1章

【1.1】

```
module f(x,y,z,s,c);

  input x,y,z;     // 入力ポートx,y,z
  output s,c;      // 出力ポートs,c
  wire u,v,w;      // 配線u,v,w


  assign u=x&y;
  assign u=x&y;
  assign v=x&~z;
  assign w=y|z;
  assign s=u|v;
  assign c=~(v&w);

endmodule
```

【1.2】

```
module inc(x,s);

  input [3:0] x;  // 4ビットの入力ポートx
  output [4:0] s; // 5ビットの出力ポートs
  wire [2:0] c;   // 5ビットの配線c

  ha ha0(x[0],1'b1,s[0],c[0]);
  ha ha1(x[1],c[0],s[1],c[1]);
  ha ha2(x[2],c[1],s[2],c[2]);
  ha ha3(x[3],c[2],s[3],s[4]);

endmodule
```

【1.3】

```
module inc(x,s);
  parameter N=8;

  input [N-1:0] x;  // Nビットの入力ポートx
  output [N:0] s;   // N+1ビットの出力ポートs
  wire [N-2:0] c;   // N-1ビットの配線c

  ha ha0(x[0],1'b1,s[0],c[0]);

  genvar i;
  for(i=1;i<N-1;i=i+1)
    ha ha1(x[i],c[i-1],s[i],c[i]);

  ha ha2(x[N-1],c[N-2],s[N-1],s[N]);

endmodule
```

【1.4】

```
module adder(x,y,s);
  parameter N = 8;     // 既定値8のパラメタ

  input [N-1:0] x,y;   // Nビットの入力ポートx,y
  output [N:0] s;      // N+1ビットの出力ポートs
  wire [N-2:0] c;      // N-1ビットの配線c

  fa fa0(.x(1'b0),.y(x[0]),.z(y[0]),.s(s[0]),.c(c
      [0]));

  genvar i;
  for(i=1;i<=N-2;i=i+1)
    fa fa1(.x(c[i-1]),.y(x[i]),.z(y[i]),.s(s[i]),.c
        (c[i]));

  fa fa2(.x(c[N-2]),.y(x[N-1]),.z(y[N-1]),.s(s[N
      -1]),.c(s[N]));

endmodule
```

【1.5】

(1)

$0 \leq x \leq 15$ より, 素数となる x は 2, 3, 5, 7, 11, 13 で, 真理値表で表すと以下の表になる。

x_3	x_2	x_1	x_0	$p(x)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

真理値表より, $p(x)$ を積和形の式であらわすと

$\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}+\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot x_0+\overline{x_3}\cdot x_2\cdot\overline{x_1}\cdot x_0+\overline{x_3}\cdot x_2\cdot x_1\cdot x_0+x_3\cdot\overline{x_2}\cdot x_1\cdot x_0+x_3\cdot x_2\cdot\overline{x_1}\cdot x_0$

となる. この式をカルノー図より簡単化すると

$$\overline{x_3}\cdot\overline{x_2}\cdot x_1+\overline{x_3}\cdot x_1\cdot x_0+x_2\cdot\overline{x_1}\cdot x_0+\overline{x_2}\cdot x_1\cdot x_0$$

となる。(別解, $\overline{x_3}\cdot\overline{x_2}\cdot x_1+\overline{x_3}\cdot x_2\cdot x_0+x_2\cdot\overline{x_1}\cdot x_0+\overline{x_2}\cdot x_1\cdot x_0$)

```
module p1(x,p);

  input [3:0] x;  // 4ビットの入力ポートx
  output p;       // 1ビットの出力ポートp

  assign p = (~x[3])&(~x[2])&x[1]
           | (~x[3])&x[1]&x[0]
           | x[2]&(~x[1])&x[0]
           | (~x[2])&x[1]&x[0];

endmodule
```

(2)

カルノー図より最も簡単な和積形の式は

$$(x_2+x_1)(\overline{x_2}+x_0)(\overline{x_3}+x_0)(\overline{x_3}+\overline{x_2}+\overline{x_1})$$

となる。

```
module p2(x,p);

  input [3:0] x;  // 4ビットの入力ポートx
  output p;       // 1ビットの出力ポートp

  assign p = (x[2] | x[1])
           & ((~x[2]) | x[0])
           & ((~x[3]) | x[0])
           & ((~x[3]) | (~x[2]) | (~x[1]));

endmodule
```

(3)

```
module p_tb;

  reg [3:0] x;     // 4ビットの変数x
  wire q1, q2;     // 1ビットの配線q1, q2

  p1 p1_0(x,q1);
  p2 p2_0(x,q2);

  initial begin
    $dumpvars;
    x = 0; #100
    x = 1; #100
    x = 2; #100
    x = 3; #100
    x = 4; #100
    x = 5; #100
    x = 6; #100
    x = 7; #100
    x = 8; #100
    x = 9; #100
    x = 10; #100
    x = 11; #100
    x = 12; #100
    x = 13; #100
    x = 14; #100
    x = 15; #100
    $finish;
  end

endmodule
```

