Vol.2013-CE-122 No.23
Vol.2013-CLE-11 No.23
2013/12/15

Web ベース 8085 マイクロプロセッサシミュレータの SCORM パッケージ化とその Moodle 上での評価

村松一弘†

筆者らは，e ラーニング上でマイクロプロセッサの実習を行うために，8085 マイクロプロセッサシミュレータを PHP で開発し，それを Moodle をベースにした VLE（Virtual Learning Environment）上で活動モジュールとして実装してきた．しかし活動モジュールとしての実装は汎用性に欠ける．例えば，Moodle をアップデートする際には，Input Form や View Form を修正しなければならない．そこで，任意の e ラーニングプラットフォーム上で 8085 マイクロプロセッサシミュレータが容易に実装できるように，SCORM パッケージ化を施した．本稿では，初めに 8085 マイクロプロセッサシミュレータの活動モジュールとしての Moodle への実装について述べる，次に，8085 シミュレータの JavaScript による SCORM パッケージ化について述べる．最後に，SCORM パッケージの Moodle 上での評価を，前述の活動モジュールと比較しながら行う．その結果，活動モジュールの方が実行速度の速いことが明らかになる．

SCORM Packaging of Web-based 8085 Microprocessor Simulator and Its Evaluation on Moodle

KAZUHIRO MURAMATSU†

We developed an 8085 microprocessor simulator on PHP, and it was implemented on the Moodle-based LMS of our college as an activity module. However, the implementation as an activity module is a lack of portability. When the Moodle-based LMS is updated, for example, it is necessary to modify the input and view forms in the activity module according to the upgraded version of Moodle. Thus, we have packaged the 8085 microprocessor simulator for SCORM by the use of JavaScript in order to improve the portability. In this report, firstly the implementation of the 8085 microprocessor simulator as an activity module is described. Next, we describe SCORM packaging of the simulator. Finally, the SCORM package is compared with the above-mentioned activity module. The result shows that the execution speed of the activity module is faster than that of the SCORM package.

1. はじめに

近年，高等教育機関において LMS（Learning Management System）あるいは VLE（Virtual Learning Environment）が幅広く利用されている．ブータンでは，世界銀行からの資金援助と University of Colombo School of Computing（UCSC）からの技術支援により，2011 年 7 月に国内の全ての高等教育機関で Moodle 1.9 ベースの VLE が導入された[1][2].

一方，理工系の高等教育機関では実験・実習が重視されており，これらを通じて実践的な技術を習得する．日本の高等教育機関では，講義，演習，実験が別科目としてカリキュラムが構成されているが，本学では一科目が講義，演習，実験で構成されており，これらの授業を一人の教員で行っている．例えばマイクロプロセッサの科目は，週 3 時間の講義，1 時間の演習，2 時間の実習で構成されている．そのため，講義と実験が密接に関係しており，実験は実践的な技術を習得するとともに，講義を理解するのに活用されている．

しかし実験・実習のための実習装置や実験器具は，当然ながらその授業時間にのみ利用できるため，予習で使用したり，実験レポートを作成する際に再実験をすることはできない．また，これらの実験・実習の装置や器具は多岐に渡り，その維持管理には多くの費用と労力を要する．そこで筆者らは，前述の実習装置を摸擬するシミュレータを開発し，これを LMS に実装することにより，学生が LMS 上で事前にこのシミュレータを使って実験・実習のプロセスを理解し，事後に実験・実習の復習が可能にする研究開発を始めた．

ただし，理工系の高等教育における実験・実習は多岐に渡るので，本学で開講されているマイクロプロセッサ関係の科目に対象を絞り，これらの科目の実習で利用されている 8085 マイクロプロセッサトレーナーを摸擬する 8085 マイクロプロセッサシミュレータを PHP（Hypertext Preprocessor）で開発した[3][4]．PHP で開発した理由は，このシミュレータを本学の Moodle 1.9 をベースにした VLE 上の活動モジュールとして実装するためである．

