4U-5 複雑な実行順序制御方式のマイクロプログラムのためのマイクロアセンブラ

太田 寛　　迫田 行介
(株)日立製作所 システム開発研究所

1. はじめに

最近のマイクロプロセッサでは，性能を上げるため，従来にはなかった特殊な実行順序制御方式のマイクロプログラムを採用しているものがある。それに伴い，マイクロ命令のアドレス割付けなどの作業が，複雑化している。ここでは，そのようなマイクロプログラムの一例について，マイクロコード生成を効率的に行なうための方法について述べる。

2. 実行順序制御方式

対象とする実行順序制御方式を説明する。マイクロ命令は，アドレスフィールド，条件フィールド，分岐フィールドを持つ。これらのフィールドにより，第nステップで実行すべきマイクロ命令のアドレスは，以下のように決定される(図1)。

(1) まず，第n-2ステップのマイクロ命令のアドレスフィールドの値と，同ステップの分岐フィールドで指定されたデータ(以後分岐データと呼ぶ)との論理和をとる。

(2) 次に，第n-1ステップのマイクロ命令の条件フィールドで指定された条件ビットを，(1)で得られた論理和に，最下位ビットとして付加する。この値を，第nステップで実行するマイクロ命令のアドレスとする。

分岐フィールドでは，分岐データをどのバスから取り込むかを指定したり，あるいは，分岐データとして0を指定することができる。条件フィールドでは，演算結果の条件判定により条件ビットの値を決定したり，無条件に条件ビットを0または1とする(それぞれ，force0命令，force1命令による)ことができる。

この方式を利用して，分岐データによる多方向分岐や，条件ビットによる2方向分岐ができる。

3. マイクロアセンブラ開発における課題

上記実行順序制御方式のもとでマイクロコードを作成するには，以下のことを考慮する必要がある。

(1) 分岐先のマイクロ命令のアドレスは，分岐のための拘束条件を満たすように割付ける。

(2) 全てのマイクロ命令のアドレスフィールドと，条件ビットによる分岐以外のときの条件フィールドを，1または2ステップ後に実行するマイクロ命令に割付けられたアドレスに合わせて決定する。

このような処理を自動的に行なうマイクロアセンブラを開発するに際しては，

(1) マイクロソースプログラムのドキュメント性，

(2) マイクロアセンブラの処理の簡明さ，

を考慮しつつ，

(1) マイクロソースプログラム内で，アドレス割付け拘束条件を指定する方法，

(2) マイクロソースプログラム内で，アドレスフィールド，条件フィールドに設定すべき値を指定する方法，

(3) それらの指定に従って，アドレスを割付け，フィールドの実際の値を決定する方法，

を適切に設計することが課題である。

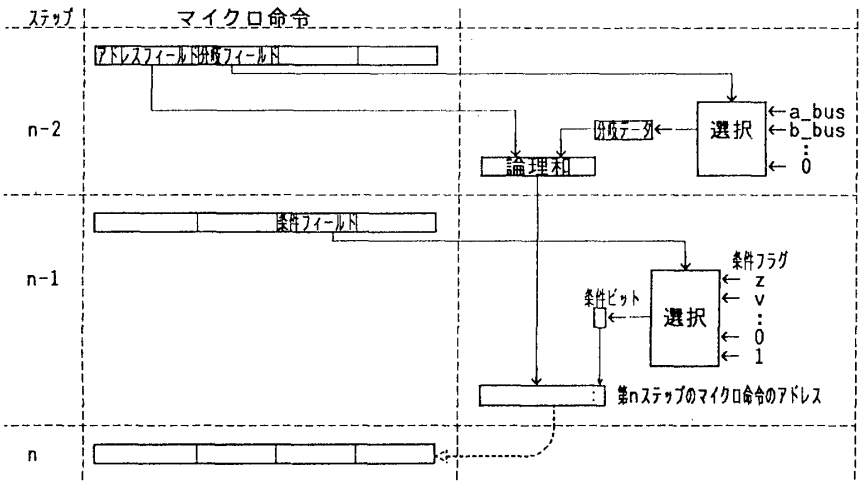


図1. 実行順序制御方式

4. 課題の解決方法

上記の課題の解決方法として，以下に述べるような，マイクロソ

A Micro-Assembler for a Microprogram with a Complex Sequence Control
Hiroshi OHTA, Kousuke SAKODA
HITACHI, Ltd.