【1.6】

```
module seg7_tb;

  reg [3:0] s;  // 4ビットの変数s
  wire [0:6] x; // 7ビットの配線x

  seg7 seg7_0(s,x);

  initial begin
    $dumpvars;
    s = 0; #100
    s = 1; #100
    s = 2; #100
    s = 3; #100
    s = 4; #100
    s = 5; #100
    s = 6; #100
    s = 7; #100
    s = 8; #100
    s = 9; #100
    $finish;
  end

endmodule
```

【1.8】

```
module selector8(s,a,x);

  input [2:0] s;  // 3ビットの入力ポートs
  input [0:7] a;  // 8ビットの入力ポートa
  output x;       // 1ビットの出力ポートx
  wire [0:3] c0;  // 4ビットの配線c0
  wire [0:1] c1;  // 2ビットの配線c1

  selector2 selector2_00(.s(s[0]),.a(a[0:1]),.x(c0
     [0]));
  selector2 selector2_01(.s(s[0]),.a(a[2:3]),.x(c0
     [1]));
  selector2 selector2_02(.s(s[0]),.a(a[4:5]),.x(c0
     [2]));
  selector2 selector2_03(.s(s[0]),.a(a[6:7]),.x(c0
     [3]));

  selector2 selector2_10(.s(s[1]),.a(c0[0:1]),.x(c1
     [0]));
  selector2 selector2_11(.s(s[1]),.a(c0[2:3]),.x(c1
     [1]));

  selector2 selector2_2(.s(s[2]),.a(c1[0:1]),.x(x
     ));

endmodule
```

【1.10】

alu_d.v

```
‘define ADD     5’b00000
‘define SUB     5’b00001
‘define MUL     5’b00010
‘define SHL     5’b00011
‘define SHR     5’b00100
‘define BAND    5’b00101
‘define BOR     5’b00110
‘define BXOR    5’b00111
‘define AND     5’b01000
‘define OR      5’b01001
‘define EQ      5’b01010
‘define NE      5’b01011
‘define GE      5’b01100
‘define LE      5’b01101
‘define GT      5’b01110
‘define LT      5’b01111
‘define NEG     5’b10000
‘define BNOT    5’b10001
‘define NOT     5’b10010
‘define SQR     5’b11011
‘define ABS     5’b11100
‘define INC     5’b11101
‘define DEC     5’b11110
```

alu.v

```
‘include "alu_d.v"
module alu(a,b,f,s);

  input signed [15:0] a,b;    // 16ビットの入力ポー
     トa,b
  input [4:0] f;  // 5ビットの入力ポートf
  output signed [15:0] s; // 16ビットの出力ポートs
  reg signed [15:0] s; // 16ビットの変数s

  always @(a,b,f) // a,b,fが変化したときに実行
    case(f) // fの値で演算が決定
      `ADD : s = b + a;
      `SUB : s = b - a;
      `MUL : s = b * a;
      `SHL : s = b << a;
      `SHR : s = b >> a;
      `BAND: s = b & a;
      `BOR : s = b | a;
      `BXOR: s = b ^ a;
      `AND : s = b && a;
      `OR  : s = b || a;
      `EQ  : s = b == a;
      `NE  : s = b != a;
      `GE  : s = b >= a;
      `LE  : s = b <= a;
      `GT  : s = b > a;
      `LT  : s = b < a;
      `NEG : s = -a;
      `BNOT: s = ~a;
      `NOT : s = !a;
      `SQR : s = a * a;
      `ABS : s = (a>=0) ? a : -a;
      `INC : s = a + 1;
      `DEC : s = a - 1;
      default : s = 16'hXXXX; //
          fが範囲外のときは不定値
    endcase

endmodule
```

alu_tb.v

```
`include "alu_d.v"
module alu_tb;

  reg signed [15:0] a,b;  // 16ビットの変数a,b
  reg [4:0] f;    // 4ビットの変数f
  wire signed [15:0] s;   // 16ビットの配線s

  alu alu_0(a,b,f,s);  // aluのインスタンス化

  initial begin
    $dumpvars;
    a = 5;  f=`SQR;  #100   // fの値を変える
    a = -5; f=`SQR;  #100
    a = 5;  f=`ABS;  #100   // fの値を変える
    a = -5; f=`ABS;  #100
    a = 5;  f=`INC;  #100   // fの値を変える
    a = -5; f=`INC;  #100
    a = 5;  f=`DEC;  #100   // fの値を変える
    a = -5; f=`DEC;  #100
    $finish;
  end

endmodule
```

【1.11】

(1)

```
module decoder1(s,e,d);

  input s;
  input e;
  output [0:1] d;

  assign d[0] = (s==1'b0) ? e : 1'b0;
  assign d[1] = (s==1'b1) ? e : 1'b0;

endmodule
```

