コンパイラレポート - Pascal-T の挙動を調べる

7JFC1121 佐藤圭一

§1：目的

パスカルコンパイラのサブセット版「Pascal-T(PASCAL-TINY)」がいったい何をやっているのか分析してみる。

§2：挙動を調べるに当たって私が取ったアプローチ

Pascal-T は、DEC 社 PDP-11 上で動作する様設計されている。
が、生憎私は PDP-11 を所持していない。
よって、PDP-11 アセンブラ言語から日本語で「何をやっているか」を解析変換表示するプログラムを製作し、それにより Pascal-T の挙動を調べていくことにした。

・私の環境では、繰り返し文の生成にランタイムエラーが発生してしまい、正常に動作しなかった。

§３：各プログラムの検証

・内容の無いプログラム

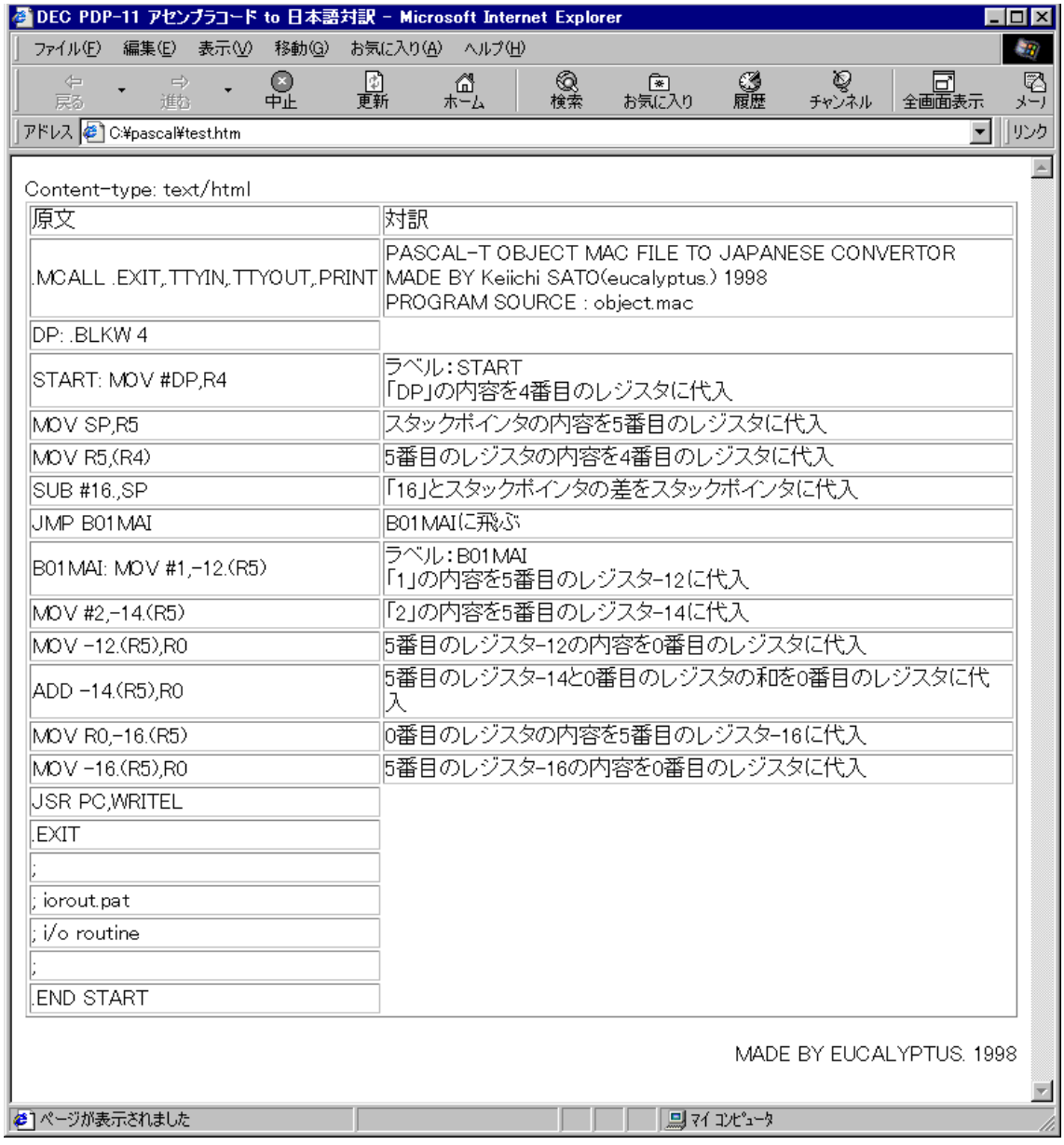
```
PROGRAM SAMPLE01(INPUT, OUTPUT);
BEGIN
END.
```

・『B01MAI』が作られない　＞　暴走することが分かる。

・単純な四則演算

```
PROGRAM SAMPLE02(INPUT, OUTPUT);
VAR X , Y , Z : INTEGER;

BEGIN
   X := 1;
   Y := 2;
   Z := X + Y;
   WRITELN(Z);
END.
```