しかし活動モジュールとしての実装には欠点がある．すなわち Moodle がアップグレードされるたびに活動モジュールの中の Input Form や View Form を修正する必要がある，また活動モジュールを修正したのち VLE の管理者に修正した活動モジュールのアップロードを依頼しなければならない．これらの欠点を克服するために，筆者らは 8085 マイクロプロセッサシミュレータの SCORM（Sharable Content Object Reference Model）パッケージ化を施した[5]．このパッケージ化により，任意の LMS でシミュレータの実装が容易になる標準化が実現できる．

†王立ブータン大学科学工科カレッジ
College of Science and Technology, Royal University of Bhutan

以下，§2 で 8085 マイクロプロセッサシミュレータの開発，§3 でマイクロプロセッサシミュレータの活動モジュールとしての VLE への実装，§4 でマイクロプロセッサシミュレータの SCORM パッケージ化について述べる．§5 が SCORM パッケージ化の評価，§6 がまとめと今後の課題である．

2. 8085 マイクロプロセッサシミュレータ

8085 マイクロプロセッサはインテルにより 1976 年に開発された 8 ビットのマイクロプロセッサで，現在では実用に供されていない．しかし，アーキテクチャが簡単でプログラミングの概念を習得するのに必要最低限の命令セットが用意されているので，教材としてはふさわしく，そのシミュレータは GNUSim8085，Gsim85，8085 SimuKit など多数存在する[6][7][8][9]．これらは Windows ベースあるいは Linux ベースで開発されており，Web プログラミング言語で開発されていないので, LMS との親和性は無い. そこで，筆者らは Moodle 上に実装することを念頭に置き，PHP 言語で 8085 マイクロプロセッサシミュレータを開発した．

開発に際して，以下の設計方針を採用した．

- ユーザが記述内容を理解しやすいアセンブリプログラム入力とする．
- アセンブリプログラムの入力ミスを低減させるため，入力があり得ない項目は入力不可能にする．
- プログラム実行モードとして，RUN モードと STEP モードの 2 モードを用意する．RUN モードはバッチ処理であり，STEP モードはプログラムの 1 ステップごとに実行結果を確認できる．
- ユーザがタブレット端末で利用することを考慮し，キーボードを使用せず全てマウス入力にする．

開発した 8085 マイクロプロセッサシミュレータは，メインモジュール，実行モジュール，GUI（Graphical User Interface）モジュールから構成される（図 1）．メインメインモジュールと実行モジュールが LMS サーバ側で実行され，GUI モジュールがクライアント側で実行される．

以下に，各モジュールを解説する．

2.1 メインモジュール

メインモジュールは，クライアント側の GUI モジュールから送られたアセンブリプログラムをマシン語プログラムに翻訳する．例えば，メインモジュール中の以下のプログラムは，命令‘HLT’をマシン語‘76H’に翻訳する．

```
if(!empty($hlt)) {
    $mnemonic[$insert] = "HLT";
    $mc[$insert] = 0x76;
    $codeinput = "opcode";
    if($insert == $addr) $addr++;
    $insert = $addr;
}
```

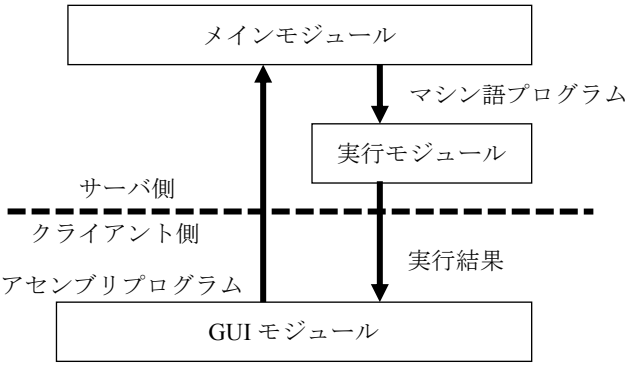


図 1　8085 マイクロプロセッサシミュレータの
モジュール構成とデータの流れ

Figure 1　Modules and Data Flows in the 8085 Microprocessor Simulator.