(2)

```
module decoder2(s,e,d);

  input [1:0] s;
  input e;
  output [0:3] d;
  wire [0:1] c;

  decoder1 decoder1_0(s[1],e,c);
  decoder1 decoder1_1(s[0],c[0],d[0:1]);
  decoder1 decoder1_2(s[0],c[1],d[2:3]);

endmodule
```

(3)

```
module decoder3(s,e,d);

  input [2:0] s;
  input e;
  output [0:7] d;
  wire [0:1] c0;
  wire [0:1] c1;
  wire [0:1] c2;

  decoder1 decoder1_0(s[2],e,c0);
  decoder1 decoder1_1(s[1],c0[0],c1);
  decoder1 decoder1_2(s[1],c0[1],c2);
  decoder1 decoder1_3(s[0],c1[0],d[0:1]);
  decoder1 decoder1_4(s[0],c1[1],d[2:3]);
  decoder1 decoder1_5(s[0],c2[0],d[4:5]);
  decoder1 decoder1_6(s[0],c2[1],d[6:7]);

endmodule
```

(4)

```
module decoder(s,e,d);
  parameter M=4;
  parameter N=2**M;

  input [M-1:0] s;
  input e;
  output [0:N-1] d;

  genvar i;
  for(i=0;i<N;i=i+1)
    assign d[i] = (s==i) ? e : 1'b0;

endmodule
```

2章

【2.1】

```
module counter(clk,rst_n,load,inc,d,q,zero,dec);
  parameter N=16; // 既定値16のビット数N

  input clk,rst_n,load,inc,zero,dec;   // 1ビットの入力ポート
  input [N-1:0] d;     // Nビットの入力ポートd
  output [N-1:0] q;    // Nビットの出力ポートq
  reg [N-1:0] q;       // Nビットの変数q

  always @(posedge clk, negedge rst_n)
    if(!rst_n) q<=0;     // 非同期リセット
    else if(load) q<=d; // dの書き込み
    else if(inc) q<=q+1;     // qが1増える
    else if(zero) q<=0; // qが0に
    else if(dec) q<=q-1;     // qが1減る

endmodule
```

【2.2】

```
module counter4(clk,rst_n,load,inc,d,q);

  input clk,rst_n,load,inc;   // 1ビットの入力ポート
  input [3:0] d;    // 4ビットの入力ポートd
  output [3:0] q;   // 4ビットの出力ポートq
  wire [2:0] c;      // 3ビットの配線c
  wire [4:0] s;    // 5ビットの配線s
  wire [3:0] x;    // 4ビットの配線x

  ha ha0(q[0],1'b1,s[0],c[0]);
  ha ha1(q[1],c[0],s[1],c[1]);
  ha ha2(q[2],c[1],s[2],c[2]);
  ha ha3(q[3],c[2],s[3],s[4]);

  selector2 selector2_0(load,{s[0],d[0]},x[0]);
  selector2 selector2_1(load,{s[1],d[1]},x[1]);
  selector2 selector2_2(load,{s[2],d[2]},x[2]);
  selector2 selector2_3(load,{s[3],d[3]},x[3]);

  ff ff0(clk,1'b1,rst_n,load|inc,x[0],q[0]);
  ff ff1(clk,1'b1,rst_n,load|inc,x[1],q[1]);
  ff ff2(clk,1'b1,rst_n,load|inc,x[2],q[2]);
  ff ff3(clk,1'b1,rst_n,load|inc,x[3],q[3]);

endmodule
```

【2.4】

```
‘define A 3’b000
‘define B 3’b001
‘define C 3’b010
‘define D 3’b011
‘define E 3’b100
‘define F 3’b101

module state2(clk,rst_n,x,q);

  input clk, rst_n, x;    // 1ビットの入力ポート
  output [2:0] q; // 3ビットの出力ポートq
  reg [2:0] q;    // 3ビットの変数q

  always @(posedge clk, negedge rst_n)
    if(!rst_n) q<=‘A;    // 非同期リセット
    else case(q)    // 次の状態を決める
          ‘A: if(x) q<=‘A; else q<=‘B;
          ‘B: if(x) q<=‘A; else q<=‘C;
          ‘C: if(x) q<=‘D; else q<=‘C;
          ‘D: if(x) q<=‘E; else q<=‘B;
          ‘E: if(x) q<=‘F; else q<=‘B;
          ‘F: if(x) q<=‘A; else q<=‘B;
          default: q<=3’bXXX;
        endcase

endmodule
```

【2.5】

```
‘define A 3’b000
‘define B 3’b001
‘define C 3’b010
‘define D 3’b011
‘define E 3’b100
‘define F 3’b101

module state3(clk,rst_n,x,q);

  input clk, rst_n, x;    // 1ビットの入力ポート
  output [2:0] q; // 3ビットの出力ポートq
  reg [2:0] q;    // 3ビットの変数q

  always @(posedge clk, negedge rst_n)
```

```
    if(!rst_n) q<=`A;    // 非同期リセット
    else case(q)    // 次の状態を決める
          `A: if(x) q<=`B; else q<=`A;
          `B: if(x) q<=`B; else q<=`C;
          `C: if(x) q<=`D; else q<=`A;
          `D: if(x) q<=`B; else q<=`E;
          `E: if(x) q<=`F; else q<=`A;
          `F: if(x) q<=`B; else q<=`E;
          default: q<=3'bXXX;
        endcase

endmodule
```