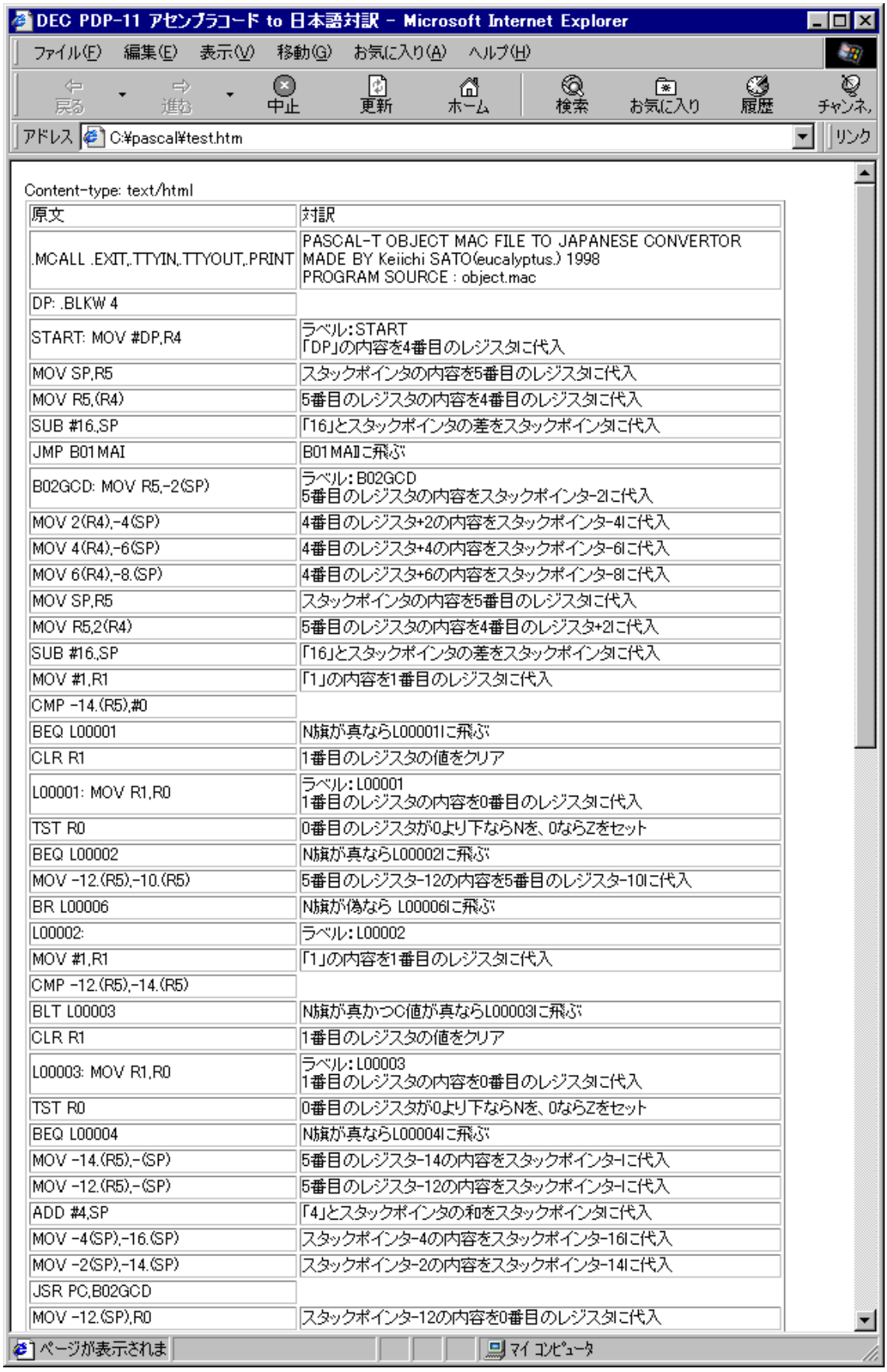


原文	対訳
.MCALL .EXIT,.TTYIN,.TTYOUT,.PRINT	PASCAL-T OBJECT MAC FILE TO JAPANESE CONVERTOR MADE BY Keiichi SATO(eucalyptus.) 1998 PROGRAM SOURCE : object.mac
DP: .BLKW 4	
START: MOV #DP,R4	ラベル：START 「DP」の内容を4番目のレジスタに代入
MOV SP,R5	スタックポインタの内容を5番目のレジスタに代入
MOV R5,(R4)	5番目のレジスタの内容を4番目のレジスタに代入
SUB #16.,SP	「16」とスタックポインタの差をスタックポインタに代入
JMP B01MAI	B01MAIに飛ぶ
B01MAI: MOV #1,-12.(R5)	ラベル：B01MAI 「1」の内容を5番目のレジスタ-12に代入
MOV #2,-14.(R5)	「2」の内容を5番目のレジスタ-14に代入
MOV -12.(R5),R0	5番目のレジスタ-12の内容を0番目のレジスタに代入
ADD -14.(R5),R0	5番目のレジスタ-14と0番目のレジスタの和を0番目のレジスタに代入
MOV R0,-16.(R5)	0番目のレジスタの内容を5番目のレジスタ-16に代入
MOV -16.(R5),R0	5番目のレジスタ-16の内容を0番目のレジスタに代入
JSR PC,WRITEL	
.EXIT	
;	
; iorout.pat	
; i/o routine	
;	
.END START	

