アーキテクチャ・プログラマビリティを備えたマイクロプロセサの設計手法†

前島英雄†† 木田博之†† 馬場志朗†††
赤尾泰††† 中田邦彦†††

本論文では，多様化するマイクロプロセサ応用に対して，1つのプロセサ構造をそれぞれの応用に向いたアーキテクチャに変貌可能なプログラマビリティを備えたマイクロプロセサの設計手法に関して述べている．これを実現するため，マイクロプロセサごとの命令構成やレジスタ構成を半導体チップ内に設けたテーブル上で定義する方式を提案する．まず，命令語に対してその形式やコードの制約から解放するため，解読すべき命令語のパターンと，これに対応するマイクロプログラムの先頭のマイクロ命令をペアとした命令解読定義テーブルを用いる．このテーブルとマイクロプログラム・メモリは物理的に隔離されるため，これが原因で性能低下およびサイズ増大を招く．ここではそれらを一体化したマイクロプログラム・メモリの構造も併せて提案する．次に，自由なレジスタ構成が設定できるように，十分な数のレジスタ群を高密度RAMに収納できる構造を基本とする手法を提案する．レジスタ構成をアーキテクチャに合わせてRAM上にプログラムできるように，RAMのアドレスデコーダにレジスタ指定を定義するレジスタ構成定義テーブルを一体化させる方法を採る．以上の手法により設計したマイクロプロセサ中の2つのテーブル内データを，実現すべきマイクロプロセサの命令およびレジスタ構成に合わせて記述することにより，目的のマイクロプロセサを構築できる．

1. まえがき

マイクロプロセサの応用分野はオフィスのエレクトロニクス化の波に乗って普及したパーソナルコンピュータやワードプロセサを始めとして，ビデオ・テープ・レコーダ，エアコン等の家庭電化製品，自動車のエンジンやパネル制御装置，ゲーム・マシンなど拡大の一途をたどっている．これら各々の分野に用いられているマイクロプロセサは，それぞれの応用に最適なアーキテクチャが要求されている．しかしながら，応用目的に合わせたマイクロプロセサを1つ1つ開発したのでは膨大な設計工数を必要とする．これらを少ない設計工数で短期間に仕上げる方法としては，ASIC (Application Specific IC) 技術が主流である．すなわち，ゲートアレイ方式のセミカスタムLSI[1]とスタンダードセル方式のカスタムLSI[2]の2つの方式がある．ゲートアレイLSIの場合，通常の論理ゲート (NAND, NORなど) のみの組み合わせしか使えないため，集積度，性能，価格の面で汎用マイクロプロセサには向かない．また，スタンダードセル方式のカスタムLSIの場合，マクロセル・ライブラリを充実すれば，ある程度まで目的を達することができる．しかし，この方式では種類を絞る目的で標準化したセル，マクロセルを組み合わせるため集積度，性能の点で最適化が不十分な場合が多く，専用プロセサへ適用した例[3]はあるが，これも汎用マイクロプロセサへの適用は難しい．そこで，特に命令セットに対して，ゲートアレイと同じように，一部のマスクのみの変更で応用に適合し得るマイクロプロセサを構成できると都合がよい．ここにアーキテクチャのプログラマビリティを備えたマイクロプロセサ (Microprocessor with Architecture-Programmability ; MAP) の必要性が引き出される．すなわち，FPLA (Field Programmable Logic Array) のように直接，ユーザがプログラムできるマイクロプロセサを実現したいというニーズに対して，これを実現するプロセサの設計手法が必要である．

本論文ではシステムオンチップ形のマイクロコンピュータにおけるカーネル・プロセサがアプリケーションに最適な命令セットやレジスタセットが持てるように，アーキテクチャに対し，プログラマビリティをもたせるため，プロセサを可変構造化[4]する方法について提案する．可変構造化に関して，あらゆるマイクロプロセサのアーキテクチャを想定した場合，その実現には膨大なハードウェアを必要とし事実上不可能となる．そのような可変性は，性能を犠牲にすることにな

† A Design Methodology for Microprocessor with Architecture-Programmability by HIDEO MAEJIMA, HIROYUKI KIDA (The 8th Department, Hitachi Research Laboratory, Hitachi Ltd.), SHIRO BABA, YASUSHI AKAO and KUNIHIKO NAKATA (The Micro-computer Engineering Department, Musashi Works, Hitachi Ltd.).

†† (株)日立製作所日立研究所第8部
††† (株)日立製作所武蔵工場マイコン設計部

るが，ソフトウェアによる手法（エミュレーション）が現実的である．そこで，本論文では応用分野が最も広く，応用ごとの最適化が特に強く要求される8ビットマイクロプロセサに焦点を絞り，アーキテクチャの可変性を備えながらも従来方式のマイクロプロセサと同程度のプロセサ・サイズおよび実行性能を持つプロセサ構造を提案する．

まず，アーキテクチャ・プログラマビリティの概念を述べ，命令語の分析に基づいたプロセサの構成法を示す．次に，このプロセサ構成における命令制御方式（マイクロプログラム制御方式）とレジスタ制御方式（RAM アクセス方式）を示し，それらの具体的実現法を示す．最後に，プログラマブル・アーキテクチャのマイクロプロセサを適用したシステムオンチップ形マイクロコンピュータの概要を示す．

2. アーキテクチャ・プログラマブル・マイクロプロセサの概要

2.1 アーキテクチャ・プログラマビリティの概念

図1はアーキテクチャ・プログラマブル・マイクロプロセサの概念を示したものである．プロセサは，命令形式と命令コードなど命令体系に，「依存しない部分」と「依存する部分」の2つに分けられる．

（1） 命令体系に依存しない部分：各種レジスタ群と演算回路から成る命令実行の基本部．ただし，この部分の構成により命令の実行効率が決定されるので注意深い設定が必要である（Naked Micro）．

（2） 命令体系に依存する部分：マイクロプログラム制御方式，レジスタ指定法などアーキテクチャのプログラマビリティのための命令制御部（Micro-level Software）．上記 Naked Micro を制御する．

