マイクロプログラミング実習のための小型計算機の試作

滑川英世,* 岸上利秋*

(昭和60年9月6日受理)

An Experimental Small Computer for Microprogramming Course.

Hideyo NAMEKAWA* and Toshiaki KISHIGAMI*

Abstract — This paper describes an experimental small computer for microprogramming course.

The microprogram-controlled computer is designed and implemented with bit-slice logic, and its microinstructions are 28 bits long and can be executed in single-step mode.

The execution time of microinstructions is 750nsec, and the fetch- and execute-phases are overlapped in microcycle.

The microprograms that interpret a set of macroinstructions are resident in the control ROM, and can be directly accessed from the front panel for references.

The computer is simpler than the most widely used microprocessors, but is helpful for students to understand the priciples of computer organization.

1. まえがき

今日，マイクロプログラム制御方式は制御記憶の高速化と相まって，マイクロプロセッサから大型計算機に至るほとんどの計算機の制御技術として定着し，ますます，その重要性を増してきている。

マイクロプログラミングは，元来，ごく一部のハードウェア設計者のものであったが，特に書き換え可能制御記憶（WCS）の出現により，ファームウェア技術が確立され，ハードウェア技術者だけでなく，ソフトウェア技術者，さらに，計算機システムの利用者に至る広範囲の人々に関心がもたれるようになってきた。

本学科では，従来からハードウェア教育の一環として，論理回路の実験と機械語プログラミングとの間のギャップを埋めるため，ミニコンピュータを用いたマイクロプログラミング実習を行ってきた。これは「ハードウェア記述言語によるディジタル・システムの記述とシミュレーション」の実験テーマとともに，レジスタ転送論理の理解に非常に役立っている。

商用機のマイクロアーキテクチャは複雑であり，限られた時間内でこれを理解するのは容易ではなく，マイクロプログラミング入門用としては適さない。

そこで，機能や構造は簡単であるが，基本的な要素を備えたハードウェア教育のためのマイクロプログラム可能な計算機が望まれる。

試作した小型計算機MICRO-Xは，マイクロ命令レベルのシングル・ステップ実行が可能な8ビット並列処理のマイクロプログラマブル・コンピュータである。制御記憶として，WCS，ROMを備え，ROMにはマ

* 茨城大学工学部情報工学科（日立市中成沢町）
Department of Information Science, Faculty of Engineering, Ibaraki University, Hitachi 316, Japan

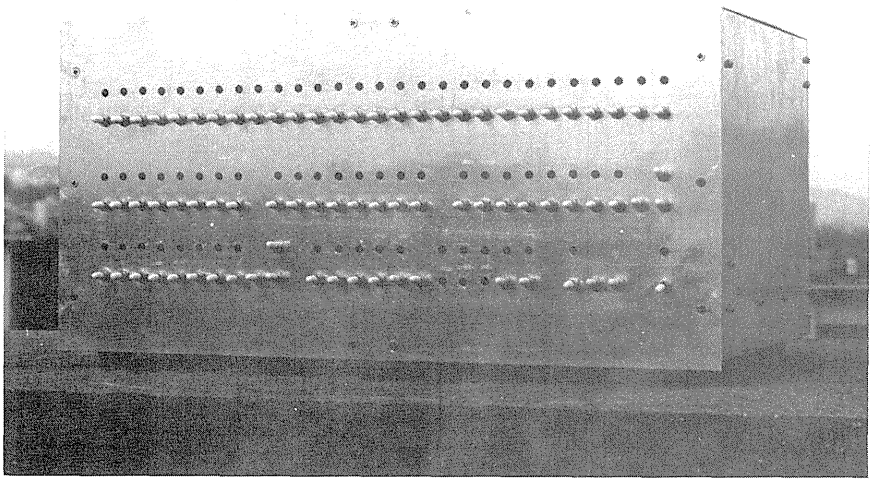

Photo. 1 Microprogrammable Computer MICRO-X.

クロ命令セットのインプリメンテーションのためのマイクロプログラムが書き込まれている。これはプログラミング例として簡単に読み出すことができるばかりでなく，シングル・ステップ実行のもとで各マイクロ命令実行後のレジスタやフラグの内容を調べることにより，マクロ命令のフェッチや実行の過程をトレースすることができる。

ここでは，MICRO-Xの概要を述べるとともに，マイクロプログラミング例，マクロ命令セットのインプリメンテーションなどについて報告する。

2. マイクロプログラム可能な計算機 MICRO-X

2.1 マイクロプログラム制御方式について

計算機の制御論理設計には，つぎの4つの方法があり，ともにレジスタ転送論理に基礎をおいている。このうち，(1)と(2)は，いわゆる，結線論理方式と呼ばれるもので，(3)は(2)の論理回路のLSI化対応と考えられる。

(1) 1状態1フリップ・フロップ法
(2) 順序レジスタとデコーダ法
(3) PLA制御
(4) マイクロプログラム制御

(1)〜(3)は，制御順序を確立するための順序回路の設計手法にあたる。状態表と励起表に基づく通常の設計手法では，制御回路の状態数が多いことから適用が難しく，また，得られる回路が不規則となることから修理や保守の点からも実用的ではない。