【2.6】

```
module stack(clk,load,push,pop,d,qtop,qnext);
  parameter N=16;

  input clk,load,push,pop;    // 1ビットの入力ポート
  input [N-1:0] d;    // Nビットの入力ポートd
  output [N-1:0] qtop, qnext; // Nビットの出力ポートqtop, qnext
  reg [N-1:0] qtop, qnext; // Nビットの変数qtop, qnext
  reg [N-1:0] q [0:3];    // Nビットの変数q[0],q[1],q[2],q[3]

  always @(q[0], q[1]) begin
    qtop <= q[0];    // qtopにq[0]を出力
    qnext <= q[1];   // qnextにq[1]を出力
  end

  always @(posedge clk) begin  // clkの立ち上がりで動作
    if(load) q[0]<=d; else if(pop) q[0]<=q[1];
    if(push) q[1]<=q[0]; else if(pop) q[1]<=q[2];
    if(push) q[2]<=q[1]; else if(pop) q[2]<=q[3];
    if(push) q[3]<=q[2]; else if(pop) q[3]<={N{1'bX}};
  end

endmodule
```

【2.7】

```
module stack8(clk,load,push,pop,d,qtop,qnext);
  parameter N=16;

  input clk,load,push,pop;    // 1ビットの入力ポート
  input [N-1:0] d;    // Nビットの入力ポートd
  output [N-1:0] qtop, qnext; // Nビットの出力ポートqtop, qnext
  reg [N-1:0] q [0:7];    // Nビットの変数q[0],q[1],q[2],...,q[7]

  assign qtop = q[0]; // qtopにq[0]を出力
  assign qnext = q[1];   // qnextにq[1]を出力

  always @(posedge clk) begin  // clkの立ち上がりで動作
    if(load) q [0]<=d; else if(pop) q[0]<=q[1];
    if(push) q [1]<=q[0]; else if(pop) q[1]<=q[2];
    if(push) q [2]<=q[1]; else if(pop) q[2]<=q[3];
    if(push) q [3]<=q[2]; else if(pop) q[3]<=q[4];
    if(push) q [4]<=q[3]; else if(pop) q[4]<=q[5];
    if(push) q [5]<=q[4]; else if(pop) q[5]<=q[6];
    if(push) q [6]<=q[5]; else if(pop) q[6]<=q[7];
    if(push) q [7]<=q[6]; else if(pop) q[7]<={N{1'bX}};
  end

endmodule
```

【2.8】

```
module stackM(clk,load,push,pop,d,qtop,qnext);
  parameter N=16,M=8;

  input clk,load,push,pop;    // 1ビットの入力ポート
  input [N-1:0] d;    // Nビットの入力ポートd
  output [N-1:0] qtop, qnext; // Nビットの出力ポートqtop, qnext
  reg [N-1:0] q [0:M-1];    // Nビットの変数q[0],q[1],q[2],...,q[M-1]

  assign qtop = q[0]; // qtopにq[0]を出力
  assign qnext = q[1];   // qnextにq[1]を出力

  integer i;
  always @(posedge clk) begin  //
     clkの立ち上がりで動作
    if(load) q [0]<=d; else if(pop) q[0]<=q[1];
    for(i=1;i<M-1;i=i+1)
      if(push) q[i]<=q[i-1]; else if(pop) q[i]<=q[i
          +1];
    if(push) q[M-1]<=q[M-2]; else if(pop) q[M-1]<={
        N{1'bX}};
  end

endmodule
```

3章

【3.1】

(1)

$$
\begin{aligned}
\texttt{[a+(b-(c+d*e))]} &\rightarrow \texttt{[a][b-(c+d*e)]+} \\
&\rightarrow \texttt{[a][b][c+d*e]-+} \\
&\rightarrow \texttt{[a][b][c][d*e]+-+} \\
&\rightarrow \texttt{[a][b][c][d][e]*+-+} \\
&\rightarrow \texttt{abcde*+-+}
\end{aligned}
$$

(2)

$$
\begin{aligned}
\texttt{[a+b-c+d-e*f]} &\rightarrow \texttt{[a+b-c+d][e*f]-} \\
&\rightarrow \texttt{[a+b-c+d]+[e][f]*-} \\
&\rightarrow \texttt{[a+b-c][d]+[e][f]*-} \\
&\rightarrow \texttt{[a+b][c]-[d]+[e][f]*-} \\
&\rightarrow \texttt{[a][b]+[c]-[d]+[e][f]*-} \\
&\rightarrow \texttt{ab+c-d+ef*-}
\end{aligned}
$$

