Java 言語を用いた KUE-CHIP2 シミュレータの開発

山村 周史　新實 治男

京都工芸繊維大学 工芸学部 電子情報工学科

1997 年 1 月 8 日

1　はじめに

KUE-CHIP2 (Kyoto University Education Chip 2) は, 大学などでの計算機教育を目的として開発された 8 ビットマイクロプロセッサであり, これを装着した「KUE-CHIP2 教育用ボード」(以下では **KC2 ボード**と略記する）により, その内部動作を観察できるようになっている. 本学科でも, 3 回生の学生実験の一課題として, この **KC2 ボード**を利用しており, 多大な教育効果を挙げてきている.

しかし, 数年間に及ぶ利用経験を通して, 我々のところのような, 初学者を対象とする実験でこの **KC2 ボード**を利用する場合には, 幾つかの問題点があることも, 徐々に明らかになってきた. そこで我々は, この **KC2 ボード**と同等（もしくはそれ以上）の機能もつシミュレータを Java 言語 [1] を用いて開発することにより, これらの問題点の解消を図った.

以下第 2章では, **KC2 ボード**に対して我々が抱いた問題点を整理し, 本シミュレータの開発目的について述べる. 次いで第 3章では, 本シミュレータの機能について説明し, 第 4章で開発に際しての問題点を述べた後, 最後に第 5章でまとめを行う.

2　開発の目的

2.1　KUE-CHIP2 教育用ボードの問題点

我々が実際に学生実験で **KC2 ボード**を利用してきてみて, 学生や実験指導の担当者などから, 主に実験遂行上の効率という見地から, 以下のような点が問題であるとの指摘を受けることが多かった.

a) プログラムのロードに手間がかかる.

b) KUE-CHIP2 の各ハードウェアファシリティ（ACC, IX, PC など）が保持している値を, 総覧的に見ることができない.

c) メモリ内容の観察の手間も小さくない.

d) 実行の履歴を追ったり記録したりするのが大変である.

e) プログラムが少し複雑になっただけで, デバッグが極端に困難になる.

a) の問題は, たとえば立命大での学生実験のように, パソコンと接続することによって解決可能である. また **b)** についても, 九工大の KITE ボードなどでは解決されている. しかし, パソコンや KITE ボードを新たに導入するためには, さらに相当額の投資が必要であり, 本学科の学生実験の現状では, 現実的な解決策とはなり得ない.

d) や **e)** の問題も, 結局は **b)**, **c)** などに原因があり, CPU（あるいはマイクロコンピュータ) チップと簡単な周辺回路だけから成るこの種のボードでは, 自ずから限界があると考えられる.

実は, 上記の問題点はいわば第 2 段階とでも言うべきものであって, 初学者の場合には, 実際にはもっと初歩的な段階, すなわち, ボードの操作法を習得するまでにかなりの実験時間を費やしてしまう, という問題があることも事実ではある.

2.2　目的

本シミュレータの開発の目的は以下の 3 点である.

1) **KC2 ボード**のユーザインタフェースを改善し, 実験遂行上の効率を向上させる.

2) プラットホーム独立な実験環境を構築することにより, 実験環境の構築の自由度を向上させる.

3) ネットワーク環境を利用した教育環境を実現する.

KC2 ボードには動作制御や KUE-CHIP2 内部の観測を行うための各種スイッチ, および, 表示用 LED が備わっている. 本シミュレータでは, これらのユーザインタフェースに加え, プログラムやデータの入力, 表示, プロセッサの動作状態の確認などをより迅速に行えるように, いくつかの機能を追加した. これらにより, KUE-CHIP2 を用いた実験をより円滑に進めることができる.

さらに, Java 言語を用いてシミュレータを開発することにより, インターネットを介した教育を行うことができるになった. また, インターネットに接続され WWW ブラウザが利用できる計算機であれば, その機種や OS には依存せずに, 本シミュレータを利用することができる.

3 KUE-CHIP2 シミュレータ

我々は Java 言語を用いて 2 種類のシミュレータプログラムを同時に開発した. ひとつはスタンドアローンのアプリケーションであり, Java インタプリタの上で動作する. 今ひとつはアプレットであり, Java 対応の WWW ブラウザ上で動作する. 以下では, 前者を中心に説明する.

3.1 概要

KUE-CHIP2 シミュレータのメインウィンドウを図 1に示す. メインウィンドウは, 大きく分けてプロセッサの内部状態を表示する部分 (図 1の上部) と, プロセッサの動作を制御する部分 (同図の下部) とからなる.