翻訳されたマシン語プログラムをクライアント側のクッキーに格納する．クッキーのサイズ制限から，マシン語レベルで 4096 バイト以下のプログラムが入力可能である．4096 バイトというサイズはアセンブリプログラムとしては約 2000 行の長さに相当し, 教育用プログラムとして十分なサイズであると考えられる．クッキーに格納されているマシン語プログラムは，プログラムを実行する際に実行モジュールに引き渡される．

またアセンブリコードを入力する際に，次に入力可能な項目の情報を GUI モジュールに渡す. 前述のメインモジュールのプログラムにおいて，$codeinput = "opcode";がクライアント側の GUI モジュールに引き渡される情報である．

2.2 実行モジュール

実行モジュールでは，メインモジュールから渡されたマシン語プログラムを実行し，プログラムの各ステップでのレジスタ，プログラムカウンタ，スタックポインタなどの値を GUI モジュールに送る．実行モードには，ランモードとステップモードの 2 種類がある．ランモードでは，RUN ボタンによりプログラムを一括して実行し，全てのステップでのレジスタなどの値を GUI モジュールに同時に引き渡す．一方ステップモードでは，プログラムはステップごとに実行され, 各ステップでの実行結果が GUI モジュールに送られる．STEP ボタンにより次のステップに進む．

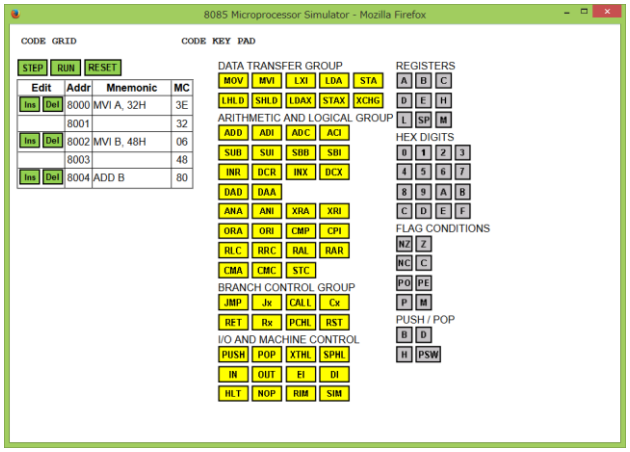
2.3 GUI モジュール

ユーザは GUI モジュールを通じてアセンブリプログラムを入力し，入力されたアセンブリプログラムはメインモジュールに引き渡される．図 2 に入力画面を示す．アセンブリコードは，処理の内容を指示するオペコードと処理の対象を指示するオペランドから構成される．図 2(a)がオプコード入力画面で，図 2(b)がオペランド入力画面である．

図 2 に示すように，入力が可能な項目に対応するボタンのみを色付き表示で有効にし，入力が不可能な項目のボタンを灰色表示で無効にすることにより，入力ミスを低減さ

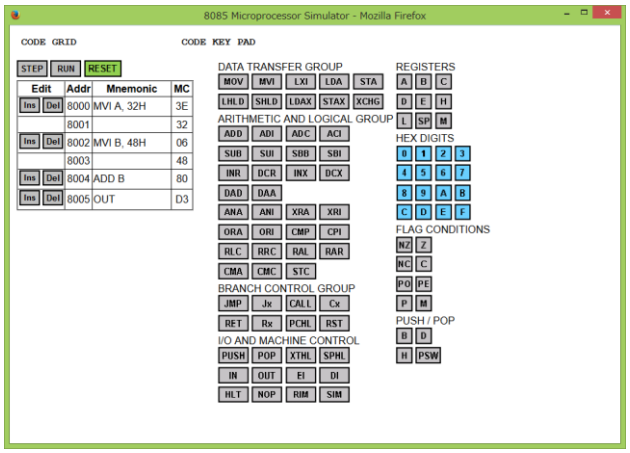
Vol.2013-CE-122 No.23
Vol.2013-CLE-11 No.23
2013/12/15

せる設計にした．図 3 に，実行モジュールから引き渡されたアセンブリプログラムの実行結果表示画面を示す．



(a) オプコード入力

(a) Opcode Input.



