PHPを用いたアセンブラ言語CASL IIプログラムの簡易シミュレーションシステム

佐　賀　信　裕（短期大学部情報メディア学科）

Simple Simulation System of Assembly Language CASL II Program Using PHP

Nobuhiro SAGA (Junior College, Department of Information and Multimedia Technology)

Abstract

In the Fundamental Information Technology Engineer Examination which is held by IPA (Information-technology Promotion Agency, Japan) biannually, questions on CASL II which is an assembly language for a virtual computer COMET II are asked. For many students, it is useful that they can visually get an output from CASL II program to confirm whether the answer to a multiple-choice question is correct or not. This paper describes a simple simulation system for CASL II programs which are converted to PHP scripts using the switch statement in PHP.

Key words: *CASL II, PHP, assembler language, programming, FE*

1．はじめに

情報処理推進機構[1]（IPA）が毎年春・秋期に行っている基本情報技術者試験では，プログラミングの問題として仮想計算機COMET IIのアセンブラ言語CASL IIの問題が出題されている。平成21年度の新試験制度になってから，プログラミング問題は量的に緩和されたが，プログラミング問題もやはり重要であることに変わりはない。

問題は解答群から正解を選択するようになっている。したがって，学生が解答群で与えられた解答をいろいろ選択して実際にプログラムを実行させ，その結果を視覚的に確認できれば，問題の正解を得るだけでなく，プログラミングやアルゴリズムの理解を深めることができる。特に，eラーニングの利用者にとって，Web上でプログラムを容易に実行でき，その結果を確認できる環境が整っておれば，学習の効果が上がると考えられる。それゆえ，学生が問題のいろいろな解答に対して容易にプログラムを実行し，その結果を確認できる環境が求められる。

ところで，筆者は基本情報技術者試験のeラーニングシステム[2]上で，いろいろな問題のシミュレーションプログラムを開発しており，平成20年度からは，午後に出題されるプログラミング問題の簡易シミュレーションシステムの開発も行っている。このeラーニングのサーバーには，短期大学部の在学生用ホームページの「資格取得」欄の「基本情報技術者eラーニング」からもアクセスできる。ただし，サーバーへのアクセスは学内からのみ可能で，外部からのアクセスには制限がかけられている。

そこで，本論文では，基本情報技術者試験の午後問題に出題されているアセンブラ言語CASL IIプログラムをPHPのswitch文を用いることによって形式的にPHPのスクリプトに変換できることと，そのPHPスクリプトによるCASL IIプログラムの簡易シミュレーションシステムについて報告する。

平成21年5月30日受付

2．プログラム内容に基づいた変換

具体的に，基本情報技術者試験【平成20年度春期午後問題】問13の問題[1,3]の〔プログラム4〕副プログラムPRINTをPHPでシミュレーションする例を報告する。いろいろな解答の選択に対して，ジャンプ命令を用いた繰り返し，または繰り返しからの脱出の場合を考える。

HTML（一部，PHP）形式で表した副プログラムPRINT（数値を10進数の数字列に変換して出力するプログラム）を図2.1に示す。解答群eの正解はselectタグを用いて選択できるようになっている。

2.1　CASL IIプログラム

〔プログラム4〕

```
PRINT    START
         RPUSH
           :
```

```
LOOP4   CALL    DIVIDE          ；数値/10
        OR      GR2,=#0030      ；剰余を数字に変換
        SUBA    GR4,=1
        ST      GR2, OBUF, GR4
        LD      GR2, GR3        ；被除数 ← 商
        ＜select name =“e”＞
          ＜option value =“0”＞ e ＜/option＞
          ＜option value =“1” ＜?php if（$e == 1）
                          echo “SELECTED”；?＞＞
                        ア  JMI   FIN4＜/option＞
                      :
          ＜option value =“6” ＜?php if（$e == 6）
                          echo “SELECTED”；?＞＞
                        カ  JZE   LOOP4＜/option＞
        ＜/select＞          ；商 = 0 ?
        LD      GR2,=#0020      ；GR2 ← 空白文字
              :
FIN4    OUT     OBUF,OLEN
        RPOP
        RET
OLEN    DC      3
OBUF    DS      3
        END
```

図2.1　HTML 形式で表した副プログラム PRINT

2.2　プログラム内容に基づいた CASL II プログラムの PHP による変換

図2.1における CASL IIのジャンプ命令による繰り返しは，PHP では while 型，または do 〜 while 型の繰り返しを用いる必要がある。また，繰り返しからの脱出の場合(例えば，LOOP4）は，フラグレジスタ（演算結果の状態を表す）の値を条件にした if 文と break 文の組合せが必要になる。シミュレーションで用いる関数は，文献４）を参考に casl2lib.php の中に作成されている。