- 表示部
  表示部には実行中のフェーズ, 各レジスタの値, および, フラグの値が表示される. これらを利用することでプロセッサの動作観測が行える.

  KC2 ボードでは, 各レジスタやフラグのうち, 観測したい対象を 1 つだけスイッチで選ぶ必要があった. 本シミュレータでは, すべてのレジスタの値とフラグの状態を常に表示するので, ユーザはスイッチを用いてそれらの切替えを行う必要がなくなった.

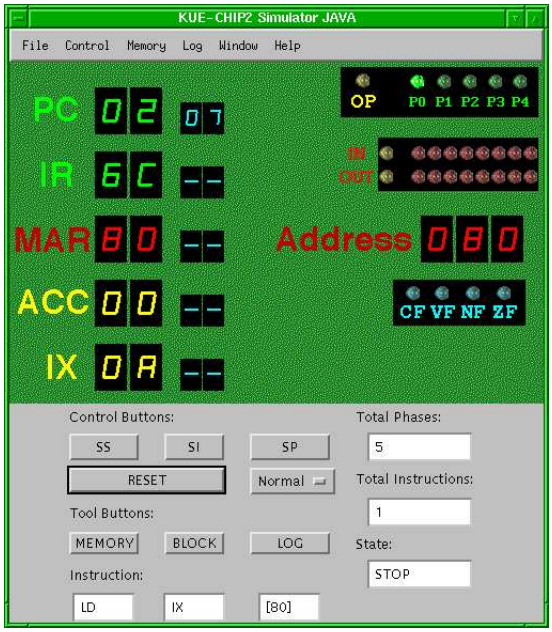

図 1: KUE-CHIP2 シミュレータメインウィンドウ

- 制御部
  制御部にある SS, SI, SP, RESET ボタンは, いずれも **KC2 ボード**と同様の機能を提供する. 主としてこれらの 4 つのボタンでプログラムの実行, 停止, 再開などの操作を行う.

  本シミュレータでは, これらに加え以下の 3 種類の Tool ボタンを設けた.

  MEMORY: このボタンを押すことで, 図 2 に示すメモリエディタ・ウィンドウが表示される. これにより, 一度に多くのアドレスのデータを一覧することができる. さらに, プログラムやデータの入力にもこのウィンドウを使用する. すなわち, 変更したいデータ (1 バイト) を反転表示させ, 値を 16 進数で入力することでメモリの内容を変更できる.

  BLOCK: このボタンを押すことで, 図 3に示すような KUE-CHIP2 のブロック図が表示される. プロセッサが停止中の時は, すべてのデータ線は青色で表示され, プロセッサが動作中の時には, フェーズ毎に有効となっているデータ線が赤色で表示される.

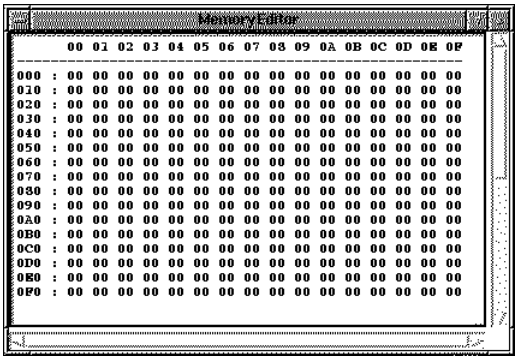

図 2: メモリエディタ ・ウィンドウ

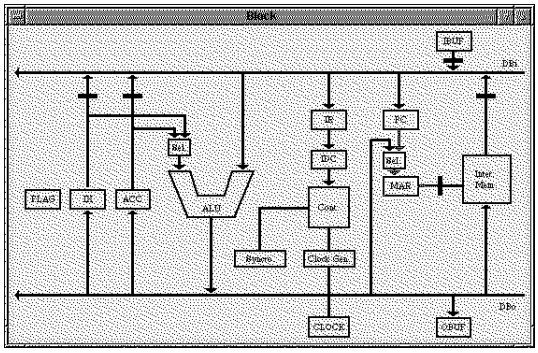

図 3: ブロック図表示ウィンドウ

LOG: このボタンを押すことで図 4に示すログ・ウィンドウが表示される. 必要に応じてプログラムの実行履歴を記録することができる.

これらの他にも, 以下のようなユーザインタフェースを提供している.

実行速度の選択 図 1中では Normal と表示されているボタンを用いて, プログラムの実行速度を 5 段階に調節できる.

Instruction: 3 つの表示用サブウィンドウの中に, 現在実行中の命令の逆アセンブル表示を行う.

Total Phases: プログラムの実行を開始してからの総実行フェーズ数をカウントして表示する.