(3)

$$
\begin{aligned}
\texttt{[(a+b*c<d)\&\&!(a>d-e)]} &\rightarrow \texttt{[(a+b*c<d)][!(a>d-e)]\&\&} \\
&\rightarrow \texttt{[(a+b*c<d)][(a>d-e)]!\&\&} \\
&\rightarrow \texttt{[(a+b*c<d)][a][d-e]>!\&\&} \\
&\rightarrow \texttt{[(a+b*c<d)][a][d][e]->!\&\&} \\
&\rightarrow \texttt{[a+b*c][d]<[a][d][e]->!\&\&} \\
&\rightarrow \texttt{[a][b*c]+[d]<[a][d][e]->!\&\&} \\
&\rightarrow \texttt{[a][b][c]*+[d]<[a][d][e]->!\&\&} \\
&\rightarrow \texttt{abc*+d<ade->!\&\&}
\end{aligned}
$$

【3.2】

(1)

$$
\begin{aligned}
\texttt{[1+(2+3*4-5)-6]} &\rightarrow \texttt{[1+(2+3*4-5)][6]-} \\
&\rightarrow \texttt{[1][2+3*4-5]+[6]-} \\
&\rightarrow \texttt{[1][2+3*4][5]-+[6]-} \\
&\rightarrow \texttt{[1][2][3*4]+[5]-+[6]-} \\
&\rightarrow \texttt{[1][2][3][4]*+[5]-+[6]-} \\
&\rightarrow \texttt{1234*+5-+6-}
\end{aligned}
$$

```
`include "alu_d.v"
module opstack_tb1;

  reg clk,num,op; // 1ビットの変数clk,num,op
  reg [15:0] x;   // 16ビットの変数x
  wire [15:0] qtop;   // 16ビットの配線qtop

  opstack opstack0(clk,num,op,x,qtop);

  initial clk=0;
  always #50 clk=~clk;

  initial begin   // 後置記法のオペランド・演算を
     xに代入
    $dumpvars;
    num = 1; op = 0; x = 1; #100
    num = 1; op = 0; x = 2; #100
    num = 1; op = 0; x = 3; #100
    num = 1; op = 0; x = 4; #100
    num = 0; op = 1; x = `MUL; #100
    num = 0; op = 1; x = `ADD; #100
    num = 1; op = 0; x = 5; #100
    num = 0; op = 1; x = `SUB; #100
    num = 0; op = 1; x = `ADD; #100
    num = 1; op = 0; x = 6; #100
    num = 0; op = 1; x = `SUB; #100
    $finish;
  end

endmodule
```

(3)

```
[(1+2>4*5)||!(6<7)&&(8>9)] → [1+2>4*5][!(6<7)&&(8>9)]||
                           → [1+2>4*5][!(6<7)][(8>9)]&&||
                           → [1+2>4*5][!(6<7)][8][9]>&&||
                           → [1+2>4*5][(6<7)]![8][9]>&&||
                           → [1+2>4*5][6][7]<![8][9]>&&||
                           → [1+2][4*5]>[6][7]<![8][9]>&&||
                           → [1+2][4][5]*>[6][7]<![8][9]>&&||
                           → [1][2]+[4][5]*>[6][7]<![8][9]>&&||
                           → 12+45*>67<!89>&&||
```

```
`include "alu_d.v"
module opstack_tb1;

  reg clk,num,op; // 1ビットの変数clk,num,op
  reg [15:0] x;   // 16ビットの変数x
  wire [15:0] qtop;   // 16ビットの配線qtop

  opstack opstack0(clk,num,op,x,qtop);

  initial clk=0;
  always #50 clk=~clk;

  initial begin   // 後置記法のオペランド・演算を
     xに代入
    $dumpvars;
    num = 1; op = 0; x = 1; #100
    num = 1; op = 0; x = 2; #100
    num = 1; op = 0; x = 3; #100
    num = 1; op = 0; x = 4; #100
    num = 0; op = 1; x = `MUL; #100
    num = 0; op = 1; x = `ADD; #100
    num = 1; op = 0; x = 5; #100
    num = 0; op = 1; x = `SUB; #100
    num = 0; op = 1; x = `ADD; #100
    num = 1; op = 0; x = 6; #100
    num = 0; op = 1; x = `SUB; #100
    $finish;
  end

endmodule
```

【3.4】

```
`include "state_d.v"
module fetch(clk,rst_n,run,irout);

    input clk,rst_n,run;   // 1ビットの入力ポート
    output [15:0] irout;    // 16ビットの出力ポート
       irout
    wire [1:0] cs; // 2ビットの配線cs
    wire [11:0] pcout;  // 12ビットの配線pcout
    wire [15:0] ramout; // 16ビットの配線ramout

    state state0(.clk(clk),.rst_n(rst_n),.run(run
       ),.halt(cs==`EXEC && ramout==16'h0000),.q(cs
       ));
    counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load
       (1'b0),.inc(cs==`FETCH),.q(pcout));
    counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load
       (cs==`FETCH),.inc(1'b0),.d(ramout),.q(irout
       ));
    ram ram0(.clk(clk),.load(1'b0),.addr(pcout),.q(
       ramout));

endmodule
```

