のプログラマによる開発も比較的スムーズに行われ，設計目標をほぼ満足していることを確認している．

本システム自身は，汎用のものではないので，他の機種のマイクロプログラムに対してそのまま利用できるものではないが，前節で述べた拡張を含め，その思想と方式については，ほとんど番地制約条件が存在しない特殊なマイクロプログラム方式のものを除き，多くの機種において効果が大きいと考えられる．

**謝辞** リンケージ・エディタの初期の研究に当った田村智幸氏，システム開発の一部を担当した渡久地政恭，稲葉裕一の両氏をはじめ，熱心に検討いただいた榎本・片山研究室の方々に深く感謝します．

## 参考文献

1) 萩原 宏：マイクロプログラミング，産業図書 (1977).
2) IEEE : Special Edition on Microprogram Optimization, Tr. on Computers, Vol. C-25, No. 11, pp. 961-999 (1976. 11).
3) 宮地，田村智幸，榎本，片山卓也：データ構造処理装置のためのマイクロアセンブラ・システム，電子通信学会，総合全国大会 No. 1314 (1977. 3).
4) 榎本，片山研究室：マイクロプログラミング説明書 (1976. 6).
5) 馬場敬信，藤本裕司，萩原 宏：MPG マイクロプログラム・コンパイラ，情報処理，Vol. 19, N.o 1, pp. 16-25 (1978. 1).
6) Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R. and Graham, R. L.: Worst-case Performance Bounds for Simple One-dimensional Packing Algorithms, SIAM J. Comput., Vol. 3, No. 4, pp. 299-325 (1974. 12).
7) 宮地，榎本 進，片山卓也，榎本：Lorel-2 実行システムの構成，電子通信学会，電子計算機研究会，EC 77-4, pp. 37-48 (1977. 4).

(昭和 53 年 11 月 24 日受付)
(昭和 54 年 9 月 20 日採録)