一応四則演算になっている。

・サンプルプログラム

・PASCAL-T のサンプルプログラムのソースをそのまま解析してみた



Content-type: text/html

原文	対訳
.MCALL .EXIT,.TTYIN,.TTYOUT,.PRINT	PASCAL-T OBJECT MAC FILE TO JAPANESE CONVERTOR MADE BY Keiichi SATO(eucalyptus.) 1998 PROGRAM SOURCE : object.mac
DP: .BLKW 4	
START: MOV #DP,R4	ラベル:START 「DP」の内容を4番目のレジスタに代入
MOV SP,R5	スタックポインタの内容を5番目のレジスタに代入
MOV R5,(R4)	5番目のレジスタの内容を4番目のレジスタに代入
SUB #16.,SP	「16」とスタックポインタの差をスタックポインタに代入
JMP B01MAI	B01MAIに飛ぶ
B02GCD: MOV R5,-2(SP)	ラベル:B02GCD 5番目のレジスタの内容をスタックポインター2に代入
MOV 2(R4),-4(SP)	4番目のレジスタ+2の内容をスタックポインター4に代入
MOV 4(R4),-6(SP)	4番目のレジスタ+4の内容をスタックポインター6に代入
MOV 6(R4),-8.(SP)	4番目のレジスタ+6の内容をスタックポインター8に代入
MOV SP,R5	スタックポインタの内容を5番目のレジスタに代入
MOV R5,2(R4)	5番目のレジスタの内容を4番目のレジスタ+2に代入
SUB #16.,SP	「16」とスタックポインタの差をスタックポインタに代入
MOV #1,R1	「1」の内容を1番目のレジスタに代入
CMP -14.(R5),#0	
BEQ L00001	N旗が真ならL00001に飛ぶ
CLR R1	1番目のレジスタの値をクリア
L00001: MOV R1,R0	ラベル:L00001 1番目のレジスタの内容を0番目のレジスタに代入
TST R0	0番目のレジスタが0より下ならNを、0ならZをセット
BEQ L00002	N旗が真ならL00002に飛ぶ
MOV -12.(R5),-10.(R5)	5番目のレジスター12の内容を5番目のレジスター10に代入
BR L00006	N旗が偽なら L00006に飛ぶ
L00002:	ラベル:L00002
MOV #1,R1	「1」の内容を1番目のレジスタに代入
CMP -12.(R5),-14.(R5)	
BLT L00003	N旗が真かつC値が真ならL00003に飛ぶ
CLR R1	1番目のレジスタの値をクリア
L00003: MOV R1,R0	ラベル:L00003 1番目のレジスタの内容を0番目のレジスタに代入
TST R0	0番目のレジスタが0より下ならNを、0ならZをセット
BEQ L00004	N旗が真ならL00004に飛ぶ
MOV -14.(R5),-(SP)	5番目のレジスター14の内容をスタックポインターに代入
MOV -12.(R5),-(SP)	5番目のレジスター12の内容をスタックポインターに代入
ADD #4,SP	「4」とスタックポインタの和をスタックポインタに代入
MOV -4(SP),-16.(SP)	スタックポインター4の内容をスタックポインター16に代入
MOV -2(SP),-14.(SP)	スタックポインター2の内容をスタックポインター14に代入
JSR PC,B02GCD	
MOV -12.(SP),R0	スタックポインター12の内容を0番目のレジスタに代入

