福井大学
工学部　研究報告
第39巻　第2号
1991年9月

X Window 環境下での 68000 マイクロコンピュータシミュレータ

奥村彰二*　高田隆弘*　柳瀬龍郎*

A Simulator of 68000 Microcomputer System under the X Window Environment

Shoji OKUMURA, Takahiro TAKADA, and Tatsurou YANASE

(Received Aug. 31, 1991)

A software system is developed for simulating the operations of a 68000 microcomputer system under the X Window environment. The system not only executes precisely the machine instructions of the 68000 processor on the computer in a software model, but also provides many debugging schemes , which may be especially useful for students to study the assembly language in their laboratory works. In this system the X Toolkit Intrinsics of the X Window system is used and it makes graphical user interface to be realized with programming of an object oriented style. The features of the interactive operations on the X Window system, including various kinds of breakpoint setting, single step tracing, diassembler output and modification of memory contents, are described in this paper.

1　まえがき

計算機の利用者がアセンブリ言語を用いてプログラムを書く機会は近年非常に少なくなってきているが、計算機システムまたは機器の制御のためのマイクロコンピュータシステムの開発においては、アセンブリ言語が重要な役割を果たしている場合が多い。特に計算機の専門教育においては、機械語と1対1の対応を持つアセンブリ言語に接することは、計算機の基本的な動作のメカニズムを理解する上で大きな手助けになると考えられる。アセンブリ言語の実習において、対象とする計算機を、汎用の計算機、ワークステーションまたはパーソナルコンピュータとし、それぞれの計算機自身のアセンブラを利用して初歩的なプログラムを実行することには教育上次のような支障がある。

* 福井大学工学部情報工学科

1. 使用する計算機によりアセンブリ言語が左右される。使用できる計算機がアセンブリ教育に適していない場合もあり、計算機が更新される度に、計算機の違いによる実習内容の手直しが必要になる。

2. 高度なオペレーティングシステムを備えた計算機では、メモリやレジスタの直接アクセスや使用する命令に複雑な制約がある。

3. シングルステップ実行やブレイクポイントの設定などのデバッグ機能が使い難く、機能も十分なものではない。

4. 数多くのシステムコールの存在は初期の段階の教育にとってはそれほど必要でなく、かえって余計なトラブルの発生の原因となる。

また、マイクロコンピュータシステムの開発専用のハードウエアシステムを教育に利用することは、同時・多人数の対応の点で問題がある。

一方、アセンブリ言語の対象となる計算機のクロスアセンブラを用い、そのまま出力オブジェクトコードの実行をソフトウェアで仮想的に構成した計算機システム上で実行することは、上記の問題を回避することができ、教育上非常に有効な方法である[1]。この際、メモリ上の機械語の直接の書き換えおよびその表示、逆アセンブル表示、各種デバッグ機能の設定とその実行、等において効果的なシステム機能と、計算機と利用者との対話的なやりとりの使い易さが重要である。X Window システムは、最近たいていのワークステーションに標準的移植され、優れた計算機の利用環境を提供してくれる。これは、ネットワーク上でクライアント(アプリケーション)とサーバ(ディスプレイ)に分散した環境を前提としており、ライブラリルーチンさえあれば、計算機間の移植性は極めて優れている。特に、X Toolkit[4, 5] と呼ばれるライブラリールーチンは、オブジェクト指向の形態のプログラミングを可能にし、スクロールバー、ボタン、メニュー、対話ボックス等、標準的で使い易いユーザー・インターフェースを実現する道具を提供している。

この論文では、ワークステーション上でマイクロプロセッサ68000をCPU[2]とするマイクロコンピュータの機械語プログラムの実行をソフトウェアでシミュレートするとともに、X Window の環境の下で対話的なデバッグ機能をもつひとつのソフトウェアシステムを提案する。マイクロプロセッサの機能をソフトウェアで実現し、プログラム開発または計算機教育に役立てようとする試みは今までなされているが[3]、この論文では、新たにX Windowを用いるなど対話機能と使い易さを特に重視して開発している。68000を選んだのは、機械語として適当に高度であり、上で述べたアセンブリ言語の教育目的に沿った機能を十分具備していること、現在もよく用いられ、機器の制御など応用上重要なマイクロプロセッサであることなどの理由による。

2 CPU機能のソフトウェア上の実現

CPUの実行は、メモリ上に置いた機械語すなわち命令語を読みとり(fetch)、その命令語を解釈して決められた動作の実行(execute)を行なうことであるが、この両方の動作をワークステーション上でC言語によるプログラミングにより実現する。

68000マイクロプロセッサの一つの命令は、1から5ワード(1ワード=16ビット)までの可変長コードになっているが、そのうち常に最初の1ワードのオペレーションワードのみで命令語の種類は決定

される。基本的な命令語の種類は約100であるが、ブランチ命令における条件、アドレッシングモード、演算データサイズの指定、MOVEMやシフト命令などのように移動の方向や演算対象の指定、等が1ワードのビットパターンに織り込まれている[2]。全ての命令について、命令語の種類の数から予想される7ビットまでの固定したビット位置で命令のデコードが可能なようにはなっておらず、アドレッシングモードや演算データサイズなどの違いなどによって2つの命令の区別をしなければならない場合もある。そのために、まずオペレーションワードの上位4ビットにより大まかな分類をし、個々の命令について必要なビットパターンを照合することにより命令語がデコードされる。

D0-D7、A0-A7の16個のレジスターについて、それぞれ一つの変数に割り当てる。また、プログラムカウンター、ステータスレジスター、ユーザおよびスーパバイザースタックポインターなども変数として割り当てる。また5つのコンディションコードもソフトウェアの都合上独立な変数を割り当て、常にステータスレジスターと矛盾のないようにする。

68000のアドレス空間は16メガ バイトであるが、メモリとして同じ大きさの広域的な配列変数を確保すると、実行の際のバッファメモリ領域が大きくなり過ぎて計算機システムに大きな負荷となる。そのために、16メガ バイトのアドレス空間を1キロ バイト毎のセグメントに分け、プログラムのロードおよびプログラムの実行中の書き込みとして新たなセグメントの必要が生じた時に、1セグメントずつ確保してメモリを割り当てていく方法を取っている。従って、プログラム実行中に、あるメモリに書き込み無しでその場所を読むことは、自動的にエラーとして検出される。

ステータスレジスタの内容により、スーパバイザモードとユーザモードのどちらかに設定され、これらのモードによる規制から外れた特権違反をした命令に遭遇すれば、エラーメッセージを出力し、決められたトラップを起こすようになっている。トレースモードの設定は別のコマンドで実行するので意味を持たせていない。例外処理のうちトラップはソフトウェア上の動作であり、その動作を模擬させることに問題はないが、割り込みは外部信号によるものであるのでソフトウェア上の完全な実現は困難である。そこで指定されたレベルのオートベクタによる例外処理ルーチンへのジャンプを、後で説明するコマンドによって起動する。これにより、本来の偶発的に起こる割り込みは実現されないが、意識的に割り込みの動作に入ってどのように計算機が実行されていくかを確認することは可能である。リセットも同じようなレベルの動作で対応している。

入出力の機能はCPUの動作とは言えないが、コンピュータシステムとしては基本的なモニタコールとして装備されていなければならない。その機能がないと、ターミナルに1文字出力するにしても、シリアルラインのインターフェースの幾つかのレジスタを数回アクセスして状態を調べ、割り込みを発生するなどの一連の動作をしなければならない。モニタコールには一般的にトラップ機能を使われるが、このシステムでは、入出力のトラップが現れた時、それに相当した動作を実行するようにC言語のプログラムで組み込んでおく。このことにより、通常のアセンブリプログラムで行なう方法と同じやり方で、ターミナル上への1文字づつの文字出力、および文字列の出力、ターミナルから文字または文字列入力などができる。用意されたトラップによるモニタコールは、呼ぶときのレジスタの設定も合わせて表1に示す。このうちrawモードでの入出力はキャラクタ端末用のシステムでは使用できたが、今回のX Window対応のもでは現在のところ組み込まれていない。これらの機能の使用は、アセンブリプログラムにおける一般的なプログラミング手法によってターミナルへメッセージを出力したり、ターミナルからデータを入力したりすることを可能にしている。コマンドによりこのトラップ機能を無効にすることもできる。

レジスタの設定	機 能
d0 = 0	プログラムの終了, ストップ
d0 = 1	キーボードから1文字入力(Raw Mode), あり d0 = 入力文字
d0 = 2 , d1 = 出力文字	ターミナルへ1文字出力
d0 = 8	キーボードから1文字入力(Raw Mode) d0 = 入力文字, エコー出力なし
d0 = 9 , a0 = 文字列Address	ターミナルへ文字列出力
d0 = 10 , a0 = 入力文字列Address d1 = 入力最大文字数	キーボード文字列入力, CRで終了

表1 Trap #0によるモニタコール

3 X Window システムを用いた対話的なデバッグ機能

開発するシステムは前章で述べた68000マイクロプロセッサのCPUとしての動作の他に、ユーザの指示により、プログラムのロードと実行、メモリ上でのプログラムの修正、表示、逆アセンブル表示、ブレークポイントの設定などができるようなユーザ・インターフェースの機能を兼ね備えたコンピュータシステムを模擬するものでなくてはならない。