【3.6】

```
`include "state_d.v"
module formula(clk,rst_n,run,qtop);

    input clk,rst_n,run;    // 1ビットの入力ポート
    output [15:0] qtop;     // 16ビットの出力ポート qtop
    wire [1:0] cs;  // 2ビットの配線cs
    wire [11:0] pcout;  // 12ビットの配線pcout
    wire [15:0] ramout, irout;  // 16ビットの配線 ramout,irout
    wire num,op;    // 1ビットの配線num,op

    state state0(.clk(clk),.rst_n(rst_n),.run(run),.halt(cs==`EXEC && ramout==16'hFFFF),.q(cs));
    counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(1'b0),.inc(cs==`FETCH),.q(pcout));
    counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(cs==`FETCH),.inc(1'b0),.d(ramout),.q(irout));
    ram ram0(.clk(clk),.load(1'b0),.addr(pcout),.q(ramout));
    opstack opstack0(.clk(clk),.num(num),.op(op),.x(irout),.qtop(qtop));

    assign num=(cs==`EXEC)&&~irout[15];
    assign op=(cs==`EXEC)&&irout[15];

endmodule
```

【3.7】

```
reg [15:0] bus; // 16ビットの変数バス

always @(a,b,c,asel,bsel,csel) // busの値を決める
  case({asel,bsel,csel})
    3'b100: bus=a; // aselが1のときはa
    3'b010: bus=b; // bselが1のときはb
    3'b001: bus=c; // cselが1のときはc
    default: bus=16'hXXXX;
  endcase
```

【3.8】

```
`include "state_d.v"
module exfetch(clk,rst_n,run,halt);

```

```
  input clk,rst_n,run,halt;   // 1ビットの入力ポート
  wire [1:0] cs; // 2ビットの配線cs
  wire [11:0] pcout;  // 12ビットの配線pcout
  wire [15:0] irout,ramout; // 16ビットの配線irout,ramout
  reg [15:0] dbus;
  wire [11:0] abus;

  state state0(.clk(clk),.rst_n(rst_n),.run(run),.halt(halt),.q(cs));
  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(1'b0),.inc(cs==`FETCH),.q(pcout));
  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(cs==`FETCH),.inc(1'b0),.d(ramout),.q(irout));
  ram ram0(.clk(clk),.load(cs==`EXEC),.addr(abus),.d(dbus),.q(ramout));


  assign abus=((cs==`FETCH)?pcout:12'hZZZ);
  assign abus=((cs==`EXEC)?{4'h0,irout[15:8]}:12'hZZZ);

  always @(cs,ramout,irout)
    if(cs==`FETCH) dbus=ramout;
    else if(cs==`EXEC) dbus={8'h00,irout[7:0]};
    else dbus=12'hXXX;

endmodule
```