MOV R0,-10.(R5)	0番目のレジスタの内容を5番目のレジスタ-10に代入
BR L00005	N旗が偽なら L00005に飛ぶ
L00004:	ラベル：L00004
MOV -12.(R5),R1	5番目のレジスタ-12の内容を1番目のレジスタに代入
CLR R0	0番目のレジスタの値をクリア
DIV -14.(R5),R0	5番目のレジスタ-14と0番目のレジスタの商を0番目のレジスタに代入
MUL -14.(R5),R0	5番目のレジスタ-14と0番目のレジスタの積を0番目のレジスタに代入
MOV R1,R0	1番目のレジスタの内容を0番目のレジスタに代入
MOV R0,R1	0番目のレジスタの内容を1番目のレジスタに代入
MOV -12.(R5),R0	5番目のレジスタ-12の内容を0番目のレジスタに代入
SUB R1,R0	1番目のレジスタと0番目のレジスタの差を0番目のレジスタに代入
MOV R0,-16.(R5)	0番目のレジスタの内容を5番目のレジスタ-16に代入
MOV -14.(R5),-(SP)	5番目のレジスタ-14の内容をスタックポインターに代入
MOV -16.(R5),-(SP)	5番目のレジスタ-16の内容をスタックポインターに代入
ADD #4,SP	「4」とスタックポインタの和をスタックポインタに代入
MOV -4(SP),-16.(SP)	スタックポインター4の内容をスタックポインター16に代入
MOV -2(SP),-14.(SP)	スタックポインター2の内容をスタックポインター14に代入
JSR PC,B02GCD	
MOV -12.(SP),R0	スタックポインター12の内容を0番目のレジスタに代入
MOV R0,-10.(R5)	0番目のレジスタの内容を5番目のレジスタ-10に代入
L00005:	ラベル：L00005
L00006:	ラベル：L00006
MOV -4(R5),2(R4)	5番目のレジスタ-4の内容を4番目のレジスタ+2に代入
MOV -6(R5),4(R4)	5番目のレジスタ-6の内容を4番目のレジスタ+4に代入
MOV -8.(R5),6(R4)	5番目のレジスタ-8の内容を4番目のレジスタ+6に代入
MOV R5,SP	5番目のレジスタの内容をスタックポインタに代入
MOV -2(SP),R5	スタックポインター2の内容を5番目のレジスタに代入
RTS PC	
B01MAI: JSR PC,READLN	ラベル：B01MAI
MOV R0,-12.(R5)	0番目のレジスタの内容を5番目のレジスタ-12に代入
JSR PC,READLN	
MOV R0,-14.(R5)	0番目のレジスタの内容を5番目のレジスタ-14に代入
MOV -12.(R5),-(SP)	5番目のレジスタ-12の内容をスタックポインターに代入
MOV -14.(R5),-(SP)	5番目のレジスタ-14の内容をスタックポインターに代入
ADD #4,SP	「4」とスタックポインタの和をスタックポインタに代入
MOV -4(SP),-16.(SP)	スタックポインター4の内容をスタックポインター16に代入
MOV -2(SP),-14.(SP)	スタックポインター2の内容をスタックポインター14に代入
JSR PC,B02GCD	
MOV -12.(SP),R0	スタックポインター12の内容を0番目のレジスタに代入
MOV R0,-16.(R5)	0番目のレジスタの内容を5番目のレジスタ-16に代入
MOV -16.(R5),R0	5番目のレジスタ-16の内容を0番目のレジスタに代入
JSR PC,WR	