(b) オペランド入力

(b) Operand Input.

図 2　アセンブリプログラムの入力画面

Figure 2　Input Windows of an Assembly Program.

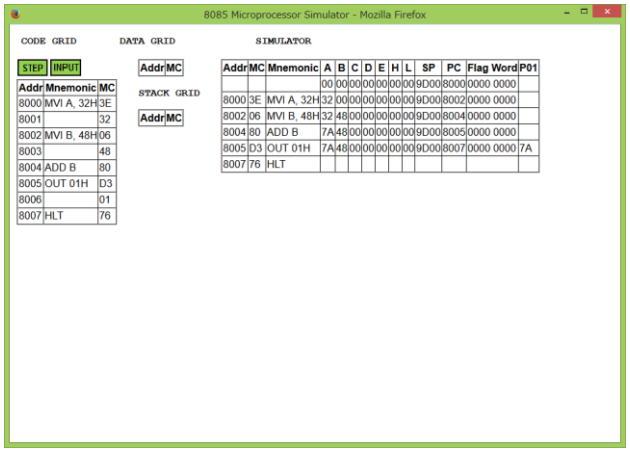


図 3　アセンブリプログラムの出力画面

Figure 3　Output Window of an Assembly Program.

3. 活動モジュールとしての LMS への実装

Moodle には，課題，フォーラム，用語集，レッスン，小テスト，チャットなどのコース機能があり，これらは活動モジュールとして実現される．ここで，コースとは授業科目に相当する．初めに，開発されたマイクロプロセッサシミュレータを Moodle の活動モジュールとして実装した．

一般に，新たな活動モジュールを開発する際には Moodle の Web サイトからテンプレート NEWMODULE をダウンロードして，これを書き換える方法が採られるので，筆者らもこの方法に従う[10][11]. テンプレート NEWMODULE は，4 つのフォルダと 15 のファイルから構成される．以下では，テンプレートからの主な修正点を述べる．

3.1 Input Form

教師が担当するコースに活動モジュールを新たに登録する際に，その活動モジュールを登録するための入力フォームを設定するファイルが mod_form.php である．ここでは図 4 に示すように，教師が担当コースでのマイクロプロセッサシミュレータのタイトルとその操作方法を入力しなければならないように mod_form.php を修正した．この Input Form は Moodle 1.9 と Moodle 2.4 で異なり，Moodle 2.4 にアップグレードする際には変更しなければならない．

3.2 View Form

登録後，学生は活動モジュールとしてマイクロプロセッサシミュレータを利用することができる．view.php は，学生が活動モジュールにアクセスした時に表示される画面を設定する．ここでは，図 5 に示すように‘Click here to run 8085 Simulator’とシミュレータへのリンクが作成され，学

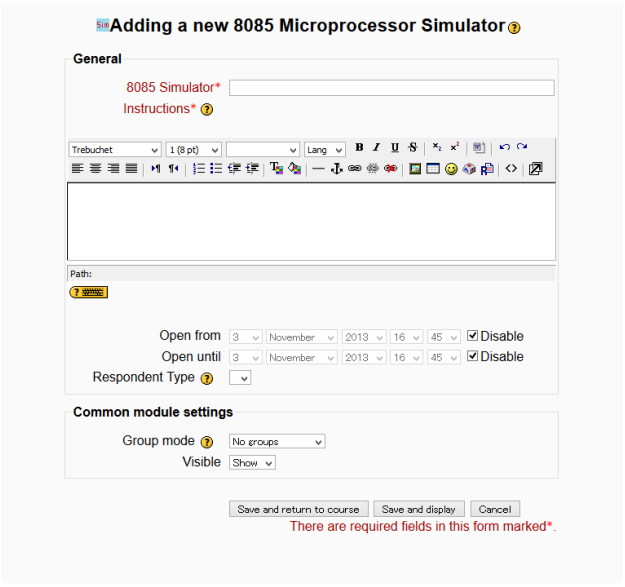


図 4　Moodle におけるマイクロプロセッサシミュレータの登録画面

Figure 4　Registration Screenshot of Microprocessor Simulator Module on Moodle.