4章

【4.1】

オペランド X は符号付き 12 ビット整数より, 最大値は $2^{11} - 1 = 2047$, 最小値は $-2^{11} = -2048$ となる。

【4.2】

```
`include "state_d.v"     // ステートマシン回路の定数定義
`include "alu_d.v"       // 算術論理演算回路の定数定義
`include "inst.v"        // 機械語コードの定義
```

```
module tinycpu(clk,rst_n,run,out);

  input clk,rst_n,run;     // 1ビットの入力ポート
  output [15:0] out;  // 16ビットの出力ポートout
  wire [1:0] cs;  // 2ビットの配線cs
  wire [11:0] pcout;  // 12ビットの配線pcout
  wire [15:0] irout,ramout,qtop,qnext,aluout;
  reg [15:0] dbus;    // 12ビットの変数dbus
  wire [11:0] abus;     // 12ビットの配線abus
  reg halt,pcinc,push,pop,abus2pc,dbus2ir,
     dbus2stack,dbus2ram,dbus2out,pc2abus,ir2abus,
     alu2dbus,ir2dbus,stack2dbus,ram2dbus; //1ビッ
     トの変数

  state state0(.clk(clk),.rst_n(rst_n),.run(run),.
     halt(halt),.q(cs));
  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(
     abus2pc),.inc(pcinc),.d(abus),.q(pcout));
  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(
     dbus2ir),.inc(1'b0),.d(dbus),.q(irout));
  alu alu0(.a(qtop),.b(qnext),.f(irout[4:0]),.s(
     aluout));
  stack stack0(.clk(clk),.load(dbus2stack),.push(
     push),.pop(pop),.d(dbus),.qtop(qtop),.qnext(
     qnext));
  ram ram0(.clk(clk),.load(dbus2ram),.addr(abus),.d
     (dbus),.q(ramout));
  counter #(16) out0(.clk(clk),.rst_n(rst_n),.load(
     dbus2out),.inc(1'b0),.d(dbus),.q(out));

  assign abus = pc2abus?pcout:12'hZZZ;
  assign abus = ir2abus?irout[11:0]:12'hZZZ;

  always @(ram2dbus,ramout,ir2dbus,irout,stack2dbus
     ,qtop,alu2dbus,aluout)
    if(ram2dbus) dbus = ramout;
    else if(ir2dbus) dbus = {{4{irout[11]}},irout
       [11:0]};
    else if(stack2dbus) dbus = qtop;
    else if(alu2dbus) dbus = aluout;
    else dbus = 16'hXXXX;

  always @(cs,irout,qtop) // 制御線の値の決定
  begin
    halt=0;pcinc=0;push=0;pop=0;abus2pc=0;dbus2ir
       =0;dbus2stack=0;dbus2ram=0;dbus2out=0;
```

```
        pc2abus=0;ir2abus=0;alu2dbus=0;ir2dbus=0;
        stack2dbus=0;ram2dbus=0;
    if(cs==`FETCH)
    begin
      pc2abus=1; ram2dbus=1; dbus2ir=1; pcinc=1;
    end
    else if(cs==`EXEC)
      case(irout[15:12])
        `HALT: halt=1;
        `PUSHI: begin ir2dbus=1; dbus2stack=1; push
            =1; end
        `PUSH: begin ir2abus=1; ram2dbus=1;
            dbus2stack=1; push=1; end
        `POP: begin ir2abus=1; stack2dbus=1;
            dbus2ram=1; pop=1; end
        `JMP: begin ir2abus=1; abus2pc=1; end
        `JZ: begin pop=1;
                if(qtop==0) begin ir2abus=1; abus2pc
                    =1; end
              end
        `JNZ: begin pop=1;
                if(qtop!=0) begin ir2abus=1; abus2pc
                    =1; end
               end
        `OUT: begin stack2dbus=1; dbus2out=1; pop
            =1; end
        `OP: begin alu2dbus=1; dbus2stack=1;
                 if(irout[4]==0) pop=1;
              end
      endcase
    end

endmodule
```

【4.3】

```
`include "state_d.v"    // ステートマシン回路の定数
    定義
`include "alu_d.v"      // 算術論理演算回路の定数定
    義
`include "inst.v"       // 機械語コードの定義
module tinycpu(clk,rst_n,run,in,out);

  input clk,rst_n,run;    // 1ビットの入力ポート
  input [15:0] in;    // 16ビットの入力ポート
  output [15:0] out;  // 16ビットの出力ポートout
  wire [1:0] cs;  // 2ビットの配線cs
  wire [11:0] pcout;  // 12ビットの配線pcout
  wire [15:0] dbus,irout,ramout,qtop,qnext,aluout;
  reg [11:0] abus;    // 12ビットの変数abus
  reg halt,pcinc,push,pop,abus2pc,dbus2ir,
     dbus2stack,dbus2ram,dbus2out,pc2abus,ir2abus,
     alu2dbus,ir2dbus,stack2dbus,ram2dbus; //1ビッ
     トの変数
  reg in2dbus;

  state state0(.clk(clk),.rst_n(rst_n),.run(run),.
     halt(halt),.q(cs));
  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(
     abus2pc),.inc(pcinc),.d(abus),.q(pcout));
  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(
     dbus2ir),.inc(1'b0),.d(dbus),.q(irout));
  alu alu0(.a(qtop),.b(qnext),.f(irout[4:0]),.s(
     aluout));
  stack stack0(.clk(clk),.load(dbus2stack),.push(
     push),.pop(pop),.d(dbus),.qtop(qtop),.qnext(
     qnext));
  ram ram0(.clk(clk),.load(dbus2ram),.addr(abus),.d
     (dbus),.q(ramout));
  counter #(16) out0(.clk(clk),.rst_n(rst_n),.load(
     dbus2out),.inc(1'b0),.d(dbus),.q(out));

  always @ (pc2abus,ir2abus,pcout,irout)
    if(pc2abus) abus = pcout;
    else if(ir2abus) abus = irout[11:0];
    else abus = 12'hXXX;

  assign dbus = ram2dbus?ramout:16'hZZZZ;
  assign dbus = ir2dbus?{{4{irout[11]}},irout
     [11:0]}:16'hZZZZ;
  assign dbus = stack2dbus?qtop:16'hZZZZ;
  assign dbus = alu2dbus?aluout:16'hZZZZ;
  assign dbus = in2dbus?in:16'hZZZZ;

  always @(cs,irout,qtop) // 制御線の値の決定
  begin
    halt=0;pcinc=0;push=0;pop=0;abus2pc=0;dbus2ir
       =0;dbus2stack=0;dbus2ram=0;dbus2out=0;
       pc2abus=0;ir2abus=0;alu2dbus=0;ir2dbus=0;
       stack2dbus=0;ram2dbus=0;
    in2dbus=0;
    if(cs==`FETCH)
    begin
      pc2abus=1; ram2dbus=1; dbus2ir=1; pcinc=1;
    end
    else if(cs==`EXEC)
      case(irout[15:12])
        `HALT: halt=1;
        `PUSHI: begin ir2dbus=1; dbus2stack=1; push=1; end
        `PUSH: begin ir2abus=1; ram2dbus=1; dbus2stack=1; push=1; end
        `POP: begin ir2abus=1; stack2dbus=1; dbus2ram=1; pop=1; end
        `JMP: begin ir2abus=1; abus2pc=1; end
        `JZ: begin pop=1;
               if(qtop==0) begin ir2abus=1; abus2pc=1; end
             end
        `JNZ: begin pop=1;
                if(qtop!=0) begin ir2abus=1; abus2pc=1; end
              end
        `OUT: begin stack2dbus=1; dbus2out=1; pop=1; end
        `OP: begin alu2dbus=1; dbus2stack=1;
               if(irout[4]==0) pop=1;
             end
        `IN: begin in2dbus=1; dbus2stack=1; push=1; end
      endcase
  end

endmodule
```