MADE BY EUCALYPTUS. 1998

§４：結果の分析および感想

PDP-１１システムは、CASL(COMET)のレジスタが 4 から6に増えたものと同等であると解釈できる。

よって、低級言語である以上、「箱の中にいくつおはじきが入って・・・」というような演算になっているのも頷ける。

今回、簡単な解釈機能を持ったプログラムを製作した。もう少し頑張れば仮想 PDP-11 が開発できたのだが、時間的余裕が無かったこと、ジャンプ命令があるために単純なインタプリタでは実行できないことなどから断念した。

§５：おまけ　-PDP-11 アセンブラを日本語で注釈を加えるプログラム

今回のレポートのために、日本語でアセンブラに注釈を加えるプログラムを開発した。
開発言語は jperl5.004 であり、デバッグは ActivePerl Ver5 上で行った。
このプログラムは、ファイルを引数として与えると、標準出力にハイパーテキスト形式で注釈付きデータを返す。

-Source(mac2jpn.pl)

```
#!/usr/local/bin/perl

#PASCAL-T OBJECT MAC FILE TO JAPANESE CONVERTOR
#MADE BY Keiichi SATO    (eucalyptus.) 1998

#USAGE:mac2jpn.pl objectmacrofile

#ファイルの読み込み/エラー処理
if (!open(INPUT,"$ARGV[0]")){
   print "ERROR      :ファイルが見つかりません。¥n";
   print "使用方法：mac2jpn.pl PASCAL-T が吐き出したソース¥n";
   exit;
   }
@SOURCE_FILE = <INPUT>;
close(IN);

print "Content-type: text/html¥n¥n";
print "<html><title>DEC PDP-11  アセンブラコード to  日本語対訳</title><body>¥n";
print "<table border=¥"1¥"><tr><td>原文</td><td>対訳</td></tr>¥n";

#読みこまれた文字列を一行ずつ解析
foreach $line (@SOURCE_FILE) {
#改行文字を落とす
   chop $line;

#原文出力
print "<tr><td>$line</td><td>";
#最初の2行は飛ばす(ついでに info 出力)
   $temp = index ("$line",".MCALL");
   if ($temp ne "-1"){&init_out;next;}
   $temp = index ("$line",".BLKW");
   if ($temp ne "-1"){next;}

#ラベル部分感知
   $temp = index("$line",": ");
   if ("$temp" ne "-1"){
      ($label,$temp)=split(/:/,$line);
      print "ラベル：$label<br>¥n";
#ラベルを消す:)
      $temp = length($label);
      $temp = $temp + 2;
      for(0..$temp){
         $line = " $line";
      }

   }

#命令解析
   ($temp,$line)=split(/              /,$line);
   ($op,$line)=split(/         /,$line);

#レジスタ、データ解析
#レジスタ、データが2つあるかどうか？
   $temp_ = index("$line",",");
   if ("$temp_" ne "-1"){
      ($d1,$d2)=split(/¥,/,$line);
      $r_input = "$d1"; &r_sym; $d1 = "$r_sum";
```

```
      $r_input = "$d2"; &r_sym; $d2 = "$r_sum";
   }
   if ("$temp_" eq "-1"){
      $d1 = "$line";
      $r_input = "$d1"; &r_sym; $d1 = "$r_sum";
   }
   {&op_out;}

#あとしまつ
print "</td></tr>¥n";
}
print "</table><br>¥n";
print "<div align=¥"right¥">MADE BY EUCALYPTUS. 1998</div></body></html>¥n";
exit;

###サブルーチン

#info 出力
sub init_out{