この2つの要素によって目的とする命令セットのマイクロプロセサを構築する．すなわち，Naked Micro に目的のアーキテクチャを閉じ込めた Micro-level Software を付加することで1つの命令セットを実行できるマイクロプロセサ（Object Micro）が得られる．アーキテクチャの可変性に関する考え方は，ダイナミック・マイクロプログラミング[5]に代表される手法で存在している．しかし，これは主に中大型のコンピュータ上で実現しようとするものが多く，ハードウェアは固定的である．ソフトウェアまたはファームウェア（マイクロプログラム）によるエミュレーションが主体である．そのため，エミュレーション時の性能はあまり上がらないのが普通である．本論文で提案する

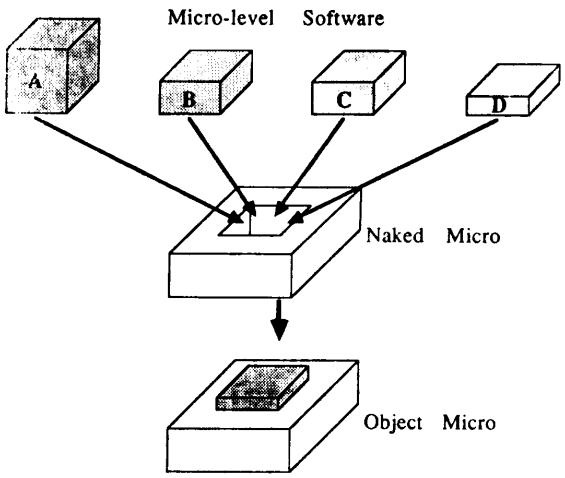

図1 アーキテクチャ・プログラマブル・マイクロプロセサの概念
Fig. 1 Concept of an architecture-programmable microprocessor.

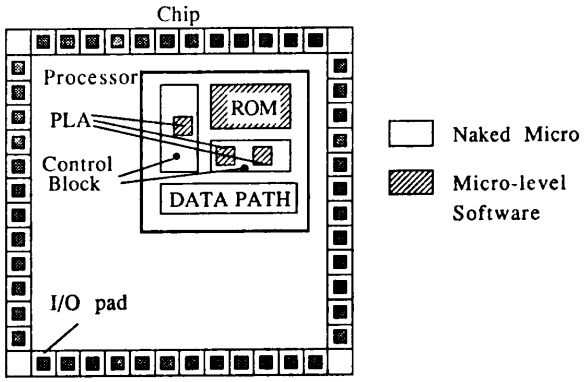

図2 アーキテクチャ・プログラマブル・マイクロプロセサの構成
Fig. 2 Construction of the architecture-programmable microprocessor.

「プログラマブル・アーキテクチャ」は，こうした固定的なハードウェアではなく，半導体チップ上で実現することを前提としている．すなわち，1つの統一したプロセサ構造の中で ROM (Read Only Memory)，PLA，配線などのマスクを変えるだけのマスク・プログラムで，種々のアーキテクチャを実現しようとするものである．**図2**に示すように，1つの半導体チップ上で，アーキテクチャに応じてプログラマブルにする部分をあらかじめ設定しておく．マイクロプログラム用 ROM やコントロール・ブロック内の PLA などの斜線部分が Micro-level Software であり，その他の部分が Naked Micro である．

2.2 アーキテクチャ・プログラマブル・マイクロプロセサ構成法

（1） 従来のプロセサ構成法

図3は従来のマイクロコード化マイクロプロセサの構成を示したものである．命令セットに最適な「専用構造」を採っており，次の2点が可変構造化のネックとなる．

（a） 命令デコーダ

命令を解読し，マイクロプログラム・メモリ(ROM)に格納される対応マイクロプログラムの先頭アドレスを作成する．複雑な命令形式を持つマイクロプロセサでは，この命令デコーダが非常に複雑になり，そのサイズも大きくなる．命令デコーダの代わりに，命令の各機能を指定するフィールドを切り出してこれをROMの入力にする方法もある．この場合もどこを切り出すかを決めるための命令形式判定回路が必要となり，命令デコーダと同様の回路となる．アーキテクチャによって変化の大きい部分である．

（b） レジスタ

ソフトウェアビジブルレジスタ，すなわち命令語から指定されるレジスタは，アーキテクチャによってその数が異なる．また，命令語中のレジスタ指定フィールドはレジスタの数によって長さ（ビット長）が異なるし，その位置はアーキテクチャあるいは命令形式によって異なる．この問題はレジスタ自身とその選択回路（レジスタデコーダ）にある．

以上述べた問題点を解決するためには，命令デコーダ，レジスタ，レジスタデコーダをマイクロプロセサ・アーキテクチャの制約から切り離す必要がある．

（2） アーキテクチャ・プログラマブル・マイクロプロセサ構成法

プロセサ中で「命令体系に依存する部分」である命令デコーダ，レジスタ，レジスタデコーダを**図4**に示した構造により，次に示す考え方で構成する．

（a） 命令デコーダ

ここでは，前に述べたように可変アーキテクチャのニーズの高い8ビットマイクロプロセサに限定し，命令語が8ビット単位に構成されている場合に絞って示す．ただし，16, 32ビット単位で構成されていても，制御は少々複雑になるが同様の考え方で扱える．

8ビットの命令語を入力として，マイクロプログラムの先頭アドレスのみを得るテーブル構成とする．単なるテーブルであれば，従来の命令デコーダと何ら変るところがないが，命令語からマイクロ命令選択までの手順をいかに手際良く行うかが課題である．

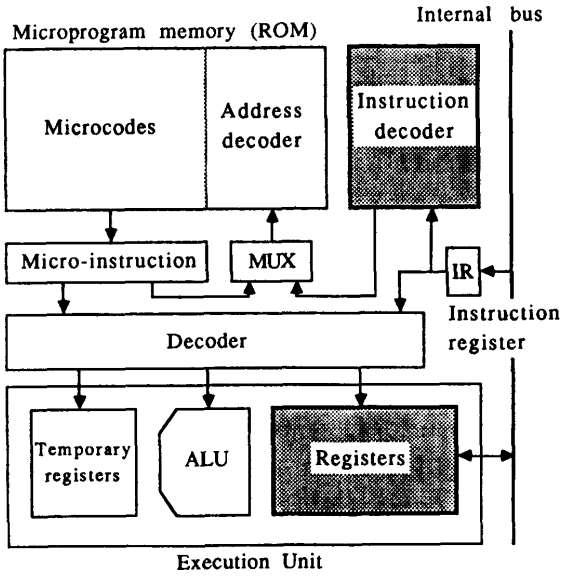

図3 従来のマイクロコード化マイクロプロセサの構成
Fig. 3 Structure of a conventional microcoded microprocessor.