Vol.2013-CE-122 No.23
Vol.2013-CLE-11 No.23
2013/12/15

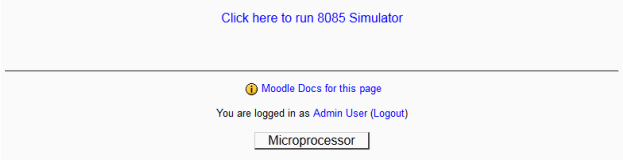


図 5　Moodle におけるマイクロプロセッサシミュレータの起動画面

Figure 5　Execution Screenshot of Microprocessor Simulator Module on Moodle.

生はこの表示をクリックすると，ポップアップ形式で前節のシミュレータが起動する設定にした．これは，シミュレータの GUI を Moodle の GUI と独立させるためである．Moodle 1.9 では，view.php 中のポップアップ起動に相当する下記のコードを挿入する．

```
link_to_popup_window("/mod/simulator/simulator.php",
    "simulator$course->id$simulator->id$groupparam",
    "$strentersimulator", 650, 950,
    get_string('modulename', 'simulator'));
```

ここで，simulator.php は前節で述べたマイクロプロセッサシミュレータのメインモジュールである．これが Moodle 2.4 では，

```
$link = new moodle_url
    ('/mod/simulator/simulator.php');
echo $OUTPUT->
    action_link($link,get_string(
    'entersimulator', 'simulator'),
    new popup_action ('click', $link,
    "simulator$course->id", array(
    'height' => 650, 'width' => 950)));
```

のように変更しなければならない．

3.3 Local Language Form

これはローカル言語ファイルと呼ばれ，Moodle の GUI の表記を変更する．例えば，

```
$string['modulename'] = '8085 Microprocessor Simulator';
$string['simulatorname'] = '8085 Simulator';
$string['timeavailable'] = 'Open from';
$string['timedue'] = 'Open until';
$string['entersimulator'] = 'Click here to run 8085
    Simulator';
```

と変更すると，図 4 に示すように modulename が‘8085 Microprocessor Simulator’, simulatorname が‘8085 Simulator’, timeavailable が‘Open from’，timedue が‘Open until’と表示される．また，図 5 の起動画面でシミュレータへのリンクが‘Click here to run 8085 Simulator’と表示される．

3.4 Icon Form

マイクロプロセッサシミュレータを表すアイコン作成した．これにより，教師がマイクロプロセッサシミュレータを登録した後，コース画面においてシミュレータの実行モジュールが図 6 のようにアイコンとともに表示される．

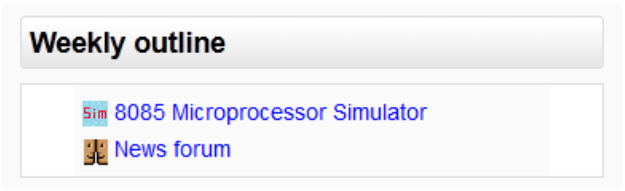


図 6　マイクロプロセッサシミュレータ登録後のコース画面

Figure 6　Course View after Registration of Microprocessor Simulator Module.

4. SCORM パッケージ化

前節で述べたように，Moodle 1.9 を Moodle 2.4 にアップグレードする際には，活動モジュールの中の Input Form や View Form を修正しなければならない．今後も Moodle がアップデートするたびに，活動モジュールを修正する必要があると推測される．また活動モジュールを修正したのち，VLE の管理者に修正した活動モジュールのアップロードを依頼する必要がある．

このような欠点を克服するために，SCORM 1.2 と SCORM 2004 の両方を取り上げ，8085 マイクロプロセッサシミュレータの SCORM パッケージ化を施した．SCORM は e ラーニングの標準化技術であり，SCORM パッケージ化により任意の e ラーニングシステム上でシミュレータの実装が容易になる．一般に SCORM パッケージは，コース構造を表すマニフェストファイルと学習コンテンツに相当する SCO（Sharable Content Object）から構成される[12][13][14]．ここでは，8085 シミュレータが SCO に相当する．