   print "PASCAL-T OBJECT MAC FILE TO JAPANESE CONVERTOR<br>¥n";
   print "MADE BY Keiichi SATO(eucalyptus.) 1998<br>¥n";
   print "PROGRAM SOURCE : ";
   print $ARGV[0];
   print "<br>";
}

#なぁんちゃって構文解析
sub r_sym{
#データ直指定判断
   $temp = index("$r_input","#");
   if ("$temp" ne "-1"){
      ($temp2,$temp3)=split(/,/,$r_input);
      ($temp4,$temp2)=split(/#/,$r_input);
      $temp = index("$r_input",".");
      if ("$temp" ne "-1"){chop($temp2)}
      $r_sum = "「$temp2」";
      return;
   }
#演算付きの代入判断
   $temp = index("$r_input","(");
   if ("$temp" ne "-1"){
      ($temp2,$temp4)=split(/¥(/,$r_input);
      ($temp5,$temp3)=split(/¥)/,$temp4);
      $temp = index("$temp2",".");
      if ("$temp" ne "-1"){
         chop($temp2);
      }
      if ("$temp5" eq "R0"){$temp5 = "0 番目のレジスタ";}
      if ("$temp5" eq "R1"){$temp5 = "1 番目のレジスタ";}
      if ("$temp5" eq "R2"){$temp5 = "2 番目のレジスタ";}
      if ("$temp5" eq "R3"){$temp5 = "3 番目のレジスタ";}
      if ("$temp5" eq "R4"){$temp5 = "4 番目のレジスタ";}
      if ("$temp5" eq "R5"){$temp5 = "5 番目のレジスタ";}
      if ("$temp5" eq "SP"){$temp5 = "スタックポインタ";}
      if ("$temp5" eq "PC"){$temp5 = "プログラムカウンタ";}
      if ("$temp2" eq ""){
         $r_sum = "$temp5";}
      if ("$temp2" ne ""){
         $temp = index("$temp2","-");
         if ("$temp" eq "-1"){
            $r_sum = "$temp5+$temp2";
         }
         if ("$temp" ne "-1"){
            $r_sum = "$temp5$temp2";
         }
      }
      return;
   }

#そのほか判断
   if ($r_input eq "R0"){$r_sum = "0 番目のレジスタ";return;}
   if ($r_input eq "R1"){$r_sum = "1 番目のレジスタ";return;}
   if ($r_input eq "R2"){$r_sum = "2 番目のレジスタ";return;}
```

```
    if ($r_input eq "R3"){$r_sum = "3 番目のレジスタ";return;}
    if ($r_input eq "R4"){$r_sum = "4 番目のレジスタ";return;}
    if ($r_input eq "R5"){$r_sum = "5 番目のレジスタ";return;}
    if ($r_input eq "SP"){$r_sum = "スタックポインタ";return;}
    if ($r_input eq "PC"){$r_sum = "プログラムカウンタ";return;}

#どーしよーもないときは、そのまま返す
    $r_sum = "$r_input";
}


#命令解釈/出力
sub op_out{

#MOV 命令
    $temp = index("$op","MOV");
    if ($temp ne "-1"){
        print "$d1 の内容を$d2 に代入";
        return;
    }

#CLR 命令
    $temp = index("$op","CLR");
    if ($temp ne "-1"){