【4.4】

```
`include "state_d.v"    // ステートマシン回路の定数定義
`include "alu_d.v"      // 算術論理演算回路の定数定義
`include "inst.v"       // 機械語コードの定義
module tinycpu(clk,rst_n,run,out);

  input clk,rst_n,run;    // 1ビットの入力ポート
  output [15:0] out;  // 16ビットの出力ポートout
```

```verilog
wire [1:0] cs;  // 2ビットの配線cs
wire [11:0] pcout;  // 12ビットの配線pcout
wire [15:0] dbus,irout,ramout,qtop,qnext,aluout;
reg [11:0] abus;    // 12ビットの変数abus
wire halt,pcinc,push,pop,abus2pc,dbus2ir,dbus2stack,dbus2ram,dbus2out,pc2abus,ir2abus,alu2dbus,ir2dbus,stack2dbus,ram2dbus; //1ビットの配線

state state0(.clk(clk),.rst_n(rst_n),.run(run),.halt(halt),.q(cs));
counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(abus2pc),.inc(pcinc),.d(abus),.q(pcout));
counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(dbus2ir),.inc(1'b0),.d(dbus),.q(irout));
alu alu0(.a(qtop),.b(qnext),.f(irout[4:0]),.s(aluout));
stack stack0(.clk(clk),.load(dbus2stack),.push(push),.pop(pop),.d(dbus),.qtop(qtop),.qnext(qnext));
ram ram0(.clk(clk),.load(dbus2ram),.addr(abus),.d(dbus),.q(ramout));
counter #(16) out0(.clk(clk),.rst_n(rst_n),.load(dbus2out),.inc(1'b0),.d(dbus),.q(out));

always @ (pc2abus,ir2abus,pcout,irout)
  if(pc2abus) abus = pcout;
  else if(ir2abus) abus = irout[11:0];
  else abus = 12'hXXX;

assign dbus = ram2dbus?ramout:16'hZZZZ;
assign dbus = ir2dbus?{{4{irout[11]}},irout[11:0]}:16'hZZZZ;
assign dbus = stack2dbus?qtop:16'hZZZZ;
assign dbus = alu2dbus?aluout:16'hZZZZ;

assign halt = (cs==`EXEC && irout[15:12]==`HALT) ? 1'b1 : 1'b0;
assign pcinc = (cs==`FETCH) ? 1'b1 : 1'b0;
assign push = (cs==`EXEC && (irout[15:12]==`PUSHI || irout[15:12]==`PUSH)) ? 1'b1 : 1'b0;
assign pop = (cs==`EXEC && (irout[15:12]==`POP || irout[15:12]==`JZ || irout[15:12]==`JNZ|| irout[15:12]==`OUT || (irout[15:12]==`OP && irout[4]==0))) ? 1'b1 : 1'b0;
```

```
  assign abus2pc = (cs==‘EXEC && (irout[15:12]==‘JMP || (irout[15:12]==‘JZ && qtop==0) || (irout[15:12]==‘JNZ && qtop!=0))) ? 1’b1 : 1’b0;
  assign dbus2ir = (cs==‘FETCH) ? 1’b1 : 1’b0;
  assign dbus2stack = (cs==‘EXEC && (irout[15:12]==‘PUSHI || irout[15:12]==‘PUSH || irout[15:12]==‘OP)) ? 1’b1 : 1’b0;
  assign dbus2ram = (cs==‘EXEC && irout[15:12]==‘POP) ? 1’b1 : 1’b0;
  assign dbus2out = (cs==‘EXEC && irout[15:12]==‘OUT) ? 1’b1 : 1’b0;
  assign pc2abus = (cs==‘FETCH) ? 1’b1 : 1’b0;
  assign ir2abus = (cs==‘EXEC && (irout[15:12]==‘PUSH || irout[15:12]==‘POP || irout[15:12]==‘JMP || (irout[15:12]==‘JZ && qtop==0) || (irout[15:12]==‘JNZ && