Xウィンドウシステム上で動く アセンブリー言語教育システムの試作

（平成8年11月28日　原稿受付）

電気工学科　石井幸光
電気工学科　重松保弘

Development of an assembly language education system for X Window System

by Yukimitsu Ishii
Yasuhiro Shigematsu

Abstract

It is very important to learn assembly language for understanding the function of computers. There are two ways to learn assembly language, that is, using real computers and using virtual computers. For beginners, the latter is suitable. Thus, we developed an assembly language education system which contains a simulator of the virtual computer COMET and its assembler CASL. We implemented the simulator, XCOMET, using the X Window System of UNIX. As a result, remarkable improvement of user interface was obtained. In this paper we present the characteristics and functions of XCOMET system in detail. Problems which appear in using X Toolkit of X Window System are also discussed.

1．まえがき

コンピューターを学ぶ過程で，アセンブリー言語（assembly language）を避けて通ることはできない。アセンブリー言語の命令はCPUの機械語命令に1対1で対応しているため，アセンブリー言語を理解することは機械語を理解することになり，さらにはコンピューターの動作原理について理解を深めることにもなる。

アセンブリー言語の教育システムとしてまず考えられるのは，実在するコンピューターを教材として使うことである。しかし，実用化されているコンピューターは機械語の命令数も機能も多いため，初心者にとって学習することは容易ではない。これに対し，機能を制限した仮想のコンピューターを教材に用いる方法も考えられる。この場合には学生のレベルに合わせた教材を用意することができるため，入門教育に適した方法と言える。仮想コンピューターでの学習は即戦力につながらないとの考え方もあるが，実在するコンピューターの学習でもアーキテクチャーの異なるコンピューターについてはやはり即戦力に結びつかないという問題が生じる。現在，学部2年生を対象とした本学の講義（計算機言語・同演習）では，アセンブリー言語を初めて学ぶ受講生が大半であるという事情を考慮し，後者の方法を採用している。

昭和62年に情報処理技術者試験の出題用として仮想コンピューターCOMETおよびアセンブリー言語CASLの仕様が定められたこともあり，前述の講義では昭和63年度から仮想コンピューターCOMETとそのアセンブリー言語CASLを教材に用いることにした。

講義でプログラミングの実習を行なうには，COMETの動作をシミュレートするソフトウェアが必要である。そこで筆者らの研究室ではパソコン端末上で動作するCASLアセンブラーとCOMETシミュレーターを昭和62年度に開発した[5]。これらはC言語で書かれており，テキスト画面で動作する簡単なシステムである。平成4年度に情報科学センターの教育環境がUNIXワークステーションとXウィンドウシステムに統一された後も，基本的に同じ教育システムを使い続けている。

平成6年度の仕様改訂で，COMETに割り込みやポ

ート入出力機能などが追加された。一方，CASLには再配置できるオブジェクトの出力や実行開始番地指定などの機能が加わった。またリンカー(linker)CASLLINKの仕様も新たに定められた。筆者らの研究室では，この仕様拡張に対応してCASLアセンブラーとCOMETシミュレーターにこれらの機能を付け加え，さらにCASLLINKを新たに開発した[2][3]。

現在使用中のCOMETシミュレーター（以下では「従来版」と呼ぶ）は10年のあいだ講義に役立っているが，問題点もある。まず画面が見にくいことである。従来版の表示例を図1に示す（左端にある数字つき矢印は説明用，テキスト内の太字はキーボードから入力した文字である)。従来版はキャラクターベースで，しかも最下行に次々と情報が出るタイプライター方式の表示法である。シミュレーターのプロンプトやレジスターの値などが入り交じって画面に出るため，初心者には判りにくいと思われる。

また，ソースリストとの対応が分からないことも問題である。図1にはプログラムカウンター(PC)の値は表示されているが，アセンブル前のソースリストは表示されていない。ソースリストとプログラムカウンターの対応は，アセンブル時に出来るリストファイルを見て知るしかない。

こうした問題点を解決するためには，GUI環境が整備されているウィンドウシステム上にアセンブリー言語教育システムを構築することが望ましい。現在，Windows95などのパソコンOSをベースにしたCASL・COMETシステムはいくつか存在するが，UNIXのXウィンドウ上で動作するシステムは筆者らの知る限り開発されていない。