解答欄ｅにおける条件付きジャンプ命令の解答選択の場合，条件が満たされてループから脱出することができるだけでなく，解答の選択によっては，LOOP4間の無限ループになる場合もあるので，「無限ループ」であることを表示して終了させる必要がある。なお，このシミュレーションは，文献２）のｅラーニングシステム上で一部動作している。

このようなプログラム内容に基づいた変換では，それぞれの CASL IIプログラムを PHP で変換できたとしても，形式的に PHP スクリプトに変換することはできない。したがって，形式的に PHP スクリプトへ変換する方法を次節で述べる。

3．PHP の switch 文を用いた形式的な変換

第2.1節の HTML 形式で表した副プログラム PRINT（数値を10進数の数字列に変換して出力するプログラム）を PHP の switch 文用いたシミュレーションプログラム[2)]を図3.1に示す。プログラム全体を一つの関数 MAIN () で構成する。関数 MAIN () は，while (true) {…} 型の無限ループで構成され，その無限ループの中にシミュレーションプログラム (副プログラム PRINT も含まれる) が switch 文を用いて構成される。プログラムの終わりを示す END 命令に達したとき，脱出用として “break 2；” 文が置かれる。また，番地の記号表現のラベルを変数として扱い，定数を含むデータ語を作り出す Define Constant 命令 DC や連続する記憶領域を設定する Define Storage 命令 DS も変数と同じように，関数 MAIN () における無限ループ while (true) {…} より上部に置かれる。

アセンブラ言語の各行の命令は，下記の副プログラム PRINT と同様に，switch 文の case の各行に置く。ただし，それぞれの行末には break 文を置かない。

3.1　関数 MAIN（）の一部と副プログラム PRINT（一部）の PHP スクリプト

```
//MAIN
function MAIN（）{
  global $pc, $gr, $mem;
  global $a, $b, $c, $d, $e, $f;
    :
  $LOOP4   = 62；
  $FIN4    = 73；
    :
  DC($OLEN, 3)；
  DS ($OBUF, 3)；

  $pc =1； $cnt1=$cnt2=$cnt3=$cnt4=$cnt5=0；
  while（true）{
    switch（$pc）{
//MAIN
      case  1：START（）；
      case  2：CALL（$INPUT, 0)；break；
        :
      case 28：RET（）；break 2；
//END（MAIN）
        :
//PRINT
      case 57：START（）；
      case 58：RPUSH（）；
        :
//LOOP4
      case 62：if（$cnt4＋＋ ＞ 1000）
          {print “無限ループ(PRINT:LOOP4)”；exit；}
            CALL（$DIVIDE, 0)；break；  //数値/10
      case 63：OR2（$gr[2], ‘=#0030’, 0)；
```

```
                                    //剰余を数字に変換
      case 64：SUBA（$gr[4]，‘=1’，0）；
      case 65：ST（$gr[2]，$OBUF，$gr[4]）；
      case 66：LD1（$gr[2]，$gr[3]）；//被除数 ← 商
      case 67：if（$e == 1）{
                 if（JMI（$FIN4，0））break；
               } elseif（$e == 2）{
                   :
               } elseif（$e == 6）{
                 if（JZE（$LOOP4，0））break；
               }
      case 68：LD（$gr[2]，‘=#0020’，0）；
                                    //GR2 ← 空白文字
        :
//FIN4
      case 73：OUT（$OBUF，$OLEN）；
      case 74：RPOP（）；
      case 75：RET（）；break；
//END(PRINT)
    }
  }
}
```

図3.1　PHP で表した副プログラム PRINT

ここで，変数$pc は，プログラムカウンタの役割を果たす。変数$mem は，メモリをシミュレートする配列変数である。変数$gr は汎用レジスタ GR0～GR7までをシミュレートする配列変数である。これらの変数は，casl2lib2.php の中で設定している。

3.2　アセンブラ命令の PHP 関数による表現

図3.1のプログラム中の START（），CALL（…），RET（），RPUSH（），OR2（…），SUBA（…），ST（…），LD1（…），JMI（…），JZE（…），LD（…），OUT（…），RPOP（）等は，CASL II のニーモニック START，CALL，LD，RET，RPUSH，OR，SUBA，ST，LD，JMI，JZE，LD，OUT，RPOP 等を PHP で表した関数である。

また，switch 文の “case 63：OR2（$gr[2]，‘= #0030’，0）；” において，OR2（…）は，CASL II の OR 命令と PHP の OR 演算子名と名前が同じであり，また 2 語命令の OR を表しているため，特に関数名を OR2としている。また，語の番地がリテラルで表されている場合，PHP スクリプトでは文字列 ‘=#0030’ に置き換える。インデックスレジスタ番号が指定されていない場合は，0 に設定する。case 68：行の関数 LD1（…）は，命令 LD の 1 語命令を表し，2 語命令の LD と区別するために関数名を LD1としている。なお，番地の記号表現のラベルは，変数として扱う。ジャンプ命令の例として JMI は，PHP では “if（JMI（$FIN4，0））break；” として表現する。これは，関数 JMI（$FIN4，0）が真のとき，break 文が実行されて分岐する。また，DS（…），DC（…）は，CASL II のアセンブラ Define Storage 命令 DS，アセンブラ Define Constant 命令 DC を PHP で表した関数である。

これらの関数は，文献 4 ）を参考に casl2lib2.php の中に作成されている。第2.2節の casl2lib.php とこの節の casl2lib2.php の違いは casl2lib.php では，プログラムカウンタの役割を果たす変数$pc は利用していないが，casl2lib2.php では，プログラムカウンタの役割を担う変数$pc を積極的に利用している（命令を実行するたびに$pc の値を＋ 1 進めている）。前者の casl2lib.php では，ジャンプ命令を if 文とフラグレジスタを表す変数の値を陽に用いて分岐させている。

3.3　出力結果

図3.2に，第 2 節の問題において，空欄 a ～ f がすべて正解したときのシミュレーション結果を示す。

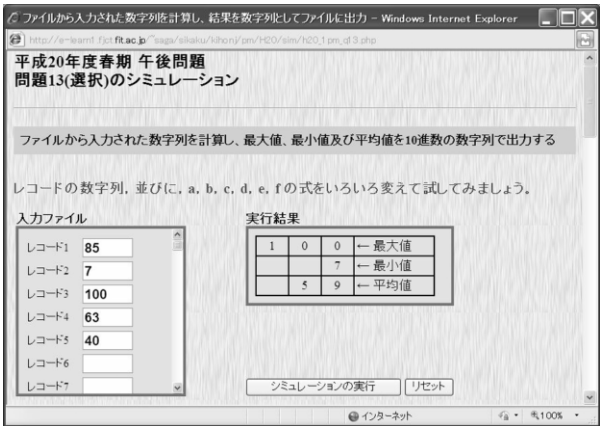

図3.2　出力結果

4．形式的な変換の検討及び例題

4.1　形式的な変換の検討

アセンブラ言語 CASL II プログラムを PHP の switch 文を用いてシミュレーションプログラムの PHP スクリプトを作成したが，CASL II プログラムのラベルは変数として扱っているので，その値を case で指定された定数から（例えば，$LOOP4=62；のように）求める必要がある。したがって，switch 文で CASL II プログラムを作成した後，switch 文中をスキャンしなければ，ラベルを表す変数の値は求まらない。アセンブラ命令の命令 DC の定数，並びにアセンブラ命令 DS の数値が10進数の整数値で与えられているため第3.1節の例では，1 回実行すれば良い。

しかし，文献 5 ）p.260のリスト処理のような場合，図4.1 のように「次のデータアドレス」はラベルで与えられるため，Define Constant 命令 DC のラベル変数の値を図4.2 に示すようにメモリを表す配列変数$mem に設定し直す必要

がある。例えば，図4.2の２行目の DC 命令のデータアドレス$LIST3はこの命令の実行中には決まらないため，10行目の式で５行目の DC 命令で決定した$LIST3の値をメモリ変数$mem に代入している。また，最初のデータアドレスを表す変数$LIST1の値は後で決まるため，最初 printf 関数を用いて$LISTBGN（先頭データの位置を保持）に初期値を設定している。

駅名（６文字）		次のデータアドレス
LIST0	京都	LIST3 番地
LIST1	東京	LIST2 番地
LIST2	名古屋	LIST0 番地
LIST3	大阪	0

図4.1　データ位置情報のリスト構造

```
DC（$LISTBGN, printf（“%s”, $LIST1））；
DC（$LIST0, “‘KYOTO   ’, $LIST3”）；
DC（$LIST1, “‘TOKYO   ’, $LIST2”）；
DC（$LIST2, “‘NAGOYA’, $LIST0”）；
DC（$LIST3, “‘OSAKA   ’, 0”）；
DC（$N6,6）；

$mem［$LISTBGN］=$LIST1；
$mem［$LIST0＋$mem［$N6］］=$LIST3；
$mem［$LIST1＋$mem［$N6］］=$LIST2；
$mem［$LIST2＋$mem［$N6］］=$LIST0；
```