        print "$d1 の値をクリア";
        return;
    }

#BIC 命令
    $temp = index("$op","BIC");
    if ($temp ne "-1"){
        print "$d1 と$d2 の論理積を$d2 に代入";
        return;
    }

#BIS 命令
    $temp = index("$op","BIS");
    if ($temp ne "-1"){
        print "$d1 と$d2 の論理和を$d2 に代入";
        return;
    }

#XOR 命令
    $temp = index("$op","XOR");
    if ($temp ne "-1"){
        print "$d1 と$d2 の排他的論理和を$d2 に代入";
        return;
    }

#NEG 命令
    $temp = index("$op","NEG");
    if ($temp ne "-1"){
        print "$d1 の値を反転";
        return;
    }

#ADD 命令
    $temp = index("$op","ADD");
    if ($temp ne "-1"){
        print "$d1 と$d2 の和を$d2 に代入";
        return;
    }

#SUB 命令
    $temp = index("$op","SUB");
    if ($temp ne "-1"){
        print "$d1 と$d2 の差を$d2 に代入";
        return;
    }

#MUL 命令
    $temp = index("$op","MUL");
    if ($temp ne "-1"){
```

```
    print "$d1 と$d2 の積を$d2 に代入";
    return;
  }

#DIV 命令
  $temp = index("$op","DIV");
  if ($temp ne "-1"){
    print "$d1 と$d2 の商を$d2 に代入";
    return;
  }

#TST 命令
  $temp = index("$op","TST");
  if ($temp ne "-1"){
    print "$d1 が 0 より下なら N を、0 なら Z をｾｯﾄ";
    return;
  }

#CMP 命令
  $temp = index("$op","TST");
  if ($temp ne "-1"){
    print "($d1)- ($d2)  が 0 より下なら N を、0 なら Z をｾｯﾄ";
    return;
  }

#BEQ 命令
  $temp = index("$op","BEQ");
  if ($temp ne "-1"){
    print "N 旗が真なら$d1 に飛ぶ";
    return;
  }

#BNE 命令
  $temp = index("$op","BBE");
  if ($temp ne "-1"){
    print "N 旗が偽なら$d1 に飛ぶ";
    return;
  }

#BLT 命令
  $temp = index("$op","BLT");
  if ($temp ne "-1"){
    print "N 旗が真かつ C 値が真なら$d1 に飛ぶ";
    return;
  }

#BLE 命令
  $temp = index("$op","BLE");
  if ($temp ne "-1"){
    print "N 旗が真かつ C 値が偽なら$d1 に飛ぶ";
    return;
  }

#BGT 命令
  $temp = index("$op","BGT");
  if ($temp ne "-1"){
    print "N 旗が偽かつ C 値が真なら$d1 に飛ぶ";
    return;
  }

#BGE 命令
  $temp = index("$op","BGE");
  if ($temp ne "-1"){
    print "N 旗が偽かつ C 値が偽なら$d1 に飛ぶ";
    return;
  }

#BR 命令
  $temp = index("$op","BR");
  if ($temp ne "-1"){
    print "N 旗が偽なら$d1 に飛ぶ";
    return;
  }
```

```
#JMP 命令
   $temp = index("$op","JMP");
   if ($temp ne "-1"){
      print "$d1 に飛ぶ";
      return;
   }
}
```