そこで筆者らは，Xウィンドウ上で動作するCASL・COMETシステムについて検討し，実際にCOMETシミュレーター“XCOMET”を試作した。本論文ではXCOMETの特徴を従来版と比較しながら述べる。また，アプリケーション開発システムとしてのXツールキットを評価する。

2．Xウィンドウと Xツールキット

Xウィンドウシステムとは，機種の異なるワークステーション上で同じ環境を実現するためにマサチューセッツ工科大学（MIT）で開発されたウィンドウシステムである。事実上UNIX用ウィンドウシステムの標

```
% comet ex1.cmt td↵
COMET simulator Ver1.24

BREAK:   PC    IR    GR0   GR1   GR2   GR3  GR4(SP)  FR   step:     0
BREAK:  0000  9000  0000  0000  0000  0000   2710     0

Command ? t↵
TRACE:   PC    IR    GR0   GR1   GR2   GR3  GR4(SP)  FR   step:     0
TRACE:  0000  9000  0000  0000  0000  0000   2710     0
1 ➡ ?s↵
TRACE:  0003  1010  0000  0000  0000  0000   2710     0
TRACE:  0005  4110  0000  0053  0000  0000   2710     0
TRACE:  0007  6100  0000  0053  0000  0000   2710     0
TRACE:  0009  1010  0000  0053  0000  0000   2710     0
TRACE:  000B  4110  0000  0053  0000  0000   2710     0
TRACE:  000D  6000  0000  0053  0000  0000   2710     0
TRACE:  0014  9100  0000  0053  0000  0000   2710     0
2 ➡ NON DIGIT
TRACE:  0017  6400  0000  0053  0000  0000   2710     0
TRACE:  0019  9200  0000  0053  0000  0000   2710     0

total step :     9

  ----------------    memory  dump   ------------------
  ADDR    0/8   1/9   2/A   3/B   4/C   5/D   6/E   7/F   --------
  0000   9000  001A  006A  1010  001A  4110  006B  6100  ..j...k.
  0008   0014  1010  001A  4110  006C  6000  0014  9100  ....l...
  0010   006D  0072  6400  0019  9100  0073  007C  6400  mr...s|.
  0018   0019  9200  0053  0000  0000  0000  0000  0000  ..S.....
  0020   0000  0000  0000  0000  0000  0000  0000  0000  ........
  0028   0000  0000  0000  0000  0000  0000  0000  0000  ........
  0030   0000  0000  0001  0030  003A  0044  0049  0047  ...0:DIG
  0070   0049  0054  0005  004E  004F  004E  0020  0044  IT.NON D
  0078   0049  0047  0049  0054  0009                     IGIT.
%
```

図1　従来版COMETの出力例

準となっている。

Xウィンドウシステム用に作られたプログラムは機種に依存しない。どんな機種でも，ソースファイルをコンパイルすれば同じ機能のアプリケーションを作り出すことができる。従って，Xウィンドウシステム用のアプリケーションソフトウェアはソースファイルの形で配布される。

Xウィンドウシステム上で動作するアプリケーションソフトウェアを容易に開発するためのライブラリーとして「Xツールキット（X Toolkit)」が用意されている。これはユーザーインターフェイスとなる押しボタンやメニューなどの部品を組み合わせ，さらに各部品を操作したときに起こる動作を指定することでアプリケーションソフトウェアを作成できるようにしたものである。

Xツールキットの場合，部品の組み合わせ方や部品の動作の仕方はC言語の関数で記述する。厳密に言うと，「Xツールキット」とはそれらの関数ライブラリーだけを指す言葉である。実際に組み合わされる部品群は「ウイジェットセット（widget set)」と呼ばれる。ウイジェットセットには，MITがサンプルとして公開しているアテナウイジェットセット（Athena Widget Set)，Open Software Foundationが推奨したMotifウイジェットセット，UNIXインターナショナルが提唱したOPENLOOKウイジェットセットなどがある。XCOMETの開発は，最も基本的なセットであるアテナウイジェットセットをXツールキットで組み合わせることにより行なった。

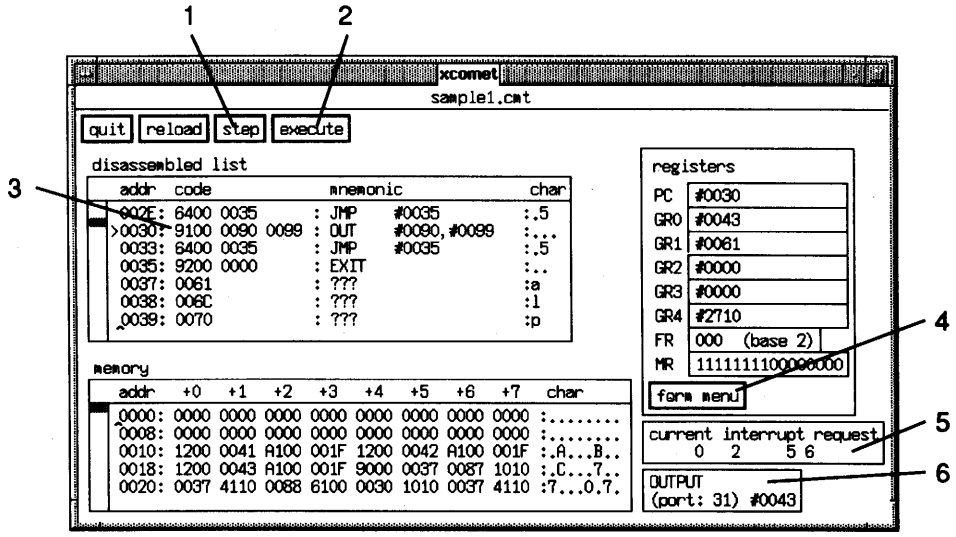
3．XCOMET

3.1　XCOMETの画面構成

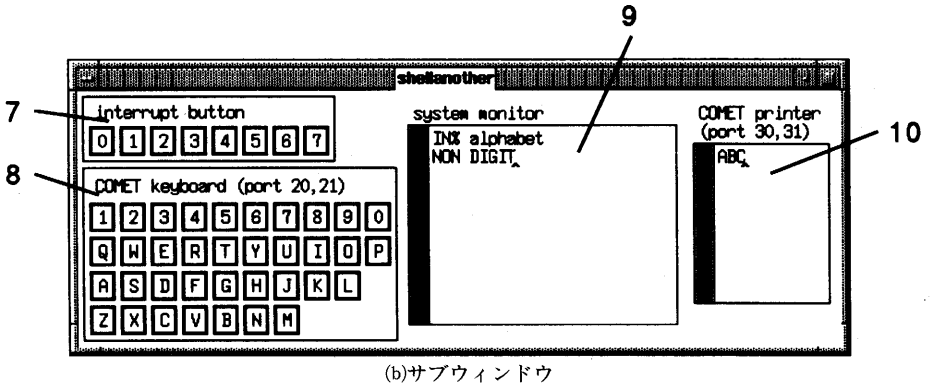
XCOMETは図2の通り2つのウィンドウから成る。(a)はCOMET本体，(b)は周辺機器である。実際のコンピューターに近い形にするため2つに分けた。図中の番号は以降の説明用に付けたものである。

3.2　逆アセンブルリスト

従来版のシミュレーション画面にはソースリストは表示されず，プログラムカウンターの値とアセンブラ



(a)メインウィンドウ



(b)サブウィンドウ

図2　XCOMETのウィンドウレイアウト

ーが出力するリストファイルを基に現在の実行位置を調べていた。このような間接的な方法でプログラムの動作を追うのは，初心者でなくとも骨の折れる作業である。ソースリストを画面に表示させ，機械語を直接見ることなく動作を調べてゆけるようにする必要がある。

ところが，XCOMETが読み込むのは変換後の機械語コードだけであるため，ソースリストを表示することはできない。そこで，逆アセンブルしたものを表示することにした(図2の3)。さらにプログラムカウンターが指している番地には“>”の印が付く。コメントやラベルなどは消えているが，プログラムのどのあたりを実行しているのかはある程度わかる。