以下に，8085 マイクロプロセッサシミュレータの SCORM パッケージにおけるマニフェストファイルと SCO について詳述する．

4.1 マニフェストファイル

SCORM 1.2 では，マニフェストファイルはメタデータ，Organization 要素，Resource 要素から構成される．メタデータはタイトルやキーワードなどの SCO の内容や特徴などを表現するデータであり，Organization 要素は SCO の階層構造を記述するために用いられる．Resource 要素で SCO を表す実際のファイルを指定する．メタデータを付与するか否かは任意であり，ここでは省略した．

8085 マイクロプロセッサシミュレータの SCORM 1.2 パッケージにおけるマニフェストファイルを以下に示す．

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest identifier="Manifest">
    <organizations default="ORG">
```

```
        <organization identifier="ORG"
                                    structure="hierarchical">
            <title>Organization</title>
            <item identifier="ITEM" isvisible="true"
                                identifierref="RESOURCE">
                <title>Run 8085 Microprocessor
                                            Simulator</title>
            </item>
        </organization>
    </organizations>
    <resources>
        <resource identifier="RESOURCE"
                        type="webcontent" href="index.html">
            <file href="index.html" />
        </resource>
    </resources>
</manifest>
```

SCORM 2004 では，学習コンテンツの提示順序の指定や学習順序の変更が可能なシーケンシング&ナビゲーション要素をマニフェストファイルに含めることができる[15][16]．しかしここでは，前節までの 8085 マイクロプロセッサシミュレータを SCORM パッケージ化するだけなので，このような情報は考慮しない．また SCORM 2004 のマニフェストファイルは, 基本的に SCORM 1.2 のマニフェストファイルの上位互換となっていので，SCORM 1.2 と SCORM 2004 の間でマニフェストファイルに差異は無い．

4.2 SCO

§2 で述べた 8085 マイクロプロセッサシミュレータは PHP で開発したので, 図 1 で示すようにサーバとクライアントの間で自由に変数の受け渡しができる．しかし SCORM 1.2 では，API（Application Programming Interface）関数 LMSGetValue および LMSSetValue，SCORM 2004 では GetValue および SetValue を用いてサーバ・クライアント間で変数を受け渡しすることができるが，サーバ側で変数の処理をすることができない．そこで筆者らは，8085 マイクロプロセッサシミュレータを PHP から JavaScript に書き直し，全ての処理をクライアント側で実行するように修正した．移植を簡単にするために，GUI 部と，メインモジュールと実行モジュールを統合した処理部の間のデータの受け渡しはクッキーを通じて行うようにした（図 7）．

4.3 SCORM 上での機能拡張