ースプログラムの言語仕様と，それに対するマイクロアセンブラの処理方法を提案する。

4.1 アドレス割付け拘束条件指定

拘束条件は，マイクロ文(1マイクロ命令に対応するニモニック列)に付けたラベルで指定する。ラベルは，コロンで区切ってマイクロ文の先頭に付ける。ラベルはラベル名とパターンから成る。パターンはラベル名の後に付けた括弧内に，0，1，xの文字を用いて指定する。例えば，図2(a)のマイクロ文3ではlab1(x…x00x)がラベルであり，この中でlab1がラベル名，x…x00xがパターンである。

ラベルは，次の二つの拘束条件を指示する。

(1) パターン内の0，1で指定されたビット位置はその0，1の値となるようにアドレスを割付ける。xで指定されたビット位置は，0，1のどちらになっても良い。

(2) 二つ以上のマイクロ文に同一のラベル名が付けられた場合，それぞれのラベル名に付随するパターン内のxで指定されたビット位置は，各マイクロ文で互いに等しい値となるようなアドレスを割付ける。

例えば，図2(a)のマイクロ文3のラベル lab1(x…x00x)は，アドレスの第1，第2ビットが0で，他のビットについては0，1のどちらでも良いことを指示する。また，マイクロ文4のラベル lab1(x…x01x) は，第1ビットが1，第2ビットが0で，他のビットについてはマイクロ文3の対応するビット位置の値に等しいようなアドレスを指示する。なお，ラベルの付いていないマイク文には，どのアドレスを割付けても良いものとする。

4.2 アドレスフィールド，条件フィールドの指定

アドレスフィールドは，'addr = ラベル名' というニモニックで指定する。これは，そのラベル名に対応するベースアドレスの最下位ビットを除いた値を設定することを指示する。ここでベースアドレスとは，そのラベル名を持つマイクロ文のうち，パターン内のどのビット位置にも1が指定されていないものに割付けられたアドレスを意味する(例えば，図2(a)で，lab1に対しては，マイクロ文3に割付けられたアドレス)。

条件フィールドに対しては，本来のニモニックに加えて 'force(ラベル名)' というニモニックを導入する。これは，そのラベル名を持つマイクロ文に割付けられたアドレスの最下位ビットが0ならばforce0を，1ならばforce1を設定することを指示する。例えば，図2(a)のマイクロ文6の条件フィールドには，マイクロ文7に割付けられたアドレスの最下位ビットが0か1かに応じて，それぞれforce0かforce1が設定される。

なお，これらのフィールドに特に指定をしなければ，それは，記述順の実行と見なして値を設定することを指示するものとする。

4.3 処理方法

マイクロアセンブラは，まず，ラベルで指定された拘束条件を満たすようにアドレス割付けを行う。そのとき，拘束条件のきついもの(パターン内のxの数が少ないもの)から順に，条件を満たす空きアドレスを見つけてゆく。もし他のマイクロ文との関係で，初めに見つけた空きアドレスが不都合だった場合は，バックトラックして，他の空きアドレスを捜す。

アドレス割付け後に，4.2の仕様に従って，アドレスフィールド，条件フィールドに値を設定する。

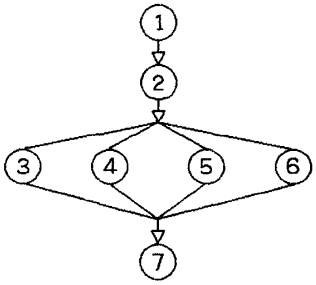
5. おわりに

マイクロアセンブラは，Prologによって記述し，約2.5Kstepのプログラムとなった。

以上に述べた方法により，複雑な実行順序制御方式のマイクロプログラムのアドレス割付けなどに要する労力が軽減され，開発効率が向上した。

	ラベル	アドレスフィールド	条件フィールド	分岐フィールド	他のフィールド
マイクロ文1		addr=lab1,		a_bus,	…….
マイクロ文2		addr=lab2,	force(lab1),		…….
マイクロ文3	lab1(x…x00x):	……	force(lab2),		…….
マイクロ文4	lab1(x…x01x):	……	force(lab2),		…….
マイクロ文5	lab1(x…x10x):	……	force(lab2),		…….
マイクロ文6	lab1(x…x11x):	……	force(lab2),		…….
マイクロ文7	lab2(x…… x):				…….

(a) マイクロソースプログラム



(b) 実行順序

図2. マイクロソースプログラム例とそれに対する実行順序