（b） レジスタ

命令体系によってレジスタ数が異なる以上，十分な数のレジスタを持つ必要がある．図3におけるレジスタ群は，演算ユニット中に演算回路(ALU; Arithmetic and Logic Unit)と共に同一バス上に配置される．したがって，ALUの1ビット分の大きさに左右され，あまり小さく構成できない．そこで，**図4**に示すように，レジスタを1つのRAMに集中させて演算ユニットから切り離す．RAM 1ビットのサイズは演算ユニット内に納められるレジスタ1ビットのそれの1/4以下となり，従来の4倍程度のレジスタ数を確保したとしてもその大きさは変らない．

（c） レジスタデコーダ

命令デコーダと同様に命令語の最小構成単位（8/16/32ビット）を入力として，命令語中のレジスタ指定フィールド（論理的レジスタアドレス）を切り出し，RAMの1語を選択するアドレス（物理的レジスタアドレス）に変換する，いわばアドレス変換テーブル構成をとる．この場合も命令語からRAM中の1語を選択する手順が重要である．

以下，命令解読およびレジスタ構成を定義するテーブルの構成法を中心に述べる．

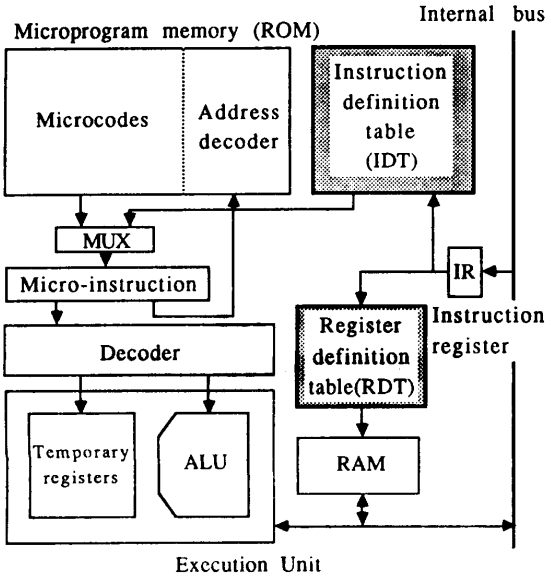

図 4 アーキテクチャ・プログラマブル・マイクロプロセサの構成
Fig. 4 Structure of the architecture-programmable microprocessor.

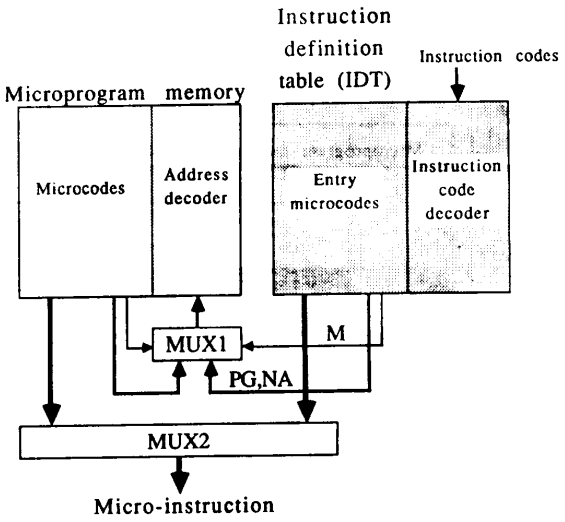

図 5 命令制御部構成
Fig. 5 Structure of an instruction control block.

3. 命令制御方式

3.1 命令制御部構成

命令解読定義テーブルの実現方法として，別報[6]に示した命令解読機能一体型の ROM 方式が有力な解となる．この方式では，ROM のアドレスデコーダ上に命令解読機能を置く．**図 5** は，命令解読定義テーブル (IDT) とマイクロプログラム用 ROM との接続関係を示したものであり，それぞれ次の働きを行う．

（1） 命令解読定義テーブル (IDT)

命令コードを入力として，これを実行するためのマイクロプログラムの先頭のマイクロ命令（エントリ・マイクロ命令）を出力するテーブルである．ただし，1つの命令コードであっても，オペランドのアドレッシング・モードを指定する部分や演算を指定する部分は異なるので，これらを分離してテーブルに書き込んでおかなければならない．そのためにテーブル記述の分離に必要な「ページ番号」をマイクロ命令中の PG (Page) フィールドに記述してある．

（2） マイクロプログラム・メモリ (ROM)

命令解読定義テーブルから読み出されるマイクロ命令中に含まれる PG フィールドおよび NA (Next Address) フィールドを入力として，これに対応する次のマイクロ命令を読み出す．テーブル (IDT) と ROM それぞれから読み出されるマイクロ命令のいずれを使用するかは，さらに，マイクロ命令中の M (Mode) フィールド1ビットが0か1かで分ける．M フィールドが“0”の場合にテーブル (IDT) を，“1”の場合に ROM を選ぶ．また，マイクロ命令中の PG フィールドを ROM のアドレスデコーダに入力している理由は，定義テーブル (IDT) と ROM を通して「ページ番号」を自由に設定できるようにするためである．

以上示した動作内容を実現する際に，命令解読定義テーブル (IDT) と ROM とを独立に配置すると命令制御部の構成が複雑になり，さらにマルチプレクサ (MUX_1, MUX_2) や配線領域が増加して命令制御部全体のサイズ増加とマイクロサイクル時間の増大を招く．そこで，定義テーブル (IDT) と ROM との合体を図り，無駄な論理回路や配線領域を省くことにより高速化を図る方法を示す．