前小節で述べた SCO は，8085 マイクロプロセッサシミュレータを PHP から JavaScript に変換しただけで, SCORM の API を全く利用していない．そこで，§4.2 で述べた API 関数 LMSGetValue/LMSSetValue あるいは GetValue/SetValue を利用して，入力途中のアセンブリプログラムをサーバ側に一時的に保存し，後で保存したプログラムをロードすることができるように機能拡張を施した．

マニフェストファイルには変更が無い．一方，SCO の処理部では，図 8 に示すように，SCORM 1.2 の場合，LMSInitialize，LMSSetValue，LMSFinish の順で API 関数を呼び出して，アセンブリプログラムを LMS に保存する．そして，LMSInitialize，LMSGetValue，LMSFinish の順で API 関数を呼び出して, 保存したプログラムをロードする．

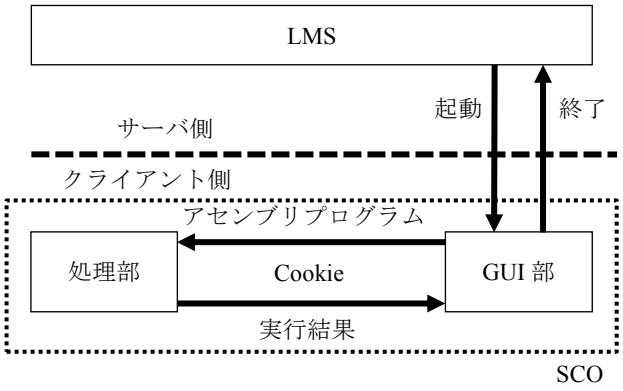
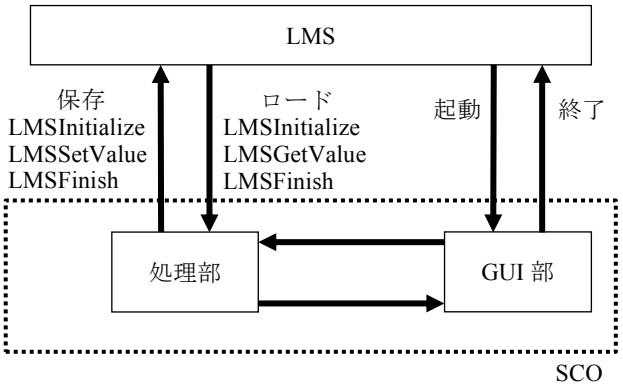
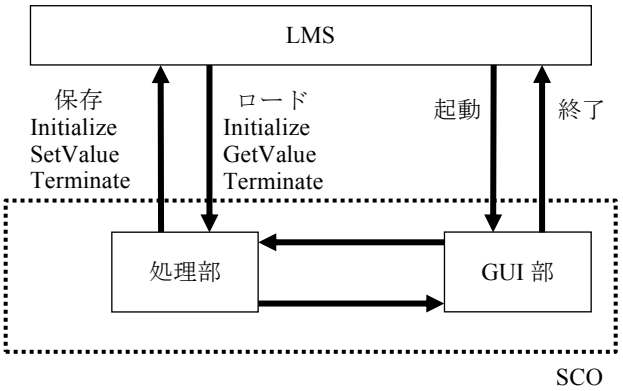


図 7　SCORM パッケージのモジュールとデータの流れ
Figure 7　Modules and Data Flows in the SCORM Package.



(a) SCORM 1.2



(b) SCORM 2004

図 8　API 関数によるプログラムの保存とロード
Figure 8　Saving and Loading of an Assembly Program by API Functions.

SCORM 2004 の場合には，保存が Initialize，SetValue，Terminate の順で API 関数を呼び出し，ロードが Initialize，GetValue，Terminate の順で API 関数を呼び出す．

図 9 に，機能拡張を施した後の入力画面を示す．

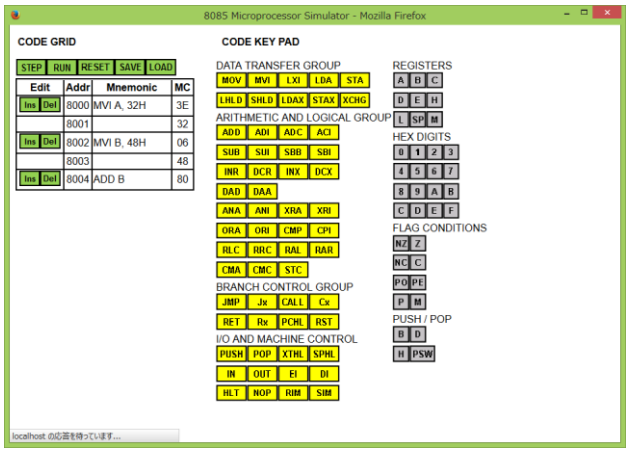


図 9　機能拡張後の入力画面

Figure 9　Input Windows with Saving and Loading Function.

5. SCORM パッケージの評価

§3 で述べたように，マイクロプロセッサシミュレータを活動モジュールとして本学の Moodle をベースにした VLE に実装した．また，全く同じ機能を持つ§4.2 の SCORM パッケージを同様に VLE に登録した．その上で，実際にマイクロプロセッサの実習に使用し，29 名の学生による実行速度の比較評価を行った．その結果を下記に示す．

表 1　実行速度の比較評価結果

Table 1　Comparison about Execution Speed.

	活動モジュール	SCORM パッケージ
非常に速い	14	0
速い	15	3
普通	0	11
遅い	0	8
非常に遅い	0	7

表 1 から，SCORM パッケージに比べて活動モジュールの実行速度が速いことが明らかになる．

6. まとめと今後の課題

本報告では，初めに 8085 マイクロプロセッサシミュレータを開発し，それを本学の VLE に活動モジュールとして実装した．しかし活動モジュールとしての実装はポータビリティに問題があるため，次にシミュレータの SCORM パッケージ化を施し，問題点を克服した．しかし双方の実行速度を比較し，活動モジュールの方が実行速度の速いことが明らかになった．

SCORM パッケージの実行速度が遅い理由は，Moodle 上での SCORM プレーヤーの実装に問題があると推測される．今後，Moodle 上での SCORM プレーヤーの実装の問題点を明らかにし，改善方法を提案していきたい．

謝辞　マイクロプロセッサシミュレータの本学 VLE への実装に御協力いただいた王立ブータン大学科学工科カレッジ情報工学科スタッフの Yeshi Wangchuk 氏に感謝の意を表する．また活動モジュールと SCORM パッケージの比較評価に協力してくれた，王立ブータン大学科学工科カレッジ電子通信工学科 3 年次の学生諸氏に感謝する．