マイクロプログラム制御方式の概念は，1951年に*M. V. Wilkes*によって初めて提案されたものであり，その主な目的は計算機の制御論理を系統的，かつ，組織的に設計する手法を与えることにあった。

この制御方式では，制御論理はマイクロプログラムの形で集中しているので，この場合の順序回路はマイクロプログラムのアクセス順序を制御し，マイクロシーケンサと呼ばれる。マイクロシーケンサから出力されたアドレスに基づいて制御記憶からマイクロ命令が読み出され，これがパイプライン・レジスタに格納される。この間，1つ前のマイクロサイクルで読み出されていたパイプライン・レジスタ内のマイクロ命令が並行して実行される。

マイクロ命令の実行とは，各フィールド内のマイクロ操作の指定に従って，対応する制御信号を発生させることである。アドレスはマイクロ命令の次アドレス制御フィールドの指定に従って，次アドレス・フィールド内の次アドレスか，あるいは，マイクロシーケンサ内で保持するアドレスのいずれかが選択される。(Fig. 1 を参照)

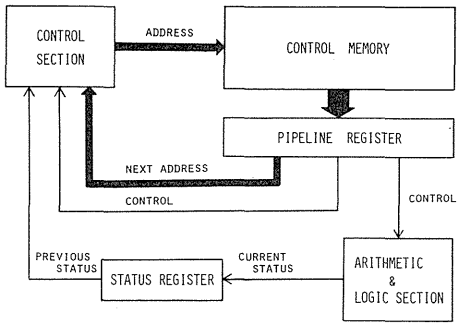

Fig. 1 Functional organization of a microprogram control unit.

マイクロ命令には，つぎのような情報が含まれなければならない。

(1) 1マイクロステップの間に同時に実行されるマイクロ操作の指定。

(2) 次アドレスおよび次アドレス制御に関する指定。

(3) マイクロ命令で使用する定数の指定。

この他に，マイクロステップの実行時間を可変にする場合にはタイミング情報が含まれる。

マイクロ命令の形式には(1)の取り扱い方の違いにより，水平型と垂直型とがある。しかし，これらは符号化の程度においてはともに両極端であり，実際にはこの中間である対角型が多く用いられている。

2.2 マイクロアーキテクチャ

マイクロプログラム制御方式の計算機では，マクロ命令セット，すなわち機械命令セットの下にさらにマイクロ命令セットがあり，これらはマイクロ命令で制御されるハードウェア・リソースとともにマイクロアーキテクチャを形成する。

Fig. 2にMICRO-Xのマイクロアーキテクチャ・レベルの機能ブロック図を示す。8ビットの内部バスがこの計算機のメイン・バスにあたり，レジスタ&算術論理演算部，主記憶部，入出力部，フロント・パネル部，制御部間のデータ転送に用いられる。ステイタス・レジスタ（4ビット），パイプライン・レジスタ（28ビット）を除いて，すべてのレジスタの長さが8ビットとなっている。

[1] マイクロ命令のワード・フォーマット

マイクロ命令に含まれなければならない情報については，2.1ですでに述べたが，その中で，(1)，(2)，(3)を同時に含めるとすると命令が極端に長くなること，次アドレスの保持にマイクロプログラムカウンタを用いていることなどを考慮して，Fig. 3のような3つのワード・タイプを決定した。各タイプに共通なフィールドとしてマイクロ命令のタイプを規定するOPフィールドとSPフィールドがある。以下，各ワード・タイプで制御できるハードウェアのマイクロ動作を説明する。

＜ワード・タイプ1＞

主記憶の読み出し／書き込み，算術論理演算，ローテイト／シフトなどを実行する。最もひんぱんに用いられるワード・タイプである。

OPフィールド……主記憶の読み出し／書き込みの起動，ローテイト／シフタのモードなどを指定する。

ALUフィールド……ALUの演算を指定する。

Lフィールド……ALUへの左入力を選択する。

Rフィールド……ALUへの右入力を選択する。

STフィールド……ALUの演算結果，または，内部バスの値を格納するレジスタを指定する。

SPフィールド……各種フラグのセット，リセットなどの特殊なマイクロ操作を指定する。

＜ワード・タイプ2＞

8ビットの即値データを内部バスに出力し，これを用いて算術論理演算を実行する。

OPフィールド…… マイクロ操作“IMM”を指定する。

ALUフィールド……ALUの演算を指定する。

DATAフィールド……8ビットの即値データを指定する。

Mフィールド……ALUへの左入力選択モードを指定する。

M＝
① 00のとき，0を入力する。
② 01のとき，Qレジスタを入力する。
③ 10のとき，STフィールドで指定されるレジスタを入力する。
④ 11のとき，使用せず。