（3） 命令解読定義テーブル内蔵 ROM

図 6 に命令解読定義テーブル (IDT) を ROM に一体化した命令制御部の構成を示した．定義テーブル IDT の命令コード判定部と ROM アドレスデコーダをマイクロ命令記憶部の両側に配置し，1つのワード線をそれぞれで共有するようにする．したがって，1つのワード線上には IDT および ROM それぞれに所属する2つのマイクロ命令を配置し，マイクロ命令中のMフィールドの内容により，ROM 中に内蔵したマルチプレクサ (MUX) でいずれか一方のマイクロ命令を選択するようにする．このような構成によって，図5と全く同じ機能を果たす上に，次に示す効果が生ま

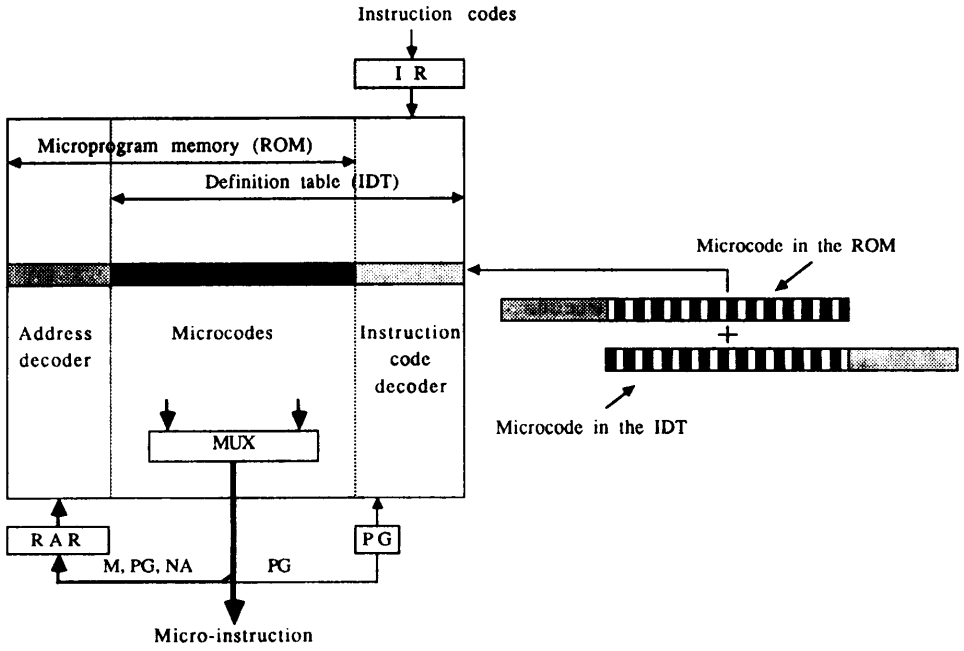

図 6 命令解読定義テーブル内蔵マイクロプログラム用 ROM
Fig. 6 A microprogram ROM with the instruction definition table.

れる.

(1) サイズ縮小

図5におけるマルチプレクサ MUX_1, MUX_2 およびその周辺の配線領域は, 図6の構造で ROM に一体化された. レイアウト試行の結果, 命令制御部全体で図3の従来方式の構成と比較して, 約 20% の面積を低減できることが確認されている. その内訳は次のとおりである. すなわち, 図5の構成では図3のものに対し, 命令デコーダおよびマイクロプログラム ROM 両方の ROM 総ビット数で比較すると, 約 4.5% の低減が図られ, これに配線領域を考慮すると, 約 15% にまで面積の低減が図られる. この効果に図6の構造による効果を加えると, 全体で約 20% の面積低減となる.

(2) マイクロサイクル時間短縮

上記サイズ縮小の効果に伴う配線負荷の低減ならびにマルチプレクサ (MUX_1, MUX_2) の省略により, 計算機シミュレーションによる評価ではマイクロ命令読出し速度は約 15% 向上することが明らかとなった.

以上の新しいマイクロプログラム用 ROM における2つのアドレッシング・モードを**図7**に示す.

(1) マイクロコード・アドレッシング

通常のマイクロプログラム制御方式と同様に, マイクロ命令中の NA (Next Address) フィールドの内容で次のマイクロ命令をアクセスするモードで, 前記した同マイクロ命令中の M フィールドを "1" にすることで得られる. 図7の例では, このモードの場合, 常に PG フィールドを "111" としているが, もちろん, M フィールドを "1" とすれば他のページをマイ

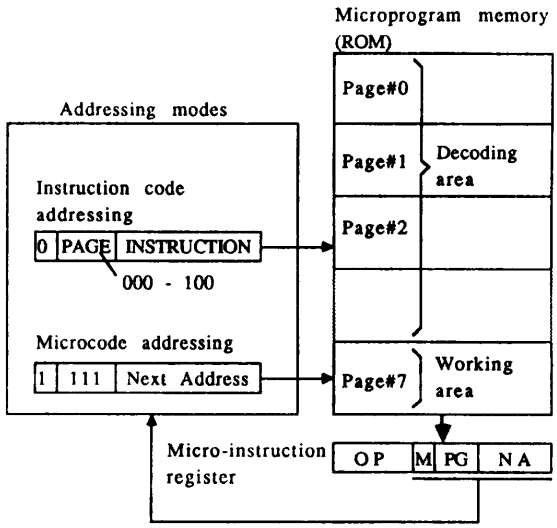

図 7 命令解読定義テーブル内蔵 ROM のアドレッシング
Fig. 7 Addressing modes of the ROM with the IDT.