参考文献

1) Annual Report, Royal University of Bhutan (2011), http://www.rub.edu.bt/images/key-documents/annual-reports/annual_report_2011.pdf
2) Muramatsu, K. and Wangchuk, S.: Current Status of e-Learning and Its Challenges at Institutions of Higher Education in Bhutan, IPSJ SIG Technical Report, Vol.2013-CLE-9, No.13 (2013).
3) Muramatsu, K: Development of Web-based 8085 Microprocessor Simulator and Its Implementation on LMS, Int. J. Applied Engineering Research, Vol.7, No.11, pp.1209-1213 (2012).
4) 村松一弘: Web ベース 8085 マイクロプロセッサシミュレータの開発とその LMS への実装，情報処理学会研究報告，Vol.2013-CLE-9, No.5 (2013).
5) SCORM, http://www.adlnet.org/scorm/
6) GNUSim8085: 8085 Simulator for Linux and Windows, http://gnusim8085.org
7) GSim85: An 8085 Microprocessor Simulator, http://gsim85.sourceforge.net
8) Chehab, A., Hanna, S., Kabalan, Y.K. and El-Hajj, A.: 8085 microprocessor simulation tool “8085 SimuKit”, Computer Applications in Engineering Education, Vol.12, No.4, pp. 249-256 (2004).
9) 8085 Simulator Project: 8085 Simulator Free Download, http://8085simulator.codeplex.com
10)　Moodle.org: NEWMODULE Documentation - MoodleDocs, http://docs.moodle.org/dev/NEWMODULE_Documentation
11)　Moore, J. and Churchward M.: Moodle 1.9 Extension Development, Packt Publishing, Birmingham (2010).
12)　SCORM 1.2 Specification, http://www.adlnet.gov/resources/SCORM-1-2-Specification?type=technical_documentation
13)　SCORM 2004 4th Edition, http://www.adlnet.gov/resources/SCORM-2004-4th-Edition-SpecificatioS?type=technical_documentation
14)　SCORM テキスト - 日本 e ラーニングコンソシアム, http://www.elc.or.jp/Portals/0/data/SCORM/SCORM_textV1.0.pdf
15)　SCORM Users Guide for Programmers, http://www.adlnet.gov/resources/SCORM-Users-Guide-for-Programmers?type=technical_documentation
16)　SCORM 2004 解説書 - 日本 e ラーニングコンソシアム, http://www.elc.or.jp/Portals/0/data/SCORM/WG1_manual_106.pdf