Total Instructions: プログラムの実行を開始してからの総実行命令数をカウントして表示する.

State: 現在のプロセッサの動作状態 (STOP, RUNNING, BREAK, RESET のいずれか) を表示する.

3.2 本シミュレータの特徴

本シミュレータは, ブロック図の表示や実行履歴の記録など, ハードウェアでは実現の困難な機能を持っている. また, GUI ベースのソフトウェアであるため, 操作法も簡単で, 短時間での習得が可能である. これらのユーザインタフェースを備えていることにより, KUE-CHIP2 を用いた実験を円滑に進めることができる.

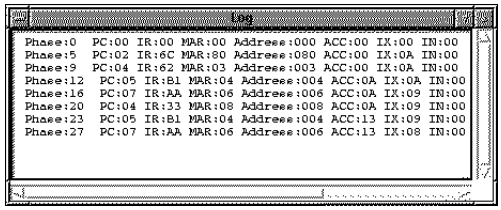

図 4: ログ ・ウィンドウ

3.2.1 プログラムの入力

プログラムの入力は, 前述したメモリエディタ ・ウィンドウを開いてプログラムを直接画面上で入力する方法の他に, File メニューからの指定でオブジェクトコードファイルを読み込む方法も可能にしている. ただし, このオブジェクトコードファイルは, テキスト形式で指定のフォーマットで記述されている必要がある.

3.2.2 プログラムの実行制御

プログラムの実行制御は **KC2 ボード**と全く同様に, SS, SI, SP ボタンを用いて行うことができる. 実行中はメインウィンドウに各レジスタの値とフラグの状態が常に表示される. また, ブロック図のウィンドウ (図 3) を表示しておけば, プロセッサ内部の動作も随時確認できる.

また, プログラムの実行を中断している間には, 各

レジスタの値とフラグの状態を自由に変更することができる. レジスタの値を変更するには, 値を変更したい 7 セグメントをマウスでクリックし, キーボードから 16 進数 2 桁で値を直接入力する. フラグの場合には, 変更したい LED をマウスでクリックすることで, 状態を反転させることができる.

さらに, ユーザはブレークポイントを設定することができる. 各レジスタの値を表示する 7 セグメントのすぐ横に, 小さな 7 セグメントがある. ここをクリックして, 先ほどと同様に値を設定しておくと, プログラムの実行中に各レジスタの値がその設定値に等しくなった時, プログラムの実行が一旦停止する.

3.2.3 プログラムの実行履歴

プログラム実行中の履歴は, 前述したログ・ウィンドウ（図 4）を開き, Log メニューから 'Start Log' の項目を選択することで, その時点からの実行履歴を記録することができる.

さらにプログラムの実行の終了後, 現在のメモリの内容, および, ログ・ウィンドウに表示されている実行履歴を, テキスト形式でファイルに保存することができる.

これらにより, プログラムの評価, 再利用などが簡単に行えるようになる.

3.2.4 マルチプラットホーム環境下での実行

本シミュレータの実行環境を図 5に示す.

Java のソースコードをコンパイルすることによって, バイト・コードと呼ばれる中間ファイルが生成される. このバイト・コードは Java Virtual Machine[4]（Java 仮想マシン, 以下 JVM と略記）のオブジェクトコードであり, 異なるプラットホーム間で共有することができる.

ローカルファイルシステム上にバイト・コードを用意しておけば, スタンドアローンのアプリケーションとして本シミュレータを利用できる. ただし, このバイト・コードの形式のプログラムを実行するためには, 各プラットホームに対応した JVM, すなわち Java インタプリタが必要となる.

一方, インターネットを通じてこのバイト・コードを送ることで, 遠隔地にあるマシンでも本シミュレータを利用できる. HTTP によって HTML ドキュメントをダウンロードする際, Java プログラム

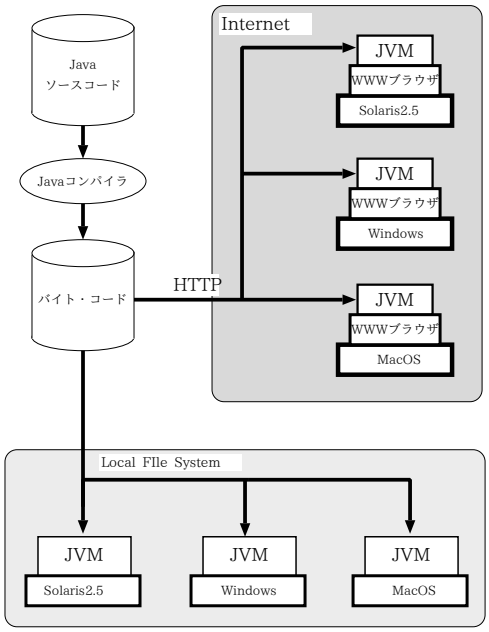

図 5: KUE-CHIP2 シミュレータの実行環境

が中に埋め込まれていると考えればよい. このとき, ユーザのマシン（WS または PC）には WWW ブラウザがインストールされていなければならないが, 最近の主要なブラウザには JVM が実装されており, アプレットと呼ばれる形式で実行される. 別途 Java インタプリタを用意する必要はない.