クロコード・アドレッシング領域（ワーク領域）にできる．

（2） 命令コード・アドレッシング

命令語を格納する命令レジスタの内容にマイクロ命令中のPGフィールドを連結させて次のマイクロ命令をアクセスするモードで，マイクロ命令中のMフィールドが“0”の時，本モードとなる．

このように，命令コードに従ってマイクロ命令を読み出す必要のある場合には命令コード・アドレッシング・モードを選択してROM中の命令解読定義テーブル(IDT)から所望のマイクロ命令をアクセスする．また，命令コードに無関係な処理ではマイクロコード・アドレッシングを選ぶ．

3.2 命令解読定義テーブル（IDT）の構成法

図8は，8ビットマイクロプロセサの命令語，処理内容とこれに対するROM中の命令解読定義テーブル(IDT)の構成を示したものである．

（1） 命令コードの意味

図8(a)は2つの命令語のコードを示している．第1は3バイトからなるインデックス・アドレッシング・モードのロード命令，LD R, (IX+d)，第2は1バイトからなるアドレスレジスタ間接アドレッシング・モードのロード命令 LD R, (A) である．それぞれのコードには同図で示したようなアドレッシング・モード，命令の種類，レジスタ指定，変位コード(displacement)の有無を示すコードが含まれる．

（2） 処理内容と命令解読定義テーブル（IDT）構成

LD R, (IX+d), LD R, (A) の2命令が連続した場合の命令処理とこれに対応した命令解読定義テーブル(IDT)とワーク領域の構成を図8(b)に示す．ここで，前記したMフィールドはページが“111”以外すべて“0”である．

① 命令レジスタに命令が格納されるとマイクロ命令でページ“000”の命令コード・アドレッシングが指定され，この部分がアクセスされる．ここでのマイクロ命令は第2バイト目の命令フェッチを指示すると共に，マイクロコード・アドレッシングでL1番地（ページ“111”）へ分岐する．

② ここではPC (Program Counter) のMAR (Memory Address Register) へのセットと更新の命令コードによらない処理を行い，ページ“001”の命令コード・アドレッシングで命令解読定義テーブル(IDT)へ分岐する．

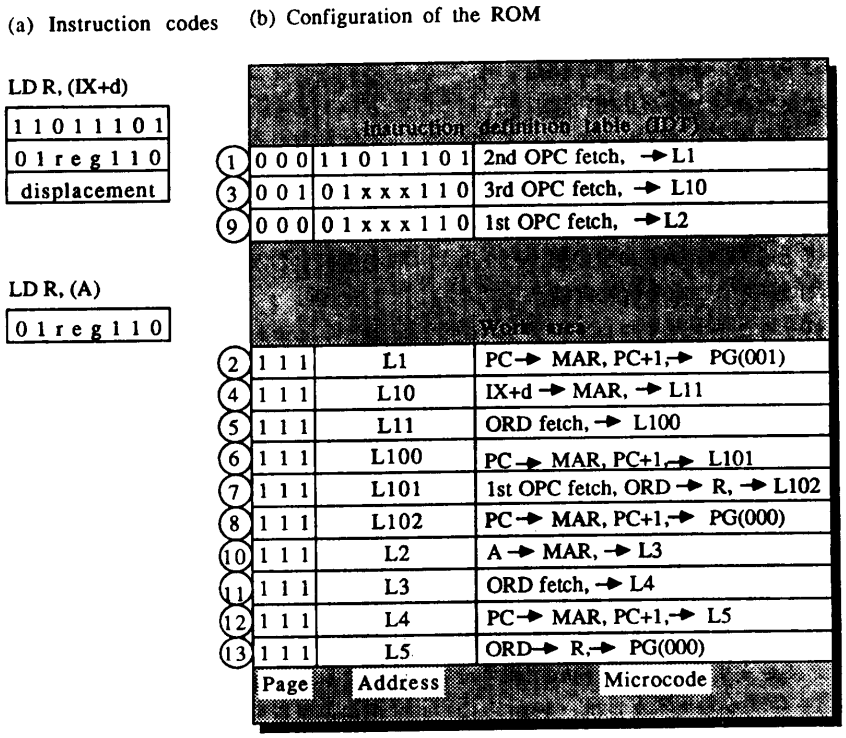

図8 8ビットマイクロプロセサにおける命令解読定義テーブルの構成
Fig. 8 Configuration of the instruction definition table (IDT) in an 8-bit microprocessor.

③ 第3バイトの目の disp の読み出しが指示され，マイクロコード・アドレッシングで L10 番地へ分岐する．

④ インデックス計算 (IX+displacement) とこの結果の MAR へのセットを行い，L11 番地へ分岐する．

⑤ 前サイクルでのインデックス計算の内容に従ってオペランドのフェッチを行い，マイクロコード・アドレッシングで L100 番地へ分岐する．

⑥ 次の命令の先取りのため PC の MAR へのセットと更新を行い，マイクロコード・アドレッシングで L101 番地へ分岐する．

⑦ 次の命令の第1バイト目のフェッチを開始し，⑤サイクルを行ったオペランド・フェッチの結果をレジスタへ格納する．このレジスタ指定方法は次章で述べるレジスタ制御方式で詳細を述べる．マイクロコード・アドレッシングで L102 番地へ分岐する．

⑧ さらに次の命令（第1バイト目または⑦でフェッチした命令の第2バイト目）フェッチの準備を行い，命令コード・アドレッシングでページ“000”の命令解読定義テーブル (IDT) へ分岐する．

⑨ ここから LD R, (A) 命令の処理が開始され，以上に示したと同様なフローで命令が実行される．

以上示したように，命令コード・アドレッシングとマイクロコード・アドレッシングを交互に繰り返すことで命令処理を行う．また，LD R, (IX+d) 命令の第2バイト目のコードと LD R, (A) 命令の第1バイト目とは同一コードであるが，命令解読定義テーブル (IDT) のページを分けることでマイクロプログラムの分岐処理が異なる．

4. レジスタ制御方式

4.1 レジスタ制御部構成

マイクロプロセサにおける各種レジスタ類は，実装密度の高い RAM に集中して収納する (RAM ベース・アーキテクチャ)．図9はこの RAM およびその周辺の構成を示したものである．RAM, RAM 内の記憶部を各種レジスタに定義するレジスタ構成定義テーブル (RDT)，このテーブルとマイクロコードからのレジスタ指定のいずれか一方を選定するマルチプレクサ (MUX) から成る．

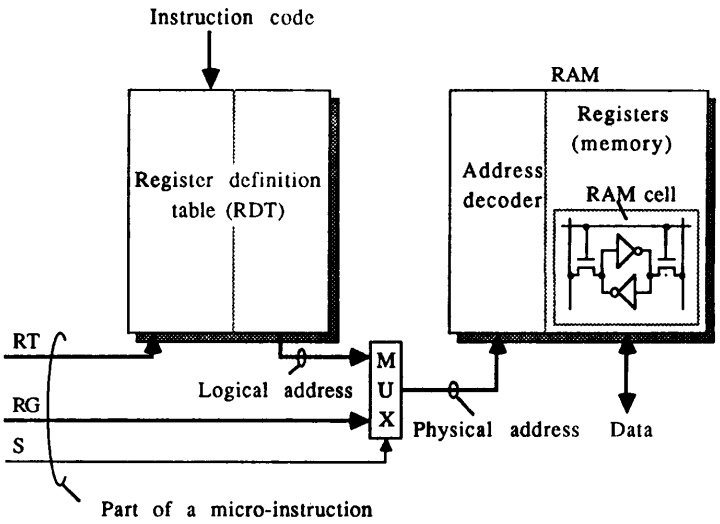

図 9 レジスタ制御部の構成
Fig. 9 Configuration of a register control part.

それぞれの働きは次のとおりである．

(1) RAM

命令実行過程で必要なワークレジスタを含めたレジスタ群を収納する．アドレスデコーダと記憶部（レジスタ群）から成る．

(2) レジスタ構成定義テーブル (RDT)

RDT は命令コードを入力として，この中のレジスタ指定フィールドの内容（論理的レジスタアドレス）を判定して RAM の対応するアドレス（物理的レジスタアドレス）を出力するテーブルである．本テーブルの機能は前記した命令解読定義テーブル (IDT) と同様の考え方である．1つの命令コードに2つのレジスタ指定がある場合にいずれのレジスタ指定フィールドを選択するかを決定するため，マイクロ命令には前記した PG フィールドに相当する RT (Register Type) フィールドが記述してある．また，本テーブルの出力とマイクロ命令中のレジスタ直接指定フィールド (RG フィールド) のいずれかを選択するマルチプレクサ (MUX) を制御するために前記したMフィールドに相当する S (Select) フィールドを設けている．

以上示したように，レジスタ構成定義テーブル (RDT) とマイクロ命令中の RG フィールドのパスを分けて構成すると，レジスタ構成定義テーブルを実現する RAM のアクセスおよびマルチプレクサ (MUX) による遅延が加わり，従来構造のものに比べて2倍以上の遅延を引き起こす．これによりマイクロサイクル時間が増加する．これも命令解読定義テーブル IDT の場合と同じである．また，命令コード中のレジスタ

指定フィールド(論理的レジスタアドレス)から RAM アドレス（物理的レジスタアドレス）への変換部と，RAM のアドレスデコーダの2種類のデコーダを必要とし，RAM まわりのサイズ増加を招く．そこで，命令制御部と同様に，RDT と RAM のアドレスデコーダとを一体化した構成を考える．

（3） レジスタ構成定義テーブル内蔵 RAM

図 10 は RDT と RAM のアドレスデコーダとを一体化したレジスタ制御部の構成を示したものである．命令から直接アクセスするための RDT と，マイクロ命令からアクセスを行うアドレスデコーダと，レジスタ部を収容する記憶部から成る．この構成は，通常のスタティック RAM と同一の構成となるので，図9に示した構成での性能低下はなくなるはずである．以下に，それぞれの構成要素の役割を述べる．

（a） アドレスデコーダ

アドレスデコーダは図9におけるマルチプレクサ (MUX) を制御するマイクロ命令中のSフィールドおよび RG フィールドを同マイクロ命令中の RT フィールドに含めた上で，RDT のコード判定部とマイクロ命令中の RT フィールドのデコーダを1つのアドレスデコーダとして合成する．このような構成により，図9の構成と全く同様の機能を果たしながら，RAM アクセスを一回で実現できると共に，マルチプレクサ (MUX) による遅延を削減できるため，レジスタ制御部のサイズ縮小と高速 RAM アクセスが可能となる．

（b） 記 憶 部

記憶部は図9の RAM 構成からアドレスデコーダを除いたものである．前記したアドレスデコーダから出力されるレジスタ選択信号を駆動するドライバを備えている．各ワードごとにレジスタが配置される．この場合のレジスタ1ビットは前述したように，従来の演算ユニットに埋め込まれたレジスタ1ビットに比較して約 1/4 のサイズで構成できる．

以上の構造によれば，レジスタ部で 25%，レジスタ制御部で 70% のサイズとなるので，全体のサイズはプロセサの可変性を保ちながらも 50% 程度に縮小される．また，図9の構成と比較して，計算機シミュレーションによれば 20% の RAM アクセス時間短縮が可能になる．

4.2 レジスタ構成定義テーブル (RDT) の構成法

図 11 は，8ビットマイクロプロセサの命令語とこの中のレジスタ指定フィールド (R*i*, R*j*) を解読する，レジスタ構成定義テーブル (RDT) の構成例を示したものである．レジスタ指定フィールドで指定されたレジスタは，それ以外の部分のコードによってその使い方が決められており，これに関しては前章に詳しく述べた．ここでは，レジスタ部分に限定して述べることにする．レジスタ間転送命令 TFR R*i*, R*j*, スタック操作命令 PUSH を例題にレジスタ構成定義テーブル (RDT) の構成を示す．

（1） レジスタ R*i* の選択

図 11 に示した命令語は8ビットであり，上位2ビットがレジスタ転送を示すコードであり，下位6ビットがレジスタ R*i*, R*j* の指定フィールドとなっている．まず，転送のソースとなる R*i* の選択は，マイクロ命令中の RT フィールドのコードを“00110”としておく．これによってレジスタ構成定義テーブル (RDT) 内のコード判定部に R*i* のフィールドのみ8種

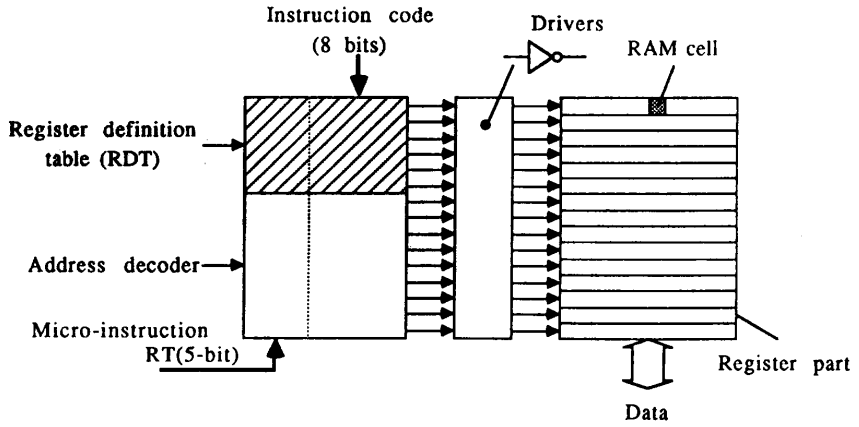

図 10 レジスタ構成定義テーブル内蔵 RAM
Fig. 10 RAM with the definition table for a free register assignment.

類 (R_0～R_7) のレジスタ選択信号を発生させる.

(2) レジスタ Rj の選択

レジスタ Ri と同様にマイクロ命令中の RT フィールドのコードを "00101" として, レジスタ構成定義テーブル内のコード判定部に Rj のフィールドのみ8種類 (R_0～R_7) のレジスタ選択信号を発生させる.

以上の2つのレジスタ指定フィールド Ri, Rj の解読結果のレジスタ選択信号は2重にあるため, これを図11に示した OR アレイによって1組のレジスタ選択信号に絞り込む.

(3) インプライドのレジスタ指定

8ビットの命令コード全体でレジスタを指定する命令, 例えばスタックポインタ (SP) を使う PUSH, POP 命令, インデックスレジスタ (IX) を用いる INCX, DECX 命令などではマイクロ命令中の RT フィールドのコードを "00010" に指定して行う.

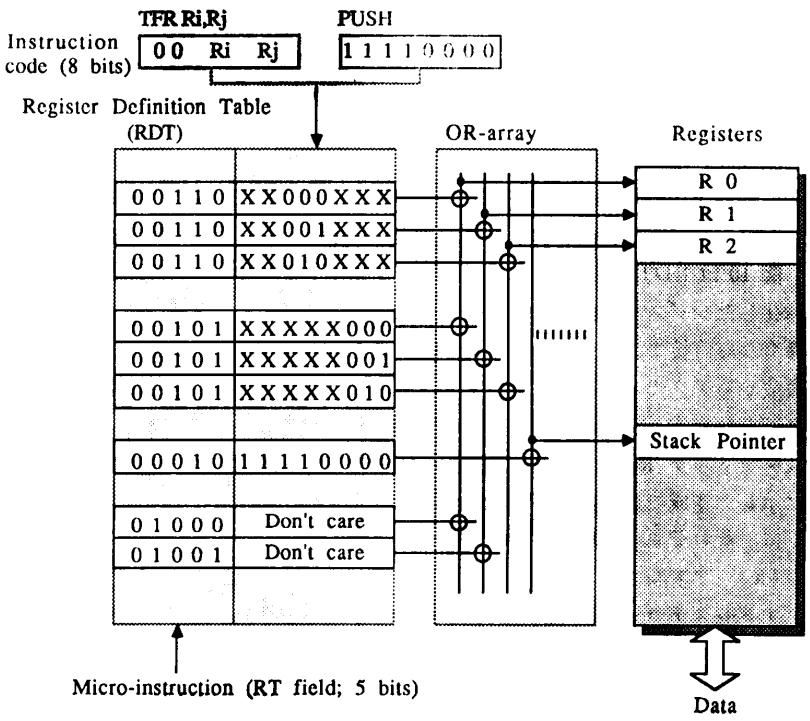

図 11 8ビットマイクロプロセサにおけるレジスタ構成定義テーブル
Fig. 11 The definition table for a register assignment of an 8-bit microprocessor.

ところで, マイクロ命令から直接レジスタを指定する場合は, マイクロ命令中に含まれる RT フィールドのコード上位2ビットを "00" 以外のコードにして下位3ビットと組み合わせてレジスタを選択するようにする. 以上の結果, レジスタ構成定義テーブル (RDT) は, 命令コードおよびマイクロコードによって指定されレジスタ選択信号を生成するコード判定部 (AND アレイ) と, この結果を統合して RAM の記憶部のワード線に絞り込む OR アレイから成る.

5. LSI 構成と機能

(1) LSI 構成

アーキテクチャ・プログラマブルなプロセサ構成法を実際に適用したシステムオンチップ形の CMOS 8 ビットマイクロコンピュータ HD 64180 のチップ写真を図 12 に示す. このマイクロコンピュータは, 2μm CMOS プロセスによって製造され, 全体で 45,000 MOS トランジスタを集積している. また, 次項に示した多くの周辺機能を持ちながら

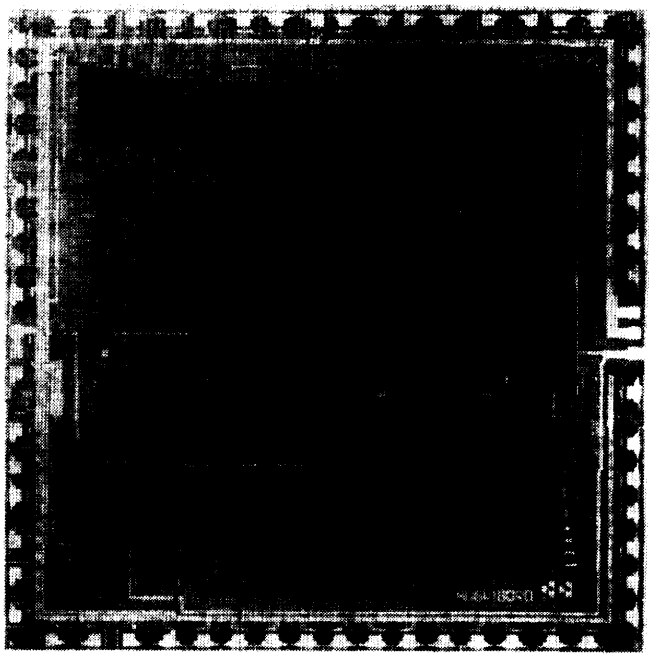

図 12 HD 64180 のチップ写真
Fig. 12 Microphotograph of the HD 64180.

表 1 HD 64180 の LSI 仕様
Table 1 LSI specifications of the HD 64180.

Items	Specifications
Process	2 μm CMOS
Integration density	45,000 MOS transistors
Chip size	5.98×5.90 mm
Power dissipation	50 mW
Clock cycle	20 MHz
Micro-cycle	10 MHz (100 ns)

表 2 マイクロコンピュータ仕様
Table 2 Specifications of the microcomputer.

Items	Specifications
Number of instructions	165
Minimum execution time of instruction	0.3 μs
Memory space	512K bytes

Peripheral I/O functions	∘ Memory management unit (MMU)
	∘ DMA controllers (2 ch.)
	∘ Serial interfaces
	  ∘ Asynchronous: 2 ch.
	  ∘ Synchronous: 1 ch.
	∘ Two 16-bit timers
	∘ DRAM refresh controller
	∘ Clock pulse generator

も 5.98×5.90 mm と極めてコンパクトなチップサイズとなった．この結果から判断して，可変性を持ちながらも，新しい構造によって非常に小型化できることを示した．さらに，応用に応じて命令やレジスタ構成を変更あるいは追加することができるため，半導体としてのプログラマビリティを飛躍的に向上することができた．

その LSI 仕様は**表 1** に示すとおりで，入力クロック 20 MHz（マイクロサイクル 10 MHz）の場合，最小命令実行時間 0.3 μs である．

（2） 機　　能

表 2 はマイクロコンピュータ仕様を示したものであり，本稿で述べたアーキテクチャ・プログラマブルなプロセサ構造は 165 の命令を有する高機能なマイクロプロセサに適用された．また，同表に示すように，多くの周辺 I/O 機能をプロセサと共に同一チップ上に集積しており，システムをコンパクトかつ低価格に実現する必要のあるプリンタなどの制御部に好適なマイクロコンピュータである．

6. ま と め

8 ビットマイクロプロセサのアーキテクチャ・プログラマビリティについて述べた．多様なマイクロコンピュータ応用に対し，目的に適合する命令機能を 1 つのプロセサ構成の中のマイクロコードを含む命令解読定義テーブル (IDT)，レジスタ構成定義テーブル (RDT) の内容をマスクで変更することで実現するアーキテクチャ・プログラマビリティを備えたマイクロプロセサの設計手法を示した．また，各要素の構成に当たり，プログラマブル構成がゆえに性能低下およびサイズ増大を招く部分を LSI 回路構成の自由度を活かし，性能を低下することなく小さなサイズで達成できた．

以上で得たアーキテクチャ・プログラマブルなマイクロプロセサ構成は，定義テーブル (IDT および RDT) に CAM (Content Addressable Memory)，ROM に RAM を用いれば，動的にアーキテクチャを切り換え得る高度なものとなる．また，EPROM (Erasable Programmable ROM) によっても大きな効果が得られる．さらに，本プロセサ構成法は，マイクロプログラミングの効果を一層高めるシステマティックな設計法としても意義が大きいと考えている．

謝辞 最後に，本稿の研究に当たり，有益な御助言をいただいた (株) 日立製作所・半導体事業部安田元氏，同武蔵工場木原利昌氏，同日立研究所増田郁朗工学博士に謝意を表します．

参 考 文 献

1) Itoh, T. et al.: A 6000-Gate CMOS Gate Array, *ISSCC Digest of Technical Papers*, Vol. 25, pp. 176-177 (1982).
2) Takeda, K. et al.: A Single Chip 80b Floating Point Processor, *ISSCC Digest of Technical Papers*, Vol. 28, pp. 16-17 (1985).
3) Maejima, H. et al.: VLSI for High Performance Graphic Control Which Utilizes Multi-Processor Architecture, *Proceedings of the International Conference on Computer Design*, pp. 586-591 (1983).
4) Maejima, H. et al.: A CMOS Microprocessor with Instruction-Controlled Register File and ROM, *ISSCC Digest of Technical Papers*, Vol. 28, pp. 12-13 (1985).
5) Wilner, W. T.: Design of the Burroughs B 1700, *Proceedings of FJCC*, pp. 579-586 (1972).
6) 前島英雄ほか：高集積マイクロコンピュータに

適したマイクロプログラム制御方式，情報処理，Vol. 23, No. 1, pp. 16-24 (1982).

(昭和62年8月12日受付)
(昭和63年7月15日採録)

前島　英雄（正会員）

昭和23年生．昭和46年東京工業大学工学部制御工学科卒業．昭和48年同大学院修士課程修了．同年(株)日立製作所入社．以来，同社日立研究所にて制御用計算機，マイクロコンピュータ，BiCMOS 技術等の研究に従事．現在，同所第8部主任研究員．工学博士．電子情報通信学会，計測制御学会，IEEE 各会員．

木田　博之

昭和33年生．昭和55年茨城大学工学部電子卒業．昭和57年同大学院修士課程修了．同年(株)日立製作所入社．以来，マイクロプロセッサの研究に従事．現在，同社日立研究所第8部所属．電子情報通信学会会員．

馬場　志朗

昭和24年生．昭和48年東京工業大学工学部電子物理卒業．昭和50年同大学院修士課程修了．同年(株)日立製作所入社．以来，同社半導体事業部武蔵工場において，マイクロコンピュータ LSI の開発に従事．昭和58年スタンフォード大学電気工学科大学院に留学，修士課程修了．現在，日立製作所武蔵工場マイコン設計部主任技師．電子情報通信学会会員．

赤尾　泰

昭和29年生．昭和52年慶應義塾大学電気工学科卒業．昭和54年同大学院修士課程修了．同年(株)日立製作所入社．昭和61年コーネル大学大学院に留学．現在日立製作所半導体事業部武蔵工場でシングルチップマイコン LSI の開発・設計に従事．

中田　邦彦

昭和34年生．昭和57年東京大学工学部物理工学科卒業．同年(株)日立製作所入社．武蔵工場にて8ビットマイコン LSI の開発・設計に従事．