図4.2　リスト処理の DC 命令

4.2　例題

基本情報技術者試験【平成10年度秋期午前問題】問13の問題を CASL II でプログラミングし，それを PHP スクリプトに変換してシミュレートする。解答群ア～エからの選択は，下図に示す if 文の$a の１～４で選択する。

```
//FUNC_MAIN
function MAIN（）{
  global $pc, $gr, $mem, $a；

  $LOOP1=4；
  $LOOP2=5；
  DC（$HEAD, printf（“%s”,$POINTER0））；
  DC（$POINTER0, “‘TOKYO              ’,
                                  $POINTER2”）；
  DC（$POINTER1, “‘NAGOYA             ’,
                                        0”）；
  DC（$POINTER2, “‘SHIN-YOKOHAMA’,
                                  $POINTER4”）；
  DC（$POINTER3, “‘HAMAMATU       ’,
                                  $POINTER1”）；
  DC（$POINTER4, “‘ATAMI              ’,
                                  $POINTER3”）；
  DC（$POINTER5, “‘SHIZUOKA           ’,
                                    $TMP”）；
  DC（$N, 13）；
  DS（$OBUF, 256）；

  if （$a ==1）{
    $TMP =$POINTER2；$POINTER1=$POINTER5；
  } elseif（$a ==2）{
    $TMP =$POINTER3；$POINTER3=$POINTER5；
  } elseif（$a ==3）{
    $TMP =$POINTER4；$POINTER1=$POINTER5；
  } elseif（$a ==4）{
    $TMP =$POINTER5；$POINTER3=$POINTER4；
  }

  $mem［$HEAD］=$POINTER0；
  $mem［$POINTER0＋$mem［$N］］=$POINTER2；
  $mem［$POINTER2＋$mem［$N］］=$POINTER4；
  $mem［$POINTER3＋$mem［$N］］=$POINTER1；
  $mem［$POINTER4＋$mem［$N］］=$POINTER3；
  $mem［$POINTER5＋$mem［$N］］=$TMP；

  $pc =1；$cnt =0；
  while（true）{
    if（$cnt＋＋＞1000）{print “無限ループ”；exit；}
    switch（$pc）{
      case  1：START（）；
      case  2：LAD（$gr[1], 1, 0）；
      case  3：LD（$gr[2], $HEAD, 0）；
//LOOP1
      case  4：LAD（$gr[3], 0, 0）；
//LOOP2
      case  5：LD（$gr[0], 0, $gr[2]）；
      case  6：ST（$gr[0], $OBUF, $gr[3]）；
      case  7：ADDL1（$gr[2], $gr[1]）；
      case  8：ADDA1（$gr[3], $gr[1]）；
      case  9：CPA（$gr[3], $N, 0）；
      case 10：if（JMI（$LOOP2, 0））  break;
      case 11：OUT1（$OBUF, $N)；
      case 12：LD（$gr[2], 0, $gr[2]）；
      case 13：if（JNZ（$LOOP1, 0））break;
      case 14：RET（）；
      default：break 2；
    }
  }
}
```

図4.3　例題の PHP スクリプト

関数 ADDL1（…），ADDA1（…）は，命令 ADDL ，ADDA の 1 語命令を表し，2 語命令の ADDL，ADDA とを区別するために関数名を ADDL1，ADDA1としている。また，OUT1（…）は，図3.1の OUT（…）と区別するために関数名を OUT1としている。関数 OUT1（…）は表作成はできないが，OUT（…）は表作成ができるようになっている。

5．おわりに

本論文では，アセンブラ言語 CASL II プログラムを PHP の switch 文を用いることによって形式的に PHP スクリプトに変換できることを示し，並びに CASL II プログラムの簡易シミュレーションシステムについての報告を行った。この簡易シミュレーションシステムは，問題で解答群の命令を選択して実行し，その場で結果を視覚的に確認することができるので，利用者にとって利便性が高いと思われる。

今後の課題としては，ユーザが入力した CASL II プログラムについてもシミュレートできるようにすることが必要と考えられる。

参考文献

1）http://www.ipa.go.jp/
2）http://e-learn1.fjct.fit.ac.jp/~saga/sikaku/
3）ノマド・ワークス：かんたん合格　基本情報技術者問題集　平成21年度春期，2008年，インプレスコミュニケーションズ.
4）玉井 浩：COMET II＆CASL II，2001年，エスアイビー・アクセス.
5）赤松 徹：基本情報技術者試験 CASL 集中ゼミ，2000年，ソフトバンク　パブリッシング.