クロスアセンブラの機能も含めた統合的なソフトウェアシステムにすれば、実行時でのソースプログラムの参照、修正時の1語単位のニューモニックコードから機械語への変換などより優れたシステムへの発展の可能性がある。この論文ではクロスアセンブラは既存のものを利用し、それの出力であるオブジェクトコードを読むことにより機械語プログラムをこのシステムに取り込むことにしている。Cコンパイラがクロスアセンブラと同じロードモジュールを発生すれば、C言語で書かれたプログラムの機械語レベルでの1語毎の進行の確認が可能である。

このシステムを起動すると後で述べるメインウィンドウが現れて、コマンドの入力を要求してくるので、それに応じながらアセンブリプログラムを走らせる。入力される機械語のプログラムの形式は、MotorolaのS-Record File Format[7]を仮定している。現在用意されているコマンドを表2に示す。このシステムの初期のバージョンでは、X Window環境を使わず、通常のターミナルを用いて実行することを前提にしてシステムを作製した。ターミナルから入力しやすいように、コマンドの全ては、1~3文字の小文字列と必要に応じてそれに続くパラメータから成る簡単な形式である。ヘルプ機能とシングルステップ実行はボタンをマウスでクリックすることによっても起動できる。

アセンブリ言語のプログラムの実行によってターミナルへの文字出力をする場合、その出力文字がシステムからのメッセージに埋もれて本来のプログラム出力が見難い。デバッグのためにプログラムをシングルステップで実行すると、1ステップ毎にレジスタの内容の変化を表示するために多くのメッセージが出力され、その傾向が顕著である。デバッグのためのメッセージを出力するための画面と、アセンブリプログラムの実行に伴って文字を入出力するための画面とを別々に設けることによってこの困難を避けることができる。その他、コマンドによっては、別の新たな画面で表示した方が操作性が良くなり、システムの性能向上が期待される。

X Window 環境では、必要に応じて自由に新たなウィンドウを開くこと、要らない時はそのウィンドウを消去すること、また表示されているウィンドウについては、ユーザの指示により、その大きさ、位置、文字の種類まで自由に変えることができる。また、一つのウィンドウの中に従属する複数

コマンド	動作
CR	シングルステップで次の命令を実行
.	すべてのレジスタ内容の表示
b	現在のブレークポイントの設定の表示
bl xxxx	プログラム番地 xxxx のブレークポイントの設定
bi mmm	命令語 mmm のブレークポイントの設定
bmr xxxx	メモリ番地 xxxx の読みとりのブレークポイントの設定
bmw xxxx	メモリ番地 xxxx の書き込みのブレークポイントの設定
bdr x	データレジスタ dx の読みとりのブレークポイントの設定
bdw x	データレジスタ dx の書き込みのブレークポイントの設定
bar x	アドレスレジスタ ax の読みとりのブレークポイントの設定
baw x	アドレスレジスタ ax の書き込みのブレークポイントの設定
cb xxxx	メモリ番地 xxxx からバイト単位でその内容修正
cw xxxx	メモリ番地 xxxx からワード単位でその内容修正
cl xxxx	メモリ番地 xxxx からロングワード単位でその内容修正
e または q	このシステムの終了
d	現在のプログラムカウンタのメモリ位置よりその内容表示
d xxxx	メモリ位置 xxxx からその内容表示
di	現在のプログラムカウンタのメモリ位置より逆アセンブル
di xxxx	メモリ位置 xxxx より逆アセンブル
g	現在のプログラムカウンタのアドレスから実行
g xxxx	アドレス xxxx から実行
int n	オートベクタ レベル n の割り込み動作の開始
k#	すべてのブレークポイントを解除
kl xxxx	プログラム番地 xxxx のブレークポイントの解除
ki mmm	命令語 mmm のブレークポイントの解除
kmr	メモリ読みとりのブレークポイントの解除
kmw	メモリ書き込みのブレークポイントの解除
kdr	データレジスタ読みとりのブレークポイントの解除
kdw	データレジスタ書き込みのブレークポイントの解除
kar	アドレスレジスタ読みとりのブレークポイントの解除
kaw	アドレスレジスタ書き込みのブレークポイントの解除
.ax yyyy	アドレスレジスタ ax の値を yyyy に設定
.dx yyyy	データレジスタ dx の値を yyyy に設定
.pc yyyy	プログラムカウンタの値を yyyy に設定
.usp yyyy	ユーザスタックポインタの値を yyyy に設定
.ssp yyyy	スーパバイザスタックポインタの値を yyyy に設定
.sr yyyy	ステータスレジスタの値を yyyy に設定
l s-file	ファイル名 s-file (S-format) をメモリにロードする
m ad1 ad2 n	アドレス ad1 からの n バイトを アドレス ad2 からのメモリへコピーする
mr	マシンリセット
r	すべてのレジスタ値をプログラムの走る前の値に復帰
ra	このシステムを起動した状態に復帰
re	このシステムのエラーをリセット
st on	システム設定のトラップ trap #0 を可能にする
st off	システム設定のトラップを不可能にする
t n	n ステップ毎のトレースモードの実行

表 2　コンマンドの一覧

のウィンドウを設定して別々にスクロールする画面を作ることもできる。このようなX Windowシステムを利用することにより、このシステムの要求に合ったユーザインターフェースを比較的容易に実現できる。

4 ウィンドウの種類と機能

本システムは、大別して5つのウィンドウから構成されている。通常は、各種レジスタの状態を表示したり、システムの状況を示すためのメッセージを表示するメインウィンドウのみが存在している。しかし、コマンドの入力や、メニューボタンをマウスでクリックすることで、必要なウィンドウが現れ(ポップアップ)、その機能に応じたの情報を示すことになる。それらのウィンドウとしては、メモリーの内容を16進数でダンプするメモリーウィンドウ、メモリの値をターミナルから直接変更するためのモディファイウィンドウ、メモリ上の内容を逆アセンブルしてニーモニックコードで表示するディアセンブルウィンドウ、コマンドの使用法を利用者に示すヘルプウィンドウがある。以下で、これらのそれぞれのウィンドウについて、その概要を述べる。

4.1 ヘルプウィンドウ

このウィンドウは、コマンドの使用法を表示するためのもので、コマンド(h)のキー入力、あるいはメインウィンドウでのヘルプボタンを押すことで現れる。ウィンドウ内には、図1で示されているように、アルファベット順に並べられたコマンドの使用法の一覧を示す領域と、その前ページ(UP)、または次ページ(DW)へ移るためのボタン、コマンドの終了とともにウィンドウを閉じるためのボタン(done)から構成されている。

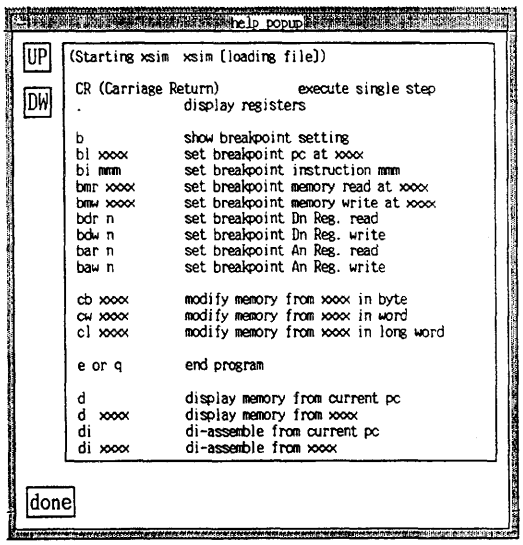

図1 ヘルプウィンドウの表示例

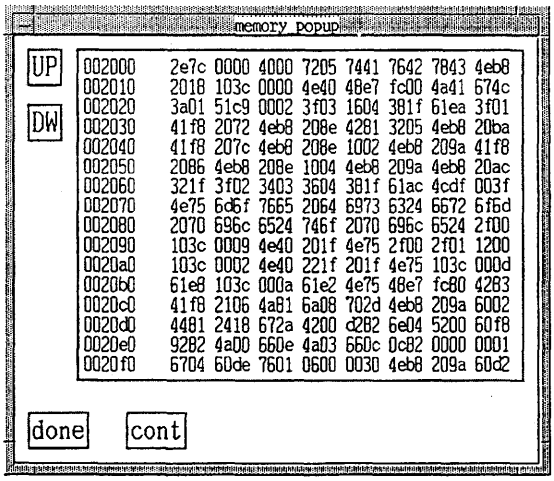

図2 メモリーウィンドウの表示例

4.2 メモリーウィンドウ

図2で示すように、メモリーから値を読み込み、単にその内容を16進数で表示するだけのものである。コンティニューボタン(CONT)を押すことにより、次々と指定されたアドレスから256バイト

ずつ表示する。ページ切替え用ボタン(UP, DW)で256バイトずつのスクロールが可能である。doneボタンを押すことにより、コマンドは終了し、このウィンドウは消える。

4.3 ディアセンブルウィンドウ

メモリーの内容を読み込み、それを逆アセンブルした結果をニューモニック記号で表示するためのウィンドウで、その表示領域と、逆アセンブルを次のメモリ位置に移動して続行するためのボタン(CONT)、他のウィンドウと同様にページ切替え用のボタンとコマンドを終了しウィンドウを閉じるためのボタン(done)から構成されている。図3にその表示例を示す。コマンドによりポップアップし、指定されたアドレスまたは現在のプログラムカウンタの位置から逆アセンブルする。

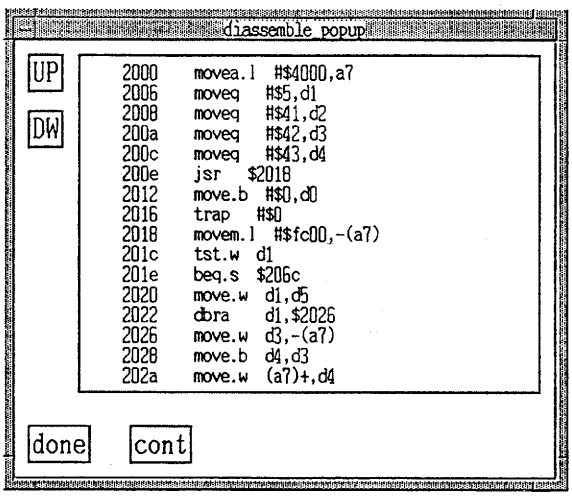

図3 ディアセンブルウィンドウの表示例

4.4 モディファイウィンドウ

メモリーから現在の内容を読み込んで表示し、その部分的な修正を催促するためのもので、アドレスと変更値の2つの入力領域と、メモリーの内容を16進数で表示する領域、コマンドの終了とともにウィンドウを閉じるためのボタン(done)から構成されている(図4参照)。コマンドの入力の際に開始アドレスは指定されるが、このコマンドの実行中にアドレス入力領域の内容を変えればそのアドレスに選択が移る。コマンドの開始によりこの画面がポップアップし、指定したアドレスの値については、選択されている位置を示すためにその内容表示部が白黒反転して表示される。入力領域に変更値を入力すると、リアルタイムに表示領域での値が更新され、入力領域と表示領域において次のアドレス値に自動的に選択が移る。

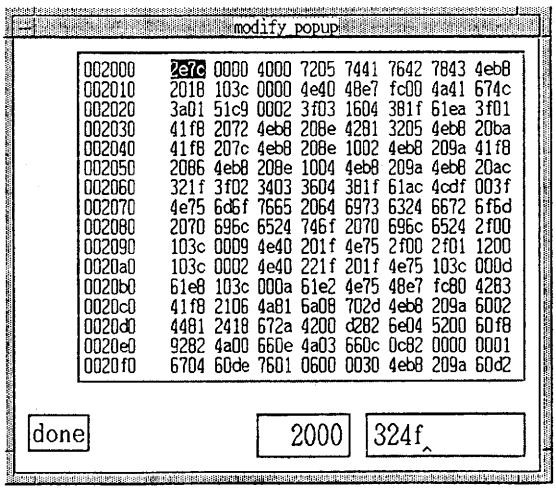

図4 モディファイウィンドウの表示例

4.5 メインウィンドウ

シュミレーターを起動した段階では、このウィンドウのみが現れる。通常の表示はこのウィンドウで行なわれる。

構成要素としては、各種レジスタの状態と表示直前に実行された命令と次に実行する予定の命令を

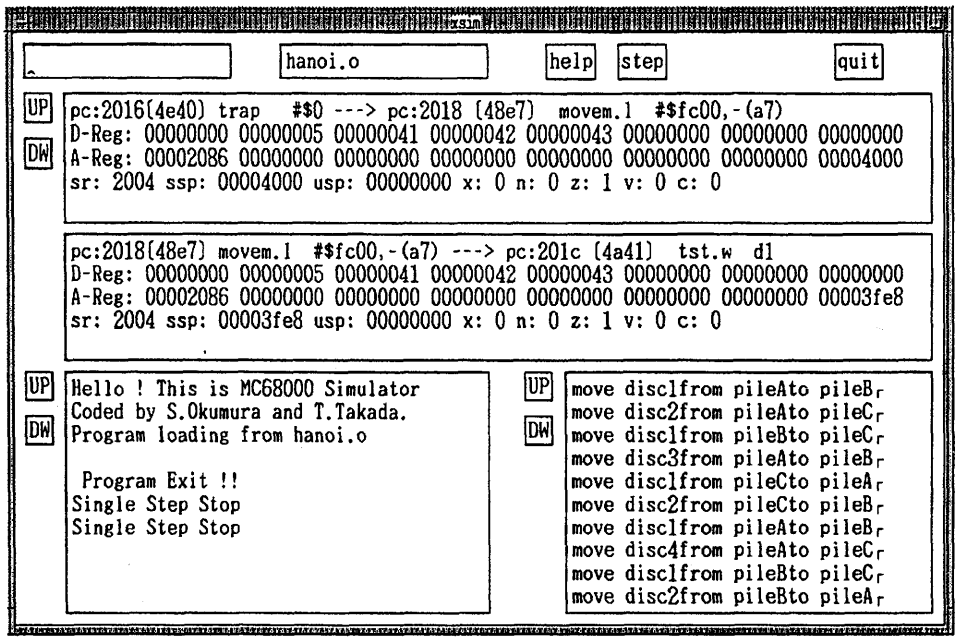

図5　メインウィンドウの表示例

表示する2つのレジスタ領域、種々のシステムからのメッセージを表示するためのメッセージ領域、アセンブリプログラムによって出力されるプログラム領域、コマンドを入力するためのコマンド領域、現在読み込まれているアセンブリプログラム名を表示するファイルネーム領域が有り、レジスタ領域、メッセージ領域、ランタイム領域には、過去に表示した内容を遡って見るためのページ切替え用ボタンが添えられている。また、ヘルプウィンドウを起動するためのヘルプボタンや、シングルステップを実行するためのステップボタン、このシステムを終了させるためのクイットボタンなどが用意されている。クイットボタンを押すことにより開かれていたウィンドウは全て消える。メインウィンドウの表示例を図5に示す。この例は、ハノイの塔として知られるパズルを解ための、再帰的コールを含むプログラムを走らせた例である。右下のウィンドウにプログラムからの出力がスクロールされ、左下がシステムからのメッセージ出力である。

5　ウィンドウアプリケーションへの機能の修正・追加手続き

本システムは、最初に通常のキャラクタディスプレイ用として開発されたものをもとに、その部分的な修正と、ウィンドウアプリケーションへの機能発展を行なうことで実現している。

以下では、このような修正・追加における各種の手続きの内容について述べる。

5.1　ウィンドウプログラミングにおける流れ

グラフィカル・ユーザー・インターフェイス(GUI)を実現するために、本論文ではOSF/MOTIF版のX Toolkit[6]と呼ばれるオブジェクト指向的な形態のプログラミングを可能にするライブラリを用いる。このライブラリを使用することにより、利用者と対話的なやりとりの処理は、決められた機能を果たす部品を用意し、サーバから送られる種々のイベントによって選択された手続きを起動する

ことによって実現される。具体的には、以下のようなステップを踏む。

1. Intrinsicsといわれる Toolkit の関数やデータ構造を初期化する。Xサーバとの接続を確立し、リソースとよばれる必要な変数としてのデータ構造体の領域を確保する。

2. ウィジェットとよばれるGUIを構成するオブジェクトとしての部品を生成して、選ばれたウィジェットを使用可能なように初期化する。本システムで使用するウィンドウや種々の表示領域、ボタンなどのウィジェットを生成する。

3. コールバックと呼ばれるそのイベントが発生した時に呼ばれる関数をそれぞれのウィジェットに登録する。このシステムでは、前章で述べたhelp、step、UP、DW、quitなどのボタンに適用している。クイットボタンについては、実行されれば全体のプログラムが終了するような関数を作成し、登録しておく。

4. ウィジェットが使用するウィンドウを生成し、ウィジェットの最終的な初期化を行なう。

5. 利用者が生成するイベントを待ち続けるループに入る。このイベントループは、イベントを受けとるとそのイベントを発生したウィジェットに登録されているコールバックに制御を移す。

これらの手続きでは、X Toolkit の関数を適切な引数を与えて呼ぶことにより、それぞれの処理を行なっている。

5.2 書式付出力関数の作成

本システムは、アセンブラシミュレーターである性格上、文字列の出力が頻繁に行なわれる。通常のCプログラムにおいての文字列の標準出力のためには、たとえばprintf()等の関数が用いられる。このウィンドウシステムでは、文字列出力をウィジェットに対して行なわなければならない。

そのために、Toolkit に用意されているテキストウィジェットを使うことにより実現する。このウィジェットは、ユーザーが1行あるいは複数行のテキストを編集するエリアを提供するもので、テキストエディタのように、文字列の挿入、削除などの処理が行なえる。また、編集は、様々な関数によって行なうことも可能であり、このことを利用して、コマンドの入力、メッセージの出力、レジスタの値の出力などを行なっている。

書式付の出力関数を作成するに当たっては、Cライブラリの関数fprintf()のような形式で、出力先、書式、引数を与えるものとする。fprintf()での出力先は、ファイルポインタであるが、これをテキストウィジェットに置き換え、出力テキストについては、fprintf()と同じ書式で出力できるようにしている。具体的には、varargs.h に定義されているユーティリティマクロと、vsprintf()を併用することで、テキストにおける複数の引数を扱っている。まず、出力先となるウィジェットをこの関数の引数文字列からva_arg()により抜きだし、テキストウィジェットであることを確認する。次に使用するフォーマット指定を含む文字列を抜き出す。さらにその文字列をvsprintfを使って最終的な出力文字列に変更する。得られた文字列を、テキストウィジェット用の関数XmTextReplace()を用いて渡す。

本来XmTextReplace()という関数は、指定範囲の文字列を別の文字列と交換するためのものであるが、指定範囲をウィジェット内の文字列の最後尾とすることで、文字列の追加を行なっている。

また、ウィンドウ上で新たに挿入されたテキストを表示しようとする機能が働くので、挿入部分の位置の選択によりスクロールをコントロールすることができる。挿入部分を現在表示されている部分に設定すればスクロールが不必要になり、出力速度を早めることが可能である。

5.3 コマンド入力の受けとり

本システムでは、デバッグのためのいろいろな機能を実行するために、コマンドによる命令の受付を行なっている。このコマンド受付のための領域も先のテキストウィジェットを用いているが、ここではシングルラインのモードで扱っている。シングルラインの場合、デフォルトではリターンキーによる改行の処理は行なわれない。そこで、リターンキーを押した場合に、コマンド入力を受け付ける関数をそのテキストウィジェットにトランスレーションとして登録しておく。その関数の中では、テキストウィジェットから文字列を受けとり、そのコマンドに応じた処理を行なった後、テキストウィジェット内の文字列を空にする。したがって、文字列の読み込みを、常に1行のコマンド領域で実行する。

5.4 ポップアップウィジェットの操作

先に述べたヘルプウィンドウやメモリウィンドウのようなウィンドウは、ポップアップウィジェットを利用する。これらのウィジェットはアプリケーション起動時に生成されるが、親のウィジェットの幾何学的管理下に置く(マネージ)ためのToolkitの関数を呼び出す時に、実際のウィンドウが表示される。

また、あるポップアップウィンドウが表示されている状態でも、他のウィンドウは独立に操作できる。例えば、ヘルプウィンドウを見ながらメインウィンドウでコマンドを実行することなども可能である。ここで述べたウインドウを消去せずにウインドウを開いた状態を図6に示す。ウィンドウを開いた状態で次々とウィンドウを変え、それぞれのコマンドの続きを実行できることが、X Windowのアプリケーションに発展させたことの大きな長所である。

6 コマンドの機能における教育的な配慮

既に表2によってこのシステムにおける全てのコマンドの機能が示されている。この章では、本システムの特徴として工夫されたコマンドの機能の操作性の改善と教育的配慮について述べる。

計算機の一連の動作がどのような基本的動作ステップで構成されていくかを実際に体得することが、初歩的なレベルでのアセンブリ言語の教育の一つの目標であると「まえがき」の章で述べた。これを最も的確に実現されるのがシングルステップモードでのプログラムの実行である。メインウィンドウでステップボタンにマウスカーソルを置き、クリックすることにより、次々とシングルステップでプログラムが進む。ステップ毎に、16個の汎用レジスタの内容、ステータスレジスタ、スタックポインタ、コンディションコードとプログラムカウンタおよび実行された命令語が表示される。このための2つの状態表示画面は、2つ合わせてスクロール画面となり、新たな表示が古いものを押し上げることになる。前の状態を見たい時は、ページ切替えボタン(UP, DW)を選択して押すことにより、自由に表示を戻すことができる。これらの機能は、サブルーチンの呼びだしにおけるスタックの変化および引数の受け渡しの機構、各命令におけるコンディションコードの変化と条件ジャンプの対応、C言語における自動変数の割り当て方法、などの機械語機構の原理的な理解の助けとして有効であ

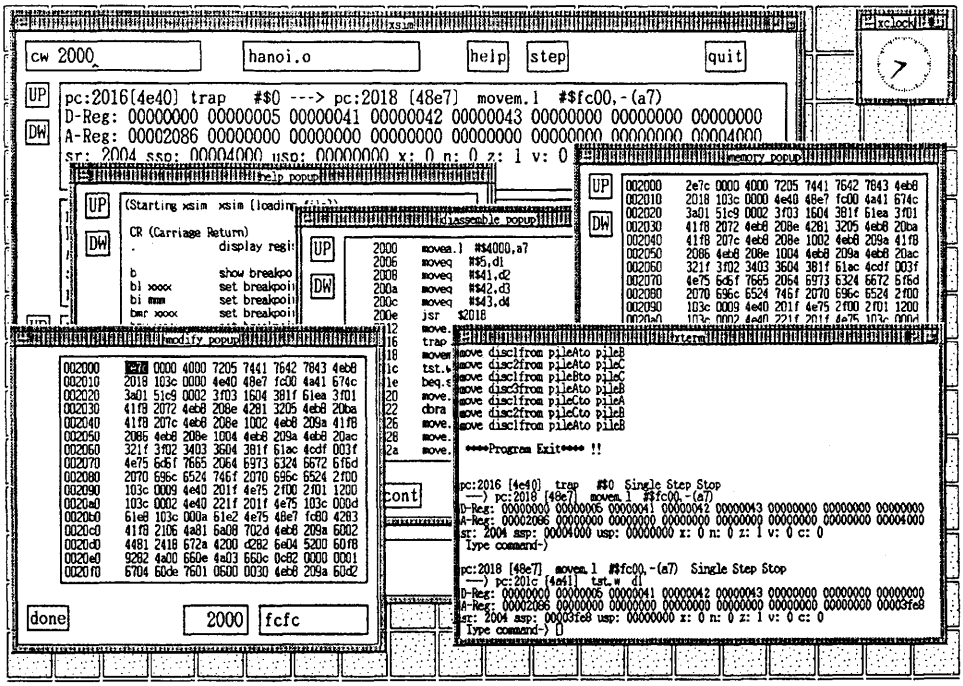

図 6　実際の使用中におけるウィンドウの表示例

る。なお、シングルステップではプログラムの進行が遅いときは、任意の数のステップ毎のトレースの設定も可能である。

また、デバッグの機能を高めるために、ブレイクポイントに関する機能を多く持たせている。通常のプログラムカウンタ、すなわち実行アドレスによるブレイク条件の設定の他に、ニューモニックで利用者が指定した命令語の実行、指定された、データレジスタ、アドレスレジスタおよびメモリへの書き込み、または読み込みなどに対して、それぞれ論理和的にブレイクポイントの設定が可能である。それぞれの種類に対して、複数のブレイクポイントの設定が可能である他に、設定されたものに対しては、その中から選択的にブレイクポイントから外すことが可能である。以上の機能は、プログラムの実行が全てソフトウェア上でなされていることを利用して、比較的容易に実現されてる。

マシンリセット(ms)では、実際CPUが行なうリセットと同じように、絶対アドレス0番地からスーパバイザスタックポインタをロードし、4番地からプログラムカウンタをセットする。その他リセット(r)は、ロードしたプログラムはそのままで、プログラムを走らす前の状態を復帰する。また別のリセット(ra)は、このソフトウェアを起動した直後の状態に戻すリセットであり、これにより機械語のプログラムを最初から繰り返し走らせたり、別のプログラムをロードして実行することなどを容易にしている。割り込みについては、オートベクタをfetchして、スーパバイザモードにセットされるまでを擬似動作させている。

7 おわりに

このシステムは福井大学情報工学科のアセンブリ言語の講義と演習を行なうために開発されたものである。当学科では1991年3月に教育用計算機として、1台のミニスーパーコンピュータ、14台のRISC-CPUを備えたワークステーション群と100台のX Windowターミナル等がネットワークで結ばれたシステムに置き換えられた。計算機が変わってもクロスアセンブラのみ移植すれば、このシステムの使用上の問題はない。通常のキャラクタ端末を使用するこの研究の前のシステムも、この新しい計算機システム上で走るが、高度なX Windowの環境を有効に利用したいという考えがこの論文の一つの動機になっている。このシステムはアセンブリ言語の初心者を意識して、教育を目標として開発したが、教育用と限定する理由は特にはなく、アセンブリ言語による一般のプログラム開発用のソフトウェアシステムとして十分使用できるものである。

以前は68000のシングルボードコンピュータを使用してアセンブリ言語の実習をしていたこともあったが、このシステムを利用することにより、そのような実験装置を教室に持ち込む手間をかけずに、使い易い環境で、ほとんど同じような機能の実習が可能となっている。ただし、このシステムの使用で実現できない計算機間の接続やディスクの入出力など、直接ハードウェアの制御に関わることを扱う実習については別に機会を設けなければならない。

更に機能を充実して、決められた問題について質疑応答をするようなCAIシステムとしてより教育的に有効なものに発展させることが望まれる。

参考文献

[1] 奥村彰二, 柳瀬龍郎. 平成2年度電気関係学会北陸支部連合大会講演論文集, B-103, pp. 243-244, 1990.

[2] Motorola Inc. *M68000 8-/16-/32-Bit Microprocessors Programmer's Reference Manual*, Prentice-Hall, Inc, 1986.

[3] John F. Wakerly. *Microcomputer Architecture and Programming*, John Wiley & Sons, Inc.1989.

[4] Adrian Nye and Tim O'Reilly. *X Toolkit Intrinsics Programming Manual*, O'Reilly & Associates, Inc. 1990.

[5] Tim O'Reilly(ed.). *X Toolkit Intrinsics Reference Manual*, O'Reilly & Associates, Inc. 1990.

[6] Douglas A. Young. *The X Window System: Programming and Applications with Xt OSF/Motif Edition*, Prentice-Hall, Inc. 1990.

[7] Motorola Inc. *M68000 Family Linkage Editor User's Manual*, Motorola, Inc, 1985.