実際に, Solaris2.5, Windows95(NT), MacOS, IRIX などで動作確認済みである.

3.3 スタンドアローン版とアプレット版の相違点

アプレット版では, 現在のところセキュリティ上の問題から, ファイルの読み込み／保存ができない. したがって, 前述した本シミュレータの機能の一部が制限される. この点に関しては, 今後 Java 言語のバージョンが上がることで解決されると考えられる.

3.4 シミュレータ開発の経緯

本シミュレータの開発は, 1996 年 4 月から開始した. 開発に利用した環境, 作業時間などを表 1に示す. 開発の際に主に参照したのは文献 [2, 3] である.

表 1: 開発環境および作業時間

開発環境	主に Solaris2.5 Windows95 MacOS
開発用ソフトウェア	Java Developers Kit (release 1.02)
作業時間	9ヶ月 (実質約 60 日)

また, Java 言語に関するメーリングリスト[1]も利用した.

4 開発上の問題点

ここでは, 本シミュレータを開発する過程で生じた主な問題点 3 つについて, 順に述べる.

4.1 異なるプラットホーム (JVM) による動作の違い

Java 言語を用いてプログラムを開発する利点として, 異なるプラットホームであっても JVM が実装されてさえいれば, そのプログラムの実行が可能になる, ということを先に述べた. しかし実際には, そのシステムに実装されている JVM の動作の違いによって, シミュレータの動作も異なる場合がある.

GUI を持つ本シミュレータのようなアプリケーションでは, 特にこの問題が顕在化することが多いようである.

図 6および 7 にその例を示す. この図は本シミュレータの開発初期のメモリエディタ ·ウィンドウである. それぞれのプラットホーム上での JVM の実装により, たとえばフォントの大きさやテキスト表示領域の大きさが異なっている. そのために, Solaris 上では図 6のように正しく表示されていても, Windows 上では図 7のように正常に表示されない, ということが起こった.

また, 同一のプラットホーム上でも, WWW ブラウザ上で実行されるアプレットと, スタンドアローン ·アプリケーションの場合とで, 動作が異なる場合もある. その例を図 8に示す.

[1] Java House ML, Java House Brewers ML

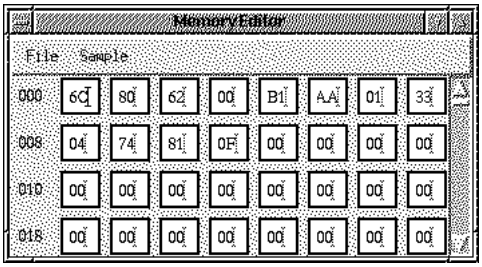

図 6: 開発初期のメモリエディタ ·ウィンドウ (Solaris2.5)

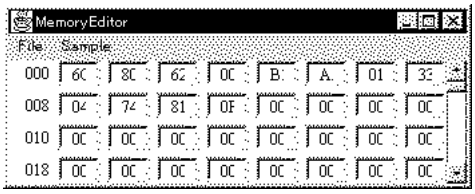

図 7: 開発初期のメモリエディタ ·ウィンドウ (Windows95)

この図は, Solaris2.5 上の Netscape3.01 でアプレットを実行した場合のメモリエディタ ·ウィンドウである. このように WWW ブラウザで実行した場合, ウィンドウの高さが図 2のスタンドアローン ·アプリケーションの場合と異なっている.

この他, 同じアプレットでも, 異なる WWW ブラウザ 例えば, Netscape と MS-Explorer) で動作が異なる場合もある.

以上のように, 同一のプログラムを異なるプラットホームで動作させるという目的で Java 言語を開発に使用する場合, 特に GUI 関連で問題が起こる可能性があることに注意しなければならない.

4.2 セキュリティ上からの機能制限

WWW ブラウザ上で実行されるアプレットには, 主として以下のようなセキュリティ上の制限事項がある.