3.3 レジスターの表示法

レジスターに入っているデータを画面に表示するときは16進数を使用することが多い（従来版ではどのレジスターの値も16進数で表示していた）が，それだけでは不便である。算術演算なら10進数，シフト演算なら2進数が初心者には分かりやすいと考えられる。文字列を扱う時はASCIIコードに対応する文字を出すようにすると処理の様子が見やすくなると思われる。

そこで，XCOMETにはレジスターの値を様々な形式で表示する機能を付けた。図2の4がその選択用ボタンである。文字，16進数，符号つき10進数，符号なし10進数，2進数の5種類を，メッセージウィンドウが画面に出ていない時ならいつでも切り替えることができる。ただしフラグレジスター（FR）とマスクレジスター（MR）の値は常に2進数で表示することにした。

3.4 機械語プログラムの実行

XCOMETでCOMETの機械語プログラムを実行するには，1命令実行ボタン（図2の1）または連続実行ボタン（図2の2）を使う（従来版には1命令実行ボタンはなかった)。1命令実行ボタンを1回クリックすると1命令だけ実行される。連続実行ボタンはプログラム終了まで自動的に実行するためのものである。なお，ソースリストのうち1命令実行ボタンの骨格となる部分を付録に示した。

3.5 文字列の入出力

3.5.1 文字列の出力

従来版はウィンドウシステムを使用していないため，マクロ命令OUTによる出力文字列とCOMETの状態を伝えるメッセージを混在させて表示せざるを得なかった（図1では2の行がマクロ命令OUTによる表示であり，その他はシミュレーター自身の状態出力である)。これに対しXCOMETではシステムモニターウィンドウ(図2の9)を設け，マクロ命令OUTによる出力を1つのウィンドウ中にまとめて表示するようにした。COMETの状態は別ウィンドウ（主に図2(a)内の各ウィンドウ）にリアルタイム的に表示する。

3.5.2 文字列の入力

従来版では，COMETがマクロ命令INを実行すると画面にプロンプト“？”が表示され，文字列の入力を促す（図1では1の行)。

XCOMETの場合は画面上に文字列入力用ウィンドウが新たに現れる。そこでデータとなる文字列をキーボードから入力してリターンキーを押すと，文字列はマクロ命令INによって読み取られ，入力用ウィンドウは消滅する。この方式では，従来版と違って入力した文字列が画面から消えてしまうため，入力された文字列をシステムモニターウィンドウ（図2の9）にプロンプト“IN %”付きで表示しておくことにした。

4．拡張機能の実装

XCOMETは拡張仕様の一部に対応している。ただし，拡張仕様は割り込みやI/OなどCOMET外部からの入力を必要とするものが多いため，判りやすい形で完全にシミュレートするのは難しい。XCOMETではウィンドウシステムを生かして自然な形でのシミュレートを試みた。

拡張機能をテキスト画面上に実装したもの（以下では「拡張版」と呼ぶ）が既に存在する［2］［3]。この章では拡張版とXCOMETの比較を行ないながらXCOMETの特徴を述べる。

4.1 割り込み

拡張版でCOMETに割り込みを要求するには，CTRLキーを押しながらZキーを押し，更にリターンキーを押す。何番の割り込みを要求するのかを尋ねてくるので，0～7の数字を入力してリターンキーを押す。以上の操作で割り込み要求がCOMET内に保持される。実際に割り込みが起こると，何番の割り込みが発生したのかを知らせるメッセージが画面に表示され，割り込み時の処理が始まる。

XCOMETでは，割り込み要求ボタン（図2の7）をクリックするだけでCOMETに割り込みを要求できる。COMETが保持している割り込み要求はカレント割り込み要求ウィンドウ（図2の5）に表示される。実際に割り込みが起こると，何番の割り込みが発生したのかを知らせるメッセージウィンドウが新たに画面上に出現する。メッセージウィンドウ内にあるOKの

ボタンをクリックするとウィンドウが消え，割り込み時の処理が始まる。

4.2 I/O

4.2.1 ポート入力

拡張版でINPUT命令を実行すると1語分のデータ入力を促すプロンプトが画面に出るので，入力したいデータをキーボードから16進数で入力する。

XCOMETでは，マクロ命令INと同様に入力用ウィンドウを開いて，1語（16ビット）長のデータを16進数で入力させるようにした。また，入力した値はシステムモニターウィンドウ中に残すことにした。

4.2.2 ポート出力

OUTPUT命令によるポート出力については拡張版もXCOMETも大差はない。XCOMETでは図2の6の形式でOUTPUT命令の実行結果を表示する。同じものを拡張版では

PORT(1F) OUT＝0043

と表示する（10進数の31は16進数では1F)。

4.2.3 仮想キーボードと仮想プリンター

ポート入出力の例題プログラムを作成するとき，実在する外部装置に似せた仮想装置を用意しておけば学習に有益であると思われる。そこで，XCOMETシステム上に仮想キーボードと仮想プリンターを置くことにした。以後，これらをCOMETキーボード，COMETプリンターと呼ぶことにする。

COMETキーボード（図2の8）には予め2つのポート（20番と21番）を割り当てている。20番のポートはキーボードの状態を示すステータスポートであり，21番のポートは押されたキーを読み取るためのデータポートである。通常のキーボードと同様，COMETキーボードのキーをどれか1つクリックすると，そのキーに相当するASCIIコードが1語長のバッファーに保持される。利用者はステータスポートからステータスワードを読み込み，その最下位ビットが1 (buffer full)であることを確認したうえで，バッファー内のデータをデータポートから読み込む。buffer fullでない時にバッファーの内容を読み込んでも，そのデータの内容は保証されない。

なお，バッファーの容量は1文字分しかないため，バッファー内にデータが存在する時にCOMETキーボードのキーをクリックしても無視される。

COMETプリンター（図2の10）にも予め2つのポート（30番と31番）を割り当てている。30番はステータスポート，31番はデータポートである。利用者はステータスポートからステータスワードを読み込み，その最下位ビットが1 (ready)であることを確認したうえで，データポートにデータを出力する。COMETプリンターはそのデータワードの下位8ビットをASCIIコードと見なし，COMETプリンターウィンドウに文字として表示する。readyでない時に出力したデータは無視される。

なお，CRコード1文字で復帰改行を行なうようにした。

5．あとがき

本研究ではXウィンドウ上で動作するアセンブリー言語教育システムXCOMETを試作した。XCOMETはGUI環境を取り入れることで従来版よりも画面を見やすくし，操作性を向上させることができた。また，逆アセンブルリストを常に表示することによって実行中の命令の位置を確認できるようになった。仮想キーボードや仮想プリンターを設けることで外部装置との入出力をシミュレートすることが可能になった。

本システムの開発に用いたXツールキットは，Xウィンドウシステム用のソフトウェアを容易に作成できるよう開発されたライブラリーである。しかし，(1)プログラマーには「イベント駆動」の概念を十分に理解していることが要求される，(2)Xツールキットによるプログラミングの参考資料が極めて少ない，(3)Windows95などと比較した場合，開発環境が整備されていない，(4)元からある部品を組み合わせて作るのは非常に楽であるが，それ以外のこと（グラフィックの描画など）をしようとすると途端にプログラミングの作業量が増える，(5)最近まで日本語に正式対応していなかった，などの問題点があり，プログラミングに予想外の労力を費やすこととなった。そのため，この論文で述べたXCOMETの機能のうち連続実行機能は未だ正常に動作していない。

今後は，ソースリストやスタックの表示，アニメーションを用いた教育システムへの発展，ヘルプ機能の付加などの点について，さらに改良を行なう予定である。

参考文献

[1] 石井幸光『ウィンドウ環境を利用した計算機教育システムの研究——X Window版COMETシミュレーターの制作——』卒業論文，九州工業大学工学部電気工学科，1996

[2] 今林英一『アセンブリ言語教育システムに関する研究——COMET/CASLの機能拡張——』卒業論文，九州

工業大学工学部電気工学科，1995
[3] 小川浩司『アセンブリ言語教育システムに関する研究——システムの機能拡張——』卒業論文，九州工業大学工学部電気工学科，1995
[4] 財団法人日本情報処理開発協会情報処理技術者試験センター『平成7年度秋期情報処理技術者試験案内書・願書』
[5] 重松保弘『CASLによるアセンブリ言語入門』啓学出版，1988
[6] 山口和紀監修『The UNIX Super Text（上・下）』技術評論社，1992
[7] 山本稔『COMET&CASLアセンブラ・シミュレーション・プログラム』フリーソフトウェア，1995

