コンピュータと教育 31−2
（1994. 1. 21）

Microprogrammed E–MIXによるアセンブリ言語教育支援システムの設計と開発

中嶋 卓雄，荒木 祐一郎，中村 良三

熊本大学 工学部

Kunthが提案した仮想計算機MIXを拡張し，マイクロプログラムエミュレーションができるアセンブリ言語教育支援システムを設計開発した．

本論文で紹介するアセンブリ言語教育支援システムは，アセンブリ言語の学習にも利用でき，さらにマイクロプログラムレベルの実行を通してゲートの制御とデータの流れをエミュレートできる．本システムはXウィンドウ上にC言語で具現化しているので，操作性にも優れ，アセンブリ言語の教育に適している．

Design and Implementation of a System for Assembly-Language Education by Microprogrammed E–MIX

Takuo Nakashima, Yuichiro Araki, Ryozo Nakamura

Faculty of Engineering, Kumamoto University
2–39–1, Kurokami, Kumamoto-shi ,860 ,Japan

This paper describes the design and implementation of a system for educating the assembly laguage. This system uses an expanded microprogrammed MIX (E-MIX) which is based on the virtual machine MIX proposed by D.E. Knuth. It consists of emulators for MIX assembler (MIXAL) and microprogramming. These environments have been implemented in C language on X window system. This enables a student to study by himself for the assembly language and the control function of microprograms.

1 はじめに

計算機科学を専攻する学生にとって，プログラミング言語と同様に計算機アーキテクチャの教育が必要である．

プログラミング言語の分野では，単に言語の形式的な記述法とかデータ型の表現方法のみならず，コンパイラの構成法やプログラミングパラダイムなど広範な領域を教える必要があり，計算機と直接接点を持つアセンブラの教育は十分でない．また，計算機アーキテクチャの分野においても，各構成要素であるディジタル論理素子の動作特性やアーキテクチャの中心であるCPUの動作をアセンブリレベルで理解することは重要であるが，アセンブリ言語とゲートなどの構成要素とを結び付けるマイクロプログラムの動作に関する教育は十分でない．

従来，アセンブラ言語およびアーキテクチャの学習用として，言語自身の学習を直接的な目的とした簡単な計算機アーキテクチャに基づくアセンブラシミュレータや，現実のマイクロプロセッサに基づき周辺チップとの結線も意識したシミュレータが提案されている [2][3]．しかし，アセンブリ言語とマイクログプログラムの動作を統合した教育支援システムは少ない．

アセンブラ言語は計算機の構造，特にCPUの構造に依存しているため，エミュレートするCPUの選択は重要である．Z80などの現実のマイクロプロセッサに基づいたシステムも考案されている [5] が，長期的な教育には不適当であると考え，Kunthが提案した仮想計算機MIXと仮想アセンブリ言語MIXALに基づいたエミュレータを設計開発した．

MIXとそのアセンブリ言語MIXALは，現実に使われている計算機とアセンブラ言語の機能を有しているため，大学における初心者への教材としては最適である．さらにKnuthの著名な著書 [1] のアルゴリズムがすべてMIXALで記述されているため，MIXALを習得することにより，データ構造やアルゴリズムを設計する能力が身につくと考えられる．

本稿では，アセンブリ言語の教育支援とともにマイクロプログラム方式の制御ユニットの動作もエミュレーションできる教育支援システムを開発したので，その機能を紹介する．

まず，2章では，拡張仮想計算機E-MIXのアーキテクチャについて概観する．3章では，今回拡張したマイクロプログラム方式の制御ユニットの構造およびマイクロ命令の形式について説明する．4章では，具現化した教育支援システムの概要，特にユーザインターフェース環境について紹介する．

2 拡張仮想計算機 E-MIX

Knuthが文献 [1] で提案した仮想計算機MIXに基づきアーキテクチャを拡張し，新たに拡張仮想計算機E-MIXを提案する．これまで，MIXを拡張したアーキテクチャとしてDLXが提案され注目を集めているが [4]，MIXALをそのまま学習できるという点から命令は追加せずアーキテクチャのみ拡張した．

MIXは2つの演算用レジスタ，6個のインデックスレジスタ，フラグおよびメモリを持つ計算機である．1ワードが5バイトと符号部からなり，データの転送や演算は1ワード単位に処理され，MIXAL命令も1ワードで表現されいる．しかし，データやアドレスが転送されるバスの構造や演算処理をするALUなどが明確に設計されていなかったので，E-MIXではバス，ALUそして制御ユニットを追加した．図1に設計したE-MIXのアーキテクチャを示す．

以下では，制御ユニットを除く各構成要素を説明する．

PC(Program Counter) 次に実行するMIXAL命令のアドレスを保持する．PCが指すMM(Main Memory)のアドレスがIR(Instruction Register)に読み込まれ，解釈・実行される．

IR(Instruction Register) 命令の内容を保持するレジスタ．

Registers レジスタ群で，2つのアキュムレータ，6つのインデックスレジスタおよびジャンプレジスタから構成される．

ALU(Arithmetic and Logical Unit) MIXAL命令の演算を実行する．

フラグレジスタ Overflow, Grater, Equal, Less の4つのフラグから構成され，ALU の演算結果によって値が変化する．

MDR(Memory Data Register) CPU から転送されたデータを一時的に保存するレジスタ．

MAR(Memory Address Register) アクセスするメモリのアドレスを格納するレジスタ．

CU(Control Unit) 制御ユニット．マイクロプログラム方式を採用．

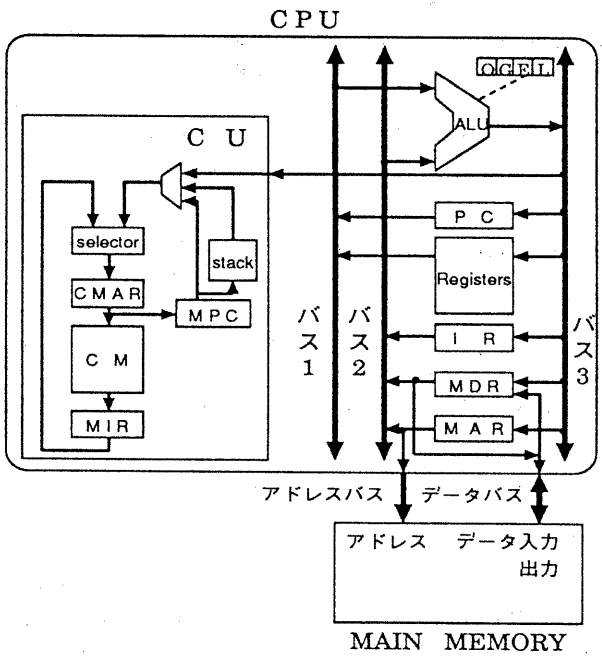

図1 E-MIX アーキテクチャ

3 マイクロプログラム制御

CPU 内のデータの流れを制御するのが制御ユニットであり，結線論理 (Wired Logis) 制御方式またはマイクロプログラム制御方式が用いられている．提案する E-MIX では，設計および機能変更の容易さなどから，一般的に用いられているマイクロプログラム制御方式を採用している．

3.1 マイクロプログラム制御方式

マイクロプログラムを構成するマイクロ命令には次のような2つの形式がある．

水平型 (horizontal type) マイクロ命令 マイクロ命令の各ビットがゲートなどの各制御点に1対1に対応している．1つのマイクロ命令は複数のフェーズに分けられ，クロックに対応してフェーズ毎に制御信号が生成される．設計は容易であるが，制御信号が増加するにつれてマイクロ命令の語長が長くなり，実用的でなくなる可能性があるため，マイクロ命令の一部を符号化することによって，語長を抑える方法が考えられている．

垂直型 (vertical type) マイクロ命令 マイクロ命令の語長の問題を解決するため，マイクロ操作の種類や数に制限を設け，まとまった動

作単位で符号化したものが垂直型の構成法である．つまり，マイクロ命令の各フィールドが操作の種類によって異なる．

3.2 制御ユニット

制御ユニットは次のような要素から構成される．

CM(Control Memory) マイクロプログラムが保存されている記憶部．

CMAR(Control Memory Address Register) 次に実行されるマイクロ命令のアドレスが格納されるレジスタ．

MIR(Micro Instruction Register) マイクロ命令を格納するレジスタ．このマイクロ命令がデコードされ制御信号が生成される．

MPC(Micro Program Counter) 次に実行するマイクロプログラムの命令のアドレスを保持する．

stack マイクロプログラムでサブルーチン処理をするとき，一時的に戻り番地を保存するためのスタック．

selector 複数の実行制御の流れから，1つの制御を選択する構成要素．

3.3 マイクロ命令

マイクロ命令を時分割して実行するクロックの機能とマイクロ命令について説明する．

3.3.1 クロック

計算機の構成要素に対する入出力のタイミングを制御するのがクロックであり，通常複数のフェーズを持ち，各フェーズの信号を繰り返し発生する．E-MIX では次に示すように，5つのフェーズに分けている．

[フェーズ1] バス1およびバス2への入出力を制御するフェーズ．

[フェーズ2] ALU での演算を制御するフェーズ．

[フェーズ3] バス3への入出力を制御するフェーズ．

[フェーズ4] メモリへの入出力を制御するフェーズ．

[フェーズ5] 順序制御およびフラグの参照による実行制御に関係したフェーズ．

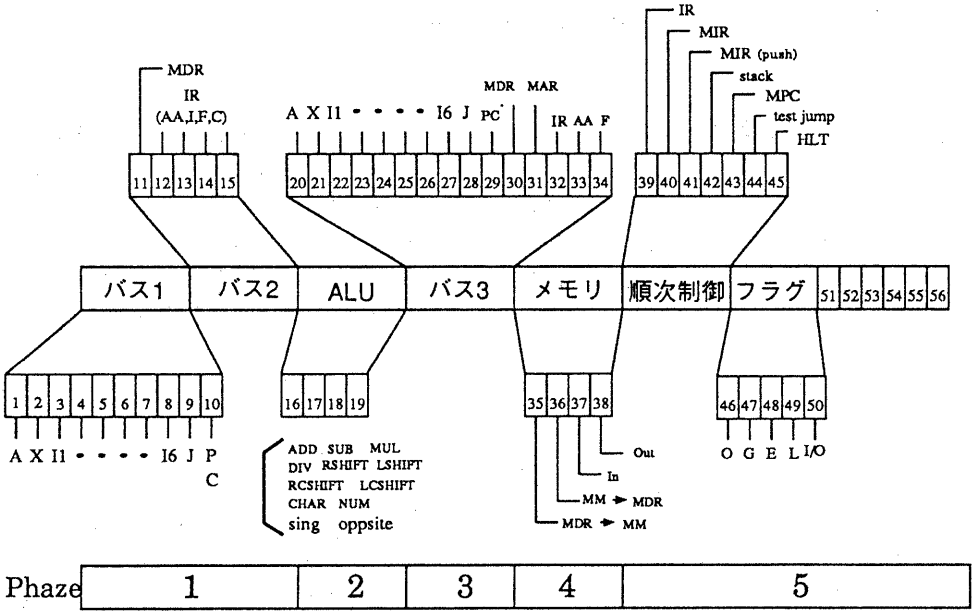

図2　1つのE-MIXマイクロ命令

3.3.2　E-MIX マイクロ命令

E-MIX のマイクロ命令は基本的には水平型であるが，演算に関連する部分についてはマイクロ命令の語長を抑えるため符号化した垂直型を用いている．1つのマイクロ命令は56ビットから構成され，デコードすることによって，ONであるビットに対応する制御信号が出力される．

図2に，E-MIXで採用している1つのマイクロ命令を示す．前述したようにクロックは5つのフェーズに分けられるが，ここでは機能をさらに分割し7つのフィールドに分けて説明する．

[バス1]マイクロ命令の1ビット目から10ビット目．フェーズ1で実行され，各レジスタおよびPCの任意のフィールドがバス1に取り出される．

[バス2]マイクロ命令の11ビット目から15ビット目．バス1と同じく，フェーズ1で実行され，IRの任意のフィールド，MDRの任意のフィールドがバス2に取り出される．12ビット目から15ビット目によって，取り出すIRのフィールドを区別している．

[ALU]マイクロ命令の16ビット目から19ビット目．フェーズ2で実行され，加減乗除，算術シフト，文字数値変換および符号の反転などを指定する．このフィールドでは12種類の演算を4ビットで符号化している．

[バス3]マイクロ命令の20ビット目から34ビット目．フェーズ3で実行され，ALUの演算結果をバス3を通して各レジスタへ転送する．

[メモリ]マイクロ命令の35ビット目から38ビット目．フェーズ4で実行され，メモリに対する読み込み，書き込みを制御する．

[順序制御]マイクロ命令の39ビット目から45ビット目．フェーズ5で実行され，マイクロプログラムの順序制御を行なう．各ビットがONのとき次のように動作する．

[39ビット目]　MIXAL命令のコードをCMARにセットする．

[40ビット目]MIRの下位13ビットをアドレスとするジャンプを意味するコードをCMARにセットする．

[41ビット目]サブルーチンへ分岐するとき，戻り番地をstackにセットする．

[42ビット目]サブルーチンから復帰するため戻り番地をstackからCMARにセットする．

[43ビット目]MPCに1を加えた次の命令のアドレスをCMARにセットする．

[44ビット目]テストによる分岐を表す．テストフィールドの値によってMPCの値を更新する．テストフィールドで指定した条件がtrueであれば2を加え，falseであれば1をCMARに加えセットする．

[43ビット目]実行を停止するHLTの信号を出力する．

フラグ　マイクロ命令の46ビット目から50ビット目．フラグレジスタの値によって実行を制御する．

51ビット目から56ビット目はリザーブしてある．

なお，今回の設計には割込みについては組み入れていない．

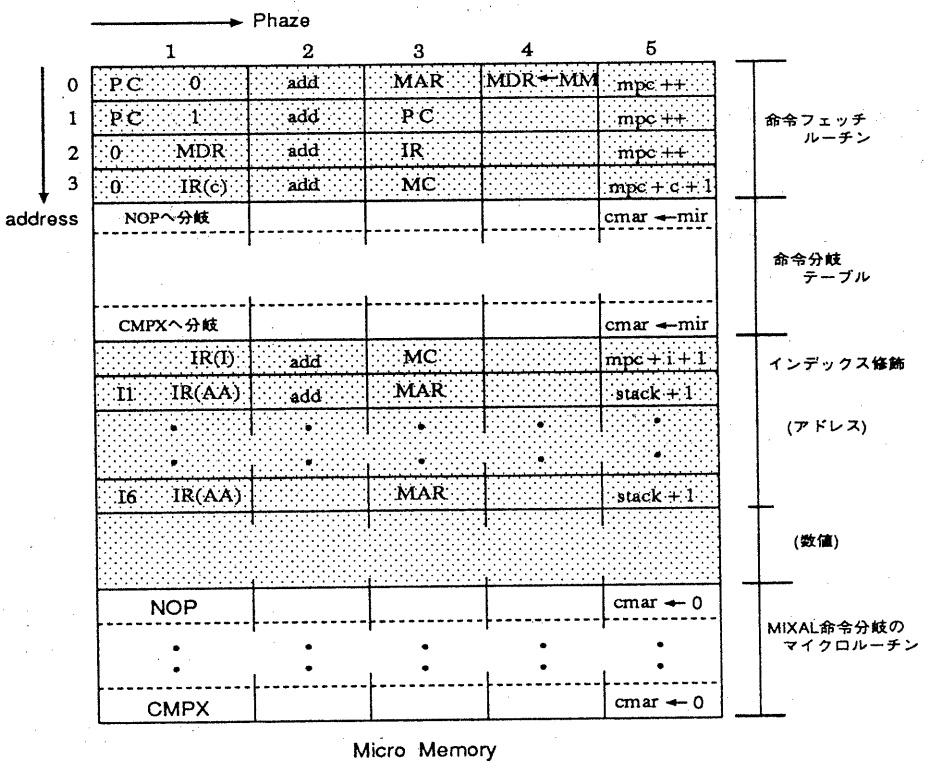

図3　E-MIX のマイクロプログラム

3.4　マイクロプログラム

マイクロメモリに格納してある E-MIX マイクロプログラムを図3に示す．ただし，図では省略した記号によって各フェーズの操作を表している．

マイクロプログラムへのアクセスは，まず，命令がフェッチされ，その命令分岐テーブルより各命令のマイクロルーチンへ分岐する．分岐テーブルより高アドレスにあるインデックス修飾部は実効アドレスを計算するサブルーチン群である．

例えば，次に示す2000番地の内容を A レジスタにロードする LDA 命令について動作を概観する．

```
LDA 2000
```

上の命令を実行すると，命令フェッチにより LDA が読み込まれ address◇で示した LDA マイクロルーチンへジャンプし，戻り番地を stack にプッシュしてから address*で示したサブルーチンへジャンプし，実効アドレスを計算し MM から内容を読み込み，最後に LDA マイクロルーチンに戻り読み込んだ内容を A レジスタに格納する．

phase	1		2	3	4	5
field	バス1	バス2	ALU	バス3	メモリ	順序制御
0	PC	0	add	MAR	MM ← MDR	mpc++
1	PC	1	add	PC		mpc++
2	0	MDR	add	IR		mpc++
3	0	IR(c)	add	MC		mpc + c + 1
				⋮		
						cmar → ◇
				⋮		
*	0	IR(I)	add	MC		MPC + i + 1
	0	IR(±AA)	add	MAR	MM → MDR	cmar ← stack
				⋮		
◇						CMAR ← *
	0	MDR	add	A		CMAR ← 0

図4　マイクロプログラムの実行例

4　教育支援システム

開発したアセンブリ言語教育支援システムを図5に示す．このシステムはUNIXワークステーション上にC言語によって具現化しており，GUIとしてXウィンドウを採用し，操作性を向上させている．

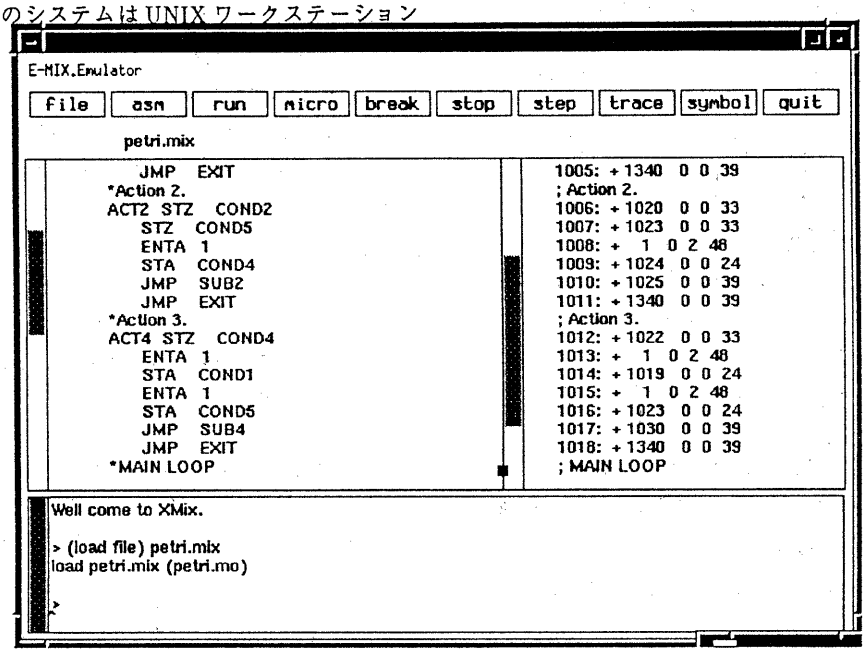

図5　アセンブリ言語教育支援システム

4.1　ウィンドウシステムの概要

システム起動時のメインウィンドウを図5に示す．メインウィンドウの各ボタンには次のような機能を割り当てている．

file MIXALのソースプログラムのロード．また，ソースファイル，マシン語コード，シンボルテーブルなど必要なファイルの印刷．

asm アセンブラを実行する．

run アセンブルされたマシン語をエミュレートする．

micro マイクロプログラムをエミュレートする．

break ブレークポイントを指定する．

stop エミュレーションを強制的に終了する．

step ステップ実行の指定．

trace トレース実行の指定．

symbol シンボル参照テーブルの表示．

quit ウィンドウシステムの終了．

さらにウィンドウシステムは次の3つのパネルを持っている．

ソースプログラム表示パネル このパネルでMIXALソースプログラムを表示する．

実行結果表示パネル 機械語の内容，エミュレーション中のレジスタの値，アセンブル後のシンボル参照テーブルの値などを表示する．

コンソールパネル システムの状態およびエラーメッセージなどを表示する．

4.2 マイクロプログラムエミュレータ

図6にマイクロプログラムをエミュレートしているときの1画面を示す．この画面はフェーズ2において，ALUにバス2からMDRの値50が入力され，バス1から値3が入力され，加算演算が実行されるときの画面を示している．

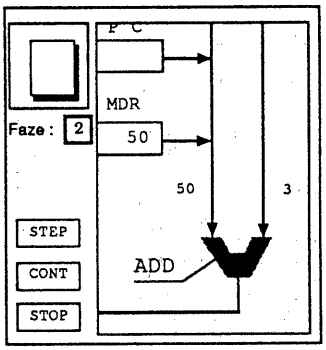

図6 マイクロプログラムエミュレータの一画面

このように，各フェーズで入力，処理，出力を関係した素子と共に表示できる画面構成としている．

5 おわりに

本論文では，仮想計算機MIXを拡張したE-MIXを考案し，アセンブリ言語教育支援を目的としたMIXALエミュレーションシステムを具現化した．

このシステムはMIXALソースプログラムのアセンブラから実行のエミュレートまで，一つの統合したシステム上で可能であり，デバッグに必要な機能も十分に備えている．さらに，1つの機械語が複数のマイクロプログラムに展開され各構成要素が制御される状態をエミュレートできる機能も備えている．

GUIとしてXウィンドウシステムを採用しているので，アセンブリ言語の初心者にも十分操作できる環境を整えており，内部動作に対する理解が深まるようになっている．

今後の課題としては，マイクロプログラムのエミュレーションが視覚的できるようにシステムを改良することと，割り込みの概念を入れ，外部との入出力などもできるように改良していくことである．

参考文献

[1] D.E. Knuth, *The Art of Computer Programming*, Vol.1, Fundamental Algorithms, Addison-Wesley, 1968.

[2] J.E. Sayers and D.E. Martin, *A Hypthetical Computer to Simulate Microprogramming and Conventional Machine Language*, Vol.20, No.4, SIGCSE BULLETIN, 1988.

[3] J.N. Fenner, J.A.Chmidt and H.A. Halab, "MASCO: The Design of a Microprogrammed Processor", Computer, pp. 41–53,1985.

[4] J.L.Hennessy and D.A. Patterson :富田眞治, 村上和彰, 新實治男訳, コンピュータアーキテクチャ, 日経BP社,1992.

[5] H.B.Diab and I. Demashkieh, *A Computer-Aided Teaching Package for Microprcessor Systems Education*, IEEE,Transaction on Education,Vol.34,No.2,May,1991.