- ローカルシステム上のファイルの読み書き
- ユーザ名, ホームディレクトリ名の取得

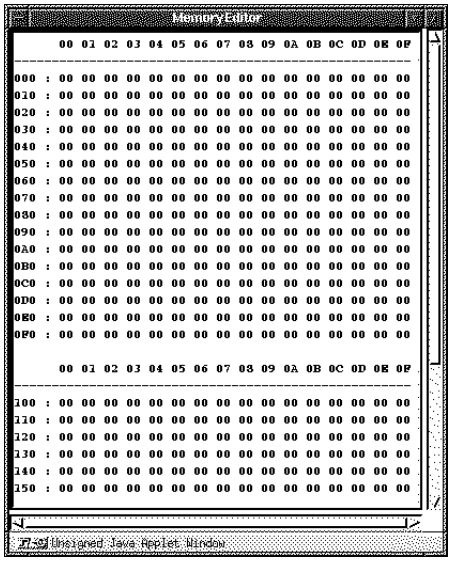

図 8: Netscape で表示したメモリエディタ・ウィンドウ

- アプレット自身のロード元以外の計算機とのネットワーク接続

したがって, 本シミュレータのアプレットプログラムでは, File メニューを使用して次のような機能を実装することができない.

- オブジェクトコード・ファイルの読み込み
- メモリ内容やログ・ウィンドウの内容の保存

これらの解決方法として, アプレットのロード元の計算機（以下アプレット・サーバと記す）と通信し, そのアプレット・サーバのファイルシステムに対して読み込みや保存を行う方法が考えられる. しかしこの方法の場合, 接続しているユーザの作成したファイルをすべてアプレット・サーバのファイルシステムに保存しておかなければならず, ユーザが非常に多い場合などに負荷が集中する可能性がある.

このような理由から, 現時点では, ファイルの入出力に関する機能をアプレット版のシミュレータでは提供していない.

ただし, 近い将来 Java 言語の新しいバージョンが開発されることによって, セキュリティ上の制限事項が緩和されると考えられる.

4.3 Java プログラムの実行速度

先にも述べたように, Java 言語のプログラムはバイト・コードの形式で, 各マシン上の JVM によって実行される. 実際には, このバイト・コードは JVM を実装した Java インタプリタによって解釈, 実行される. このため, C 言語などで記述して各プラットホームの機械命令に変換されたコード（以下ネイティブ・コードと呼ぶ）に比較すれば, 圧倒的に実行速度は遅くなる. 一般に, ネイティブ・コードに対して Java インタプリタで実行する場合, 10 倍以上遅くなると考えられている.

本シミュレータはプロセッサの動作原理の理解を目的としているので, 動作速度に関してはあまり大きな問題はないと考えている. ただし, 制御部で提供している実行速度の選択の幅は, 実際にはかなり限定されてものとなっている.

5 おわりに

Java 言語を用いて KUE-CHIP2 シミュレータを開発した. 本シミュレータを用いれば, KUE-CHIP2 の動作制御, 観測という作業を従来の **KC2 ボード**に比べて円滑に進めることができる. それと同時に, 異なる計算機環境下での実験, さらにはインターネットを介しての実験を行うこともできるようになった.

シミュレータの機能については, 現在のものが必要かつ十分なものであるか, 今後も検討していく必要があろう. ソフトウェアによる実現であるだけに, あれもこれもと種々の機能を詰め込み過ぎる傾向が出ることには注意が必要である. 本稿では, **KC2 ボード**の問題点を幾つか指摘したが, 本シミュレータにはない「実物」の持つ良さがあることも事実である. それぞれの特長を活かした実験課題を作成していくことにより, さらに様々な可能性が広がることを期待したい.

なお, 本シミュレータに関する情報は, URL アドレス: `http://alia.dj.kit.ac.jp/~yamamu2s/Java/KUE/KUE.html` で公開している.

謝辞

本シミュレータの開発に際し, 貴重な御意見を賜わりました広島市立大学の越智裕之先生に深く感謝致します.

参考文献

[1] J. Gosling, B. Joy, and G. Steele: "The Java Language Specification," *URL*: `http://www.javasoft.com/nav/read/docindex.html`.

[2] D. Flanagan(永松 健司訳): "JAVA クイックリファレンス," オライリー ・ジャパン (1996).

[3] 有我 成城, 衛藤 敏寿, 佐藤 治, 白神 一久, 西村 利浩, 村上 列: "Java 入門," 翔泳社 (1996).

[4] T. Lindbolm and F. Yellin: "The Java Virtual Machine Specification," *URL*: `http://www.javasoft.com/nav/read/docindex.html`.