付録　XCOMETのソースリスト

（1命令実行ボタン関連の部分のみ抜粋）

```
//----------------------------------------
//                XCOMET
//----------------------------------------
#include <stdio.h>
//----------------------------------------
// 必要なファイルをインクルードする
//----------------------------------------
// Xツールキットを使うためのファイル
#include <X11/StringDefs.h>
#include <X11/Intrinsic.h>
// アテナウイジェットを使うためのファイル
#include <X11/Shell.h>
#include <X11/Xaw/Form.h>
#include <X11/Xaw/Command.h>
#include <X11/Xaw/Dialog.h>
//-------- 以 下 略 --------

// 各部品のための変数を作る
Widget toplevel,
       commandstep,
       commandexecall,
//-------- 以 下 略 --------

//=======================================
// 各部品の属性を決める
//=======================================
String fallback_resources[] = {
//------------- 中 略 -------------
    //---------------------------------------
    // 1命令実行ボタン
    // （部品名commandstep）の属性
    // label……ボタンの表面に描く文字列
    // fromHoriz……このボタンを何の右隣に出すか
    //               （ここではreloadボタン）
    // top,bottom……ウインドウの大きさが
    //               変わった時に
    //               ボタンの上辺と下辺は
    //               どう動くか
    //---------------------------------------
    "*commandstep.label:      step",
    "*commandstep.fromHoriz:  commandreload",
    "*commandstep.top:        ChainTop",
    "*commandstep.bottom:     ChainTop",
//------------- 中 略 -------------
};

//----------------- 中 略 -----------------

//=======================================
// 1命令実行ボタンをクリックすると
// この関数が呼び出される。
// 仮引数は必ずこの形にしなければならない。
//=======================================
void ExecuteStep(w, client_data, call_data)
Widget     w;
XtPointer  client_data;
XtPointer  call_data;
{
//----------------- 中 略 -----------------
    //---------------------
    // 命令を解読する
    //---------------------
    // 命令の1語目を読み込む
    ir = codetab[pc];
    // 以下で次の値を調べる
    // opcode……どの命令か
    // grn……汎用レジスターの番号
    // xrn……指標レジスターの番号
    opcode = (ir >> 8) & 0x00ff;
    grn = (ir >> 4) & 0x000f;
    xrn = ir & 0x000f;
//------------ 中 略 ------------
    // 有効アドレスを求める
    if (xrn == 0)
        ea = codetab[pc+1];
    else
        ea = codetab[pc+1]+gr[xrn];
    //----------------------
    // 命令ごとの動作を行なう
    //----------------------
    switch(opcode) {
        case 0_LD : gr[grn] = codetab[ea];
                    pc = pc + 2;
                    break;
        case 0_ST : writememory(ea, gr[grn]);
                    pc = pc + 2;
                    break;
    //-------------- 以 下 略 --------------
    }
}

//=======================================
// メイン部分。ここで画面を作る
//=======================================
main(argc,argv)
int argc;
char **argv;
{

//----------------- 中 略 -----------------

    // 1命令実行ボタンを作る。各引数の意味は
    // "commandstep"……部品の名前
    // commandWidgetClass……作る物の種類。
    //                       ボタンのこと
    // formall……親となる部品の名前。
    //           フォームとは
    //           部品を取り付けるシートのこと。
    //           どのフォームに付けるかを書く。
    // NULL……引数群の終わりを表わす
    commandstep = XtVaCreateManagedWidget
                    ("commandstep",
                     commandWidgetClass,
                     formall,
                     NULL);

    // 部品commandstepでイベントが起こったら
    //  （すなわち1命令実行ボタンが押されたら）、
    // 関数ExecuteStepを呼び出すよう指定する
    XtAddCallback(commandstep, XtNcallback,
                  ExecuteStep, NULL);

//----------------- 中 略 -----------------

    // 部品を並べて作ったウインドウを実際に表示する
    XtRealizeWidget(toplevel);

//----------------- 中 略 -----------------

    // イベントが起こるのを待つためのループ
    XtAppMainLoop(app_con);
}
```