STフィールド……ALUの演算結果，または，内部バスの値を格納するレジスタを指定する。

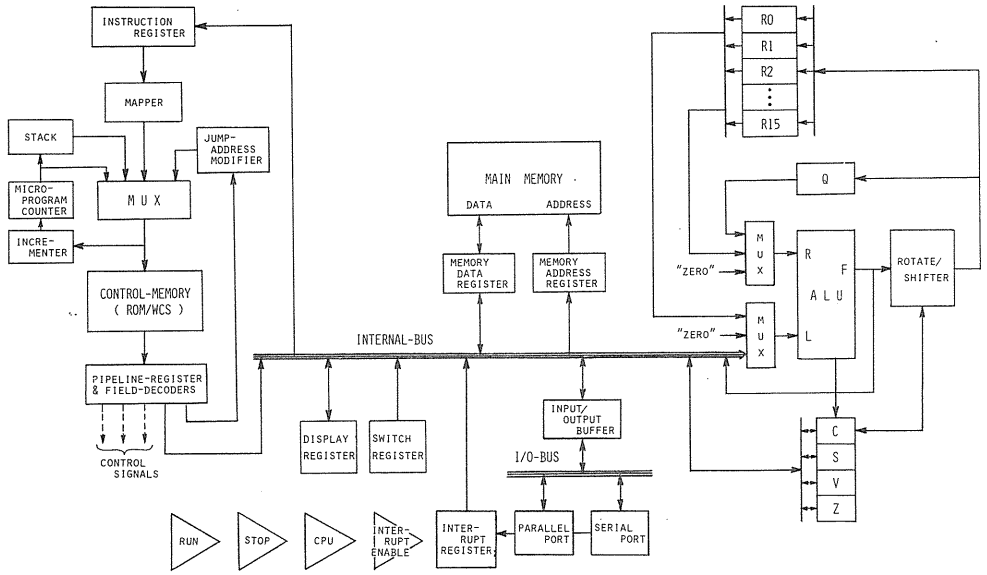

Fig. 2 Functional block diagram for MICRO-X.

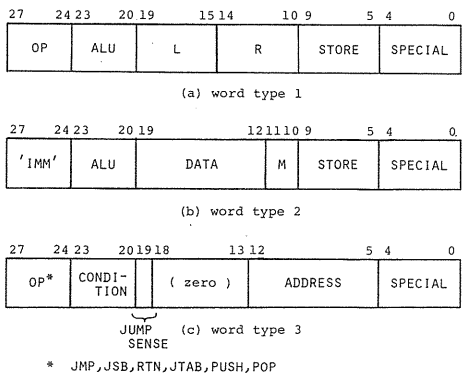

Fig. 3 MICRO-X microinstruction formats.

＜ワード・タイプ3＞

無条件ジャンプ，条件つきジャンプ，ジャンプ・サブルーチン，サブルーチン・リターンのいずれかを実行する。

OPフィールド……ジャンプの種類を指定する。

CONDフィールド……無条件か，あるいは，条件つきかを区別するとともに，条件つきの場合はその条件を指定する。また，3種類の機能ジャンプを指定できる。

JSフィールド……条件成立か，不成立のどちらによりジャンプするかを規定する。

JS＝0……条件成立のときジャンプする。

JS＝1……条件不成立のときジャンプする。

ADRSフィールド……ジャンプ先のアドレスを指定する。

SPフィールド……各種フラグのセット，リセットなどの特殊なマイクロ操作を指定する。

［2］マイクロプログラムの実行制御

マイクロプログラムの実行は制御部の中で行われる。したがって，制御部はそれ自身1つの計算機と考えることもできる。さらに，その中の制御部としてマイクロシーケンサを考えることができる。

＜マイクロ操作と次アドレスの選択＞

マイクロ命令の各フィールド内のマイクロ操作はフィールド・デコーダにより解読される。このとき，デコーダからそのマイクロ操作に関連する計算機内の制御点に，あるタイミングに従って制御信号が出され，ハードウェアの動作を引き起す。

マイクロプログラム実行の流れの変更は，ワード・タイプ3のマイクロ命令の実行により行える。この命令のOPフィールドにおかれるマイクロ操作はワード・タイプを規定するとともに，次アドレスの選択を行っている。次アドレスはつぎの4つの中から選択される。

(1) マイクロプログラムカウンタの値
(2) ADRSフィールドの値（OP＝“JMP”，“JSB”のとき）
(3) スタックからの値（OP＝“RTN”のとき）
(4) マッパからの値（OP＝“JTAB”のとき）

ここで，(1)はワード・タイプ1，2と，ワード・タイプ3のときで，かつOP＝“PUSH”，“POP”の場合の次アドレスである。

マイクロシーケンサがこれら4つの値のうちの1つを，制御記憶からつぎに読み出すマイクロ命令のアドレスとして制御記憶に与える。

＜条件つきジャンプと機能ジャンプ＞

マイクロプログラムでは，計算機内の様々な状態を示すフラグの値に基づく条件つきジャンプやジャンプ・サブルーチンが実行できなければならない。また，マクロ命令のオプコードやその他のデコードを効率的に行うために，機能ジャンプ機構を備えていることが望ましい。

マイクロ命令のCONDフィールドには条件つきジャンプの条件や機能ジャンプの種類が指定される。

機能ジャンプには，“J32”，“J10”，“JINT”がある。

J32……ジャンプ・アドレスの下位2ビットが命令レジスタのビット3，2とのビット毎のOR演算の結果に置き換わる。4方向分岐が可能。

J10……ジャンプ・アドレスの下位2ビットが命令レジスタのビット1，0とのビット毎のOR演算の結果に置き換わる。4方向分岐が可能。

JINT……ジャンプ・アドレスのビット2が割込み信号線とのOR演算の結果で置き換わる。2方向分岐が可能。

＜マイクロサブルーチンのネスティング＞

マイクロプログラムを格納する制御記憶の容量を減らすために，マイクロルーチンの共用化が計られる。特に共用ルーチンの実行後，もとの位置にもどる場合はリターン・アドレスを保存するレジスタ，または，スタック機構が必要となる。ここでは，これに対して深さ4の専用スタックを用意している。このため，サブルーチンのネスティングは深さ4まで可能となる。

スタック操作を行うのは次のマイクロ操作である。

JSB……リターン・アドレスをスタックに格納し，ジャンプを実行する。

RTN……リターン・アドレスをスタックから取り出し，これをジャンプ・アドレスとしてジャンプを実行する。

PUSH……現在の実行アドレス＋1の値をスタックに格納する。

POP……CONDフィールドとJSフィールドの指定の組み合せに従って，スタックから取り出したアドレスへジャンプするか，現在の実行アドレス＋1にすすむかが決定される。

“PUSH”と“POP”の組み合せにより，ループが構成できる。

［3］エントリ・アドレスの生成

命令レジスタの上位4ビットが“JTAB”の実行により，その値に対応する制御記憶空間内の16ケのエントリ・アドレスの1つにマップされる。

Table 1からわかるようにマッピング機構は単純であり，命令レジスタの上位4ビットをそのままエントリ・アドレスの上位4ビットとし，ビット3を常に1とすることにより得られる。

これにより，16ステップおきの16ケのマイクロプログラムに対するエントリ・アドレスが得られる。命令レジスタにマクロ命令のオプコードが入っている場合，マクロ命令の下位4ビットはマッピングの際，*don't care*となるのでさらに下位4ビットで命令を分離したいときは，機能ジャンプを用いる。

［4］レジスタ・アドレスおよび入出力ポート・アドレスの間接指定

レジスタ転送の対象となるレジスタは通常，マイクロ命令の中で指定されるが，他のレジスタ，特に命令レジスタの一部によりレジスタ・アドレスが指定できると便利なことが多い。

そこで，ワード・タイプ1のマイクロ命令のL，R，STフィールドに“EXA”や“EXB”のマイクロ操作の指定を許している。

Table 1 Mapper decoded pattern of macroinstruction.

Operation-code	Entry-address
0000xxxx	00001000
0001xxxx	00011000
0010xxxx	00101000
0011xxxx	00111000
0100xxxx	01001000
0101xxxx	01011000
0110xxxx	01101000
0111xxxx	01111000
1000xxxx	10001000
1001xxxx	10011000
1010xxxx	10101000
1011xxxx	10111000
1100xxxx	11001000
1101xxxx	11011000
1110xxxx	11101000
1111xxxx	11111000

EXA……命令レジスタのビット3，2がレジスタ・アドレスとなる。

EXB……命令レジスタのビット1，0がレジスタ・アドレスとなる。

ここで，STフィールドに“EXB”が指定されると，Rフィールドも“EXB”でなければならない。

I/Oバスを通して内部バスと入出力ポート間のデータ転送を行う場合にも，命令レジスタの一部からポート・アドレスとポート内のレジスタを指定できると便利である。

そこで，各ワード・タイプのSPフィールドに“IOR”，“IOW”のマイクロ操作の指定を許している。

IOR……入力の際，命令レジスタのビット3，2をポート・アドレスに，ビット1，0をポート内のレジスタの指定に用いて，ポート内のレジスタからIOB（入出力バッファ）へデータを転送する。

IOW……出力の際，命令レジスタのビット3，2をポート・アドレスに，ビット1，0をポート内のレジスタの指定に用いて，IOBのデータをポート内レジスタに転送する。

［5］割込み要求とその優先順位

8本の割込み要求線を備えている。これらの割込み要求には優先順位がつけられ，同時割込みの際には最高位の割込みのコードが割込みレジスタに格納される。

マイクロプログラムの中で割込み要求が待機中であるかどうかを“JINT”を用いて区別し，待機中であればその後，割込みレジスタの内容を内部バスに出力しこれを調べることにより，割込みの種類を判別できる。

これにより，マクロアーキテクチャ・レベルで割込みのベクトル化が実現できる。

2.3 ハードウェア構成

MICRO-Xはマイクロコンピュータと異なり，レジスタ&算術論理演算部と制御部とが分離され，しかも大部分はSSIやMSIといったよりディスクリートな素子で構成されているので，ブラック・ボックスではなく，その構造や機能の理解が容易である。必要ならば，回路内の端子のパルス波形をシンクロスコープやタイミング・アナライザによって観測できる。

これはハードウェア教育において，論理回路の動作とレジスタ転送論理に基づくデータ処理との関連を理解させるのに好都合である。

ここで，MICRO-Xのハードウェアの特徴を列記してみると，つぎのようになる。

＜特 徴＞

(1) 8ビット並列演算方式のユーザ・マイクロプログラマブル・コンピュータである。ビットスライス・ロジックを中心にして，約300ケのICで構成されている。

(2) マイクロプログラムのシングル・ステップ実行が可能。また，実行後の内部レジスタの内容を前面パネルに表示することができる。

(3) マイクロプログラムの実行中にも，計算機内部のレジスタの値の表示や変更が，ディスプレイ・レジスタとのデータ転送を行うことにより可能となる。

(4) マイクロ命令実行の際，マイクロ命令とそのアドレスのビット・パターンが前面パネルに表示される。

(5) 主記憶のメモリアドレス・レジスタ，メモリデータ・レジスタのビット・パターンが前面パネルに表示される。このため，主記憶の動作が目視できる。（アドレス空間＝0～255）

(6) 制御記憶として，ROM，WCSがあり，前面パネルのスイッチにより切り換えられる。ROMを選択すると，MICRO-Xのマクロ命令セットが機能する。また，WCSを選択することにより，前面パネルのスイッチ操作によりマイクロプログラミングが可能となる。(アドレス空間＝0～255)

(7) 各種のフラグの状態が前面パネルに表示される。これらは前面パネルのスイッチ操作により変更できる。

(8) 簡単ながら割込み機能をもち，その機構のON，OFFをマイクロ命令レベルで制御できる。

(9) 入出力機器接続のための入出力ポートを備えており，CRTターミナルやシリアル・プリンタが接続できる。

以下に，マイクロアーキテクチャ実現のためのハードウェアの説明にはいる。

［1］マイクロサイクルとマイクロ命令実行のタイミング

Fig. 4 にMICRO-Xのクロック・パルスとともに，マイクロ命令の実行タイミングを示す。パイプライン方式であるので，マイクロ命令の読み出しと実行をオーバラップさせている。各マイクロサイクルの初めで，OPフィールドがデコードされ，決定されたアドレスをもって同じサイクルで制御記憶の読み出しが行われる。

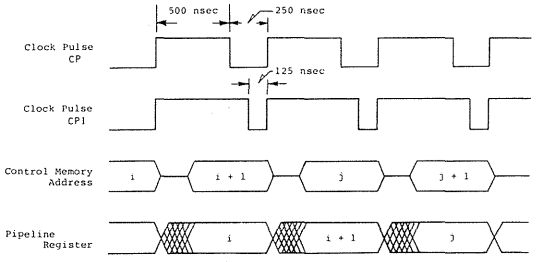

Fig. 4 Timing chart of microcycle for MICRO-X.

Fig. 4 はまた，アドレスｉ＋１の命令がアドレスｊへのジャンプを引き起すジャンプ命令であることを示している。

マイクロサイクルの時間を決定するにあたり，制御記憶のアクセス時間450nsec，演算回路，制御回路内の遅延時間などを考慮した。この時間は商用機のミニコンピュータYHP-21MXの場合の２倍となっている。

クロック・パルスは８MHzの水晶発振回路と同期式６進カウンタを用いて発生させている。

［2］マイクロプログラムのシングル・ステップおよび連続実行

クロック・パルス発生回路に単発および連続発生モードの切り換え機能を付加することにより行っている。

［3］各回路ブロックの機能と動作

機能ブロック図はマイクロプログラミングの際に意識しなければならないハードウェアを示している。MICRO-Xには，この他にいくつかの前面パネル操作回路（制御記憶，主記憶のスイッチ操作によるアクセス，内部レジスタのアクセス，マイクロプログラムのシングル・ステップ実行）と，［２］で示したクロック・パルス発生回路がある。

MICRO-Xの設計・製作にあたって，やっかいな部分は機能ブロック図に現われないこれらのハードウェアであった。

ここで，主要な回路ブロックをFig. 5 ～ Fig. 8 に示す。

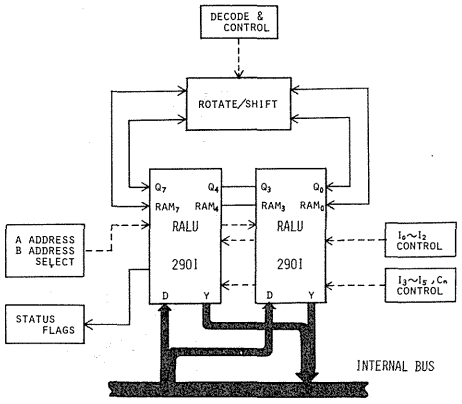

Fig. 5 Configuration for register & arithemetic and logic section.

＜レジスタ&算術論理演算部＞

４ビットスライス・ロジックであるAMD社のRALU2901を２ケ中心にして構成している。ローテイト／シフタは入力マルチプレクサで構成されている。９種類の加減算と５種類の論理演算が可能となっている。

マイクロプログラムの中で使える8ビット長の内部レジスタが16ケあり，使い方はユーザーの自由である。

ステイタス・レジスタは演算結果の状態情報である最上位桁からの桁上り（*carry*），結果の符号ビット（*sign*），オーバフローの有無（*overflow*），結果がゼロであるかどうか（*zero*）を保持する。演算時にこれらのビットを取り入れるかどうかは，マイクロ命令のSPフィールドで制御できる。また，個々にもセット，リセットできるようになっている。

2901には乗除算を行うのに便利なレジスタが設けられている。

内部レジスタR0～R15は2ポートRAM形式で，同時に2つのレジスタの読み出しが可能となっている。書き込みはBアドレスでのみ可能である。

＜マイクロ命令実行制御部＞

4ビットスライス・ロジックであるAMD社のMPC2911を2ケ用いて構成している。Fig. 6では，2911の周辺回路を示しているのでパイプライン・レジスタとそのフィールド・デコーダは省略されている。

割込み信号，命令レジスタからのビットによる機能ジャンプ回路が示されている。2911の働きは，いくつかの次アドレス候補から指示に従って1つ選択することにある。

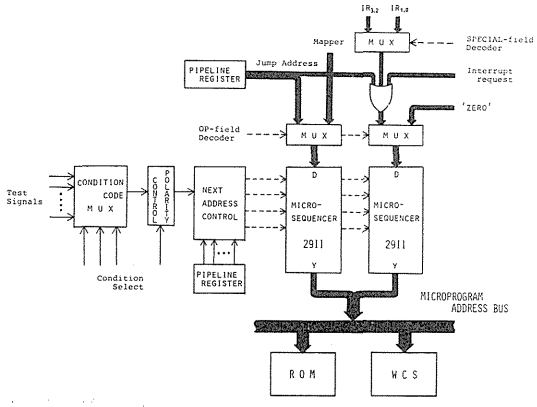

Fig. 6 Configuration for microprogram control section.

＜主記憶部＞

アクセス時間450nsecのスタティックRAMにより構成されている。

＜入出力ポート部＞

メイン・バスである内部バスとは異なるタイミングで動作するI／Oバスをもっている。I／Oバスには4つまでの周辺用LSIチップが接続できる。Fig. 8ではCRTターミナル接続のためのシリアル・ポート8251と，シリアル・プリンタ接続のためのパラレル・ポート8255が示されている。

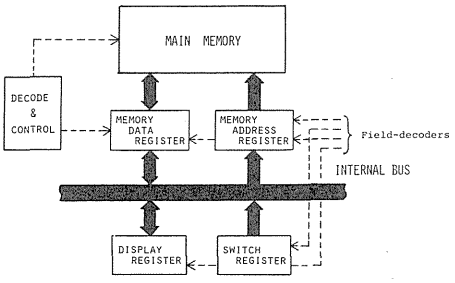

Fig. 7 Configuration for main memory and front panel section

各チップからの割込み要求線はワイヤードORがとられ，制御部へいく。割込みレジスタは割込みを要求したデバイスに対する番号を保持する。プライオリティ・エンコーダで同時割込みの制御を行っている。また，割込み信号はマイクロ操作“EI”，“DI”による割込み可能フラグのON／OFFで制御される。

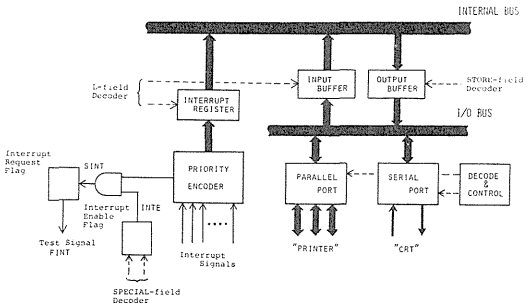

Fig. 8 Configuration for input／output section.

＜フロント・パネル部＞

Fig. 7の下部にディスプレイ・レジスタ，スイッチ・レジスタが示されている。スイッチ・レジスタは4ビット長であり，前面パネルのスイッチでその内容が変えられる。また，マイクロ操作“SWR”によりその内容を内部バスに出力できる。

3. MICRO-Xを用いたマイクロプログラミング

個々のマイクロ操作の説明には多くのスペースが要るので，ここでは，その代わりにいくつかのマイクロプログラミング例を示すにとどめる。

マイクロプログラムは機能ブロック図と個々のマイクロ操作の内容とを考え合せて，まず，ニモニック形式でコーディングした後，ユーザ自身により2進形式に変換した後，前面パネルの操作により，WCSに書き込む。

3.1 主記憶の読み出し／書き込み

任意のマイクロサイクルで行える。読み出し時はメモリアドレス・レジスタにアドレスを格納した後，つぎのサイクルで読み出しを起動し，その後のサイクルでメモリデータ・レジスタの内容を他のレジスタに転送できる。

書き込み時は書き込みの起動を行うサイクルの前までに，メモリアドレス・レジスタ，メモリデータ・レジスタの格納を終了しておく必要がある。

以下，ニモニックによるコーディング例を示しておく。ここで，ワード・タイプ1と2のマイクロ命令の各フィールドはFig. 3の順に従って一行に並べられる。ブランクのフィールドはそのフィールドがNOP（*No operation*）であることを意味する。

＜主記憶読み出しの例＞

```
 ——    OR    ZERO   R15   MAR  ——
READ   ADI   ZERO   R15   R15  ——
 ——    OR    ZERO   MDR   IR   ——
```

＜主記憶書き込みの例＞

```
 ——    OP1   ZERO   R14   MAR  ——
 ——    OR    ZERO   R14   MDR  ——
WRTE   OP1   ZERO   R14   R14  ——
```

3.2 算術演算・論理演算

＜減算の例＞

```
— SUBL  MDR    R0     R0     ENAF
```

＜ANDの例＞

```
— AND   ZERO   EXB    EXB    NAC
```

3.3 ローテイトとシフト

ワード・タイプ1のOPフィールドで指定でき，ALUの演算結果に操作できる。SPフィールドでローテイト／シフトの方向を指定する。

＜右ローテイトの例＞

```
 RS    ADD  ZERO   EXB   EXB   R
```

＜左算術シフトの例＞

```
 ARS   ADD  ZERO   EXB   EXB   L
```

3.4 入出力

入出力ポートと内部バスの間にIOBがあるので，データ転送が2段階となるので注意。

＜入力の例＞

```
        JMP   INPA                        IOR
                :
INPA : ——     OR    IOB     ZERO   R0    ——
```

＜出力の例＞

```
        ——    OR    ZERO    R0     IOB   ——
        JMP   START+1                     IOW
```

3.5 ジャンプとジャンプ・サブルーチン

ワード・タイプ3のOPフィールドに“JSB”，“RTN”を指定するときにはCONDフィールドは“UNCD”を指定しなければならない。つまり，条件つきジャンプ・サブルーチンやサブルーチン・リターンは行えない。

また，機能ジャンプを用いるときは，ジャンプ・アドレスの下位2ビットは0である必要があり，このとき，4方向分岐となる。

＜機能ジャンプ例1＞

```
 JMP   CTEST    ——   J10
```

＜機能ジャンプ例2＞

```
 JMP   NXBYTE   ——   J32
```

3.6 マクロ命令のフェッチと実行

マクロ命令セットをインプリメントする際に必要となるフェッチ・ルーチンの一例を示し，マッパの起動例を示す。

```
START :  ——     OR     DISP   ZERO     R15   ——
         JMP    INTCHK        RUN=1           CCPU
         JMP    *-1
*
INTCHK : JMP    FETCH         ——              JINT
*
FETCH :  ——     OR     ZERO   R15      MAR   ——
         READ   ADI    ZERO   R15      R15   ——
          ——    OR     MDR    ZERO     IR    ——
         JTAB
```

4. MICRO-Xに対するマクロ命令セット

4.1 マクロアーキテクチャ

マクロ命令には1バイト命令と2バイト命令があり，それらのフォーマットをFig. 9に示す。この中で指定子1，2の意味は命令グループにより異なる。

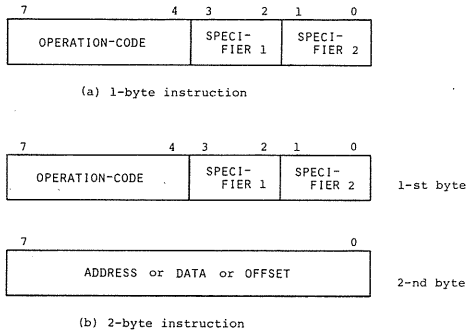

Fig. 9 MICRO-X macro-instruction formats.

A, B, C, Dの名前をもつアキュムレータがあり，内部レジスタR0～R3を割り当てている。また，インデックス・レジスタ，スタック・ポインタ，プログラムカウンタにそれぞれ，R13, R14, R15を割り当てている。

ステイタス・フラグには，*carry*, *sign*, *overflow*, *zero*がある。

メイン・メモリのアドレス空間は0～255であり，スタックはメモリ内でアドレスの低い方向にのびる。

インプリメントされている命令は全部で56種類である。

以下にグループ別にその概要を説明する。

●メモリ参照命令グループその1

2バイト命令で，第1バイトの指定子1がアドレッシング・モードを，指定子2がアキュムレータを指定する。

第2バイトはこのとき，アドレス，または，即値データがおかれる。

ロード命令，加算命令，論理演算命令などがこのグループにはいる。

$$\text{指定子1}=\begin{cases}00 & \text{イミディエイト}\\ 01 & \text{直接}\\ 10 & \text{間接}\\ 11 & \text{インデックス}\end{cases}$$

$$\text{指定子2}=\begin{cases}00 & \text{アキュムレータA}\\ 01 & \text{アキュムレータB}\\ 10 & \text{アキュムレータC}\\ 11 & \text{アキュムレータD}\end{cases}$$

ストア命令には，イミディエイト・モードはなく，指定子1＝00のときも直接アドレッシング・モードとなる。

●分岐命令グループ

2バイト命令で，指定された条件が成立したとき，命令実行後のプログラムカウンタに第2バイトのオフセット値（－128～＋127）が加算されてブランチ先が決まる。いわゆる，プログラムカウンタ・リラティブである。

●レジスタ参照命令グループ

1バイトの命令で，ローテイト／シフト，クリア，コンプリメントなどを実行する。

●入力／出力命令グループ

1バイト命令で，指定子1が入出力ポート（周辺用LSIチップ）の選択を行い，指定子2がポート内のレジスタを選択する。レジスタの機能はポートにより異なる。この命令グループの中には，ディスプレイ・レジスタやステイタス・レジスタをアクセスする命令が含まれる。

以下の命令はすべて1バイト命令である。

●スタック操作命令

アキュムレータA，Bを用いてスタックとのデータ転送を行う。

●インデックス・レジスタ操作命令

インデックス・レジスタのインクリメント／デクリメントが行える。また，アキュムレータA，Bとのデータ転送命令もある。

●割込み制御命令グループ

割込み可能フラグのON／OFFを行う。

●スイッチ・センス命令グループ

●サブルーチン・リターン命令，ホールト命令

以上をインプリメントするのに要したマイクロ命令のステップ数をTable 2に示す。

4.2　マクロ命令によるプログラミング

これらのマクロ命令によるデモ用マクロプログラムをFig.10に示す。

このプログラムはMICRO-Xの前面パネル上のディスプレイ・レジスタにローテイトした値を次々と表示することにより，前面パネル上の発光ダイオードの点灯箇所を左へシフトさせるものである。

MICRO-Xの使用者は，インプリメントされているこれらのマクロ命令を用いてプログラミングを行い，これをマイクロプログラムのシングル・ステップ実行モードでトレースすることにより，マクロ命令の実行過程を明確に把握できるであろう。

Table 2　Steps of microinstruction implementing macro-instruction set.

macro-instruction	*	steps of micro-instruction
Fetch routine	—	6
Add (IM)	M	11
Add (D)	M	16
Subtract (I)	M	19
Subtract (X)	M	16
Jump (D)	M	15
Jump (I)	M	18
Jump to subroutine (D)	M	18
Jump to subroutine (I)	M	21
Return	S	7
Conditional branch (T)	P	9
Conditional branch (F)	P	7
Move	R	3
Increment Decrement	R	4
Shift Rotate	R	4
Push down Pop up	S	7
No-operation Halt	—	1

* M: memory reference (2-byte)
S: stack operation (1-byte)
R: register reference (1-byte)
P: program-counter relative (1-byte)

```
Address  Machine-instruction      Mnemonic Form
00       10  01                   LDA  '01
02       98                NEXT:  DSPA
03       C0                       RAL
04       20  20                   STA  TEMP
06       58                       CLA
07       59                       CLB
08       51                       INB
09       7E  FD                   BNZ  *-3
0B       50                       INA
0C       7E  F9                   BNZ  *-7
0E       14  20                   LDA  TEMP
10       64  02                   JMP  NEXT
*
20       ...               TEMP:  BSS  1
```

Fig.10　Simple macro-program for demonstration.

5.　マイクロプログラミング実習の効果

MICRO-Xを用いたマイクロプログラミング実習により，つぎのような効果が期待できると思われる。

(1)　レジスタ転送論理への理解が深まり，これを仲介として論理回路の動作と計算機の動作を結びつけることができる。

(2)　計算機の基本構造を理解でき，従来，ハードウェア教育の中で取り上げにくかった制御部をテーマとした議論が可能となる。

(3)　ソフトウェア・シミュレータによるマイクロプログラミング実習も考えられるが，仮想マシンを相手にすることになり，特に入出力や割込み動作の正しい理解は困難であろう。

(4)　より高度なマイクロアーキテクチャをもつミニコンピュータや大型計算機のマイクロプログラミング技術の基礎を体得できる。

著者らの属する研究室の4年次学生，大学院学生にMICRO-Xを用いてマイクロプログラミング実習を行ってもらったところ，予想以上の好成績を上げることができた。以下，参考のためにその課題内容を示しておく。

［課題］　(1)　バブル・ソート
(2)　多倍長データに対する加減乗除算
(3)　COMP-X命令のシミュレーション

(4) 複数プロセスの並行動作
(5) CRTターミナルの入出力動作

6. あとがき

マイクロプログラミング実習を目的とする小型計算機の試作例を報告した。

より実用的なものとするためには，つぎのような点を改良する必要があろう。

(1) 制御記憶（WCS）と主記憶へのプログラムの格納法として，手動操作による他に，周辺機器からも行えるようにする。
(2) 主記憶空間を大きくする。
(3) WCS内のマイクロプログラムから，ROM内のマイクロルーチンを呼び出せるようにする。
(4) 主記憶の書き込み／読み出しのタイミングを改良して，1マイクロサイクル短くする。
(5) マイクロアーキテクチャとマクロアーキテクチャで用いるステイタス・フラグを分離する。
(6) マッピングに融通性をもたせる。

上記の改良の多くは，MICRO-Xでは不可能であるので，機会をつくってMICRO-X Ⅱの試作に挑んでみたいと思っている。

謝　辞

本計算機の試作にご協力いただいた須田秀和君（現NTT），豊島文喜君（現NEC），真山京子君（現三菱電機），森順子君（現日立），櫛田敏夫君（現日立）に感謝いたします。

参 考 文 献

(1) 原忠，伊藤誠：“バイポーラ・マイクロプロセッサによるマイクロプログラム可能な計算機の設計”，インターフェイス，Vol.5，No.4，CQ出版 (1979.4)．
(2) The 2900 Family Data Book, AMD Inc.
(3) G.J. Myers：“Digital System Design with LSI Bit-Slice Logic”，John Wiley & Sons (1980)．
(4) 萩原宏：“マイクロプログラミング”，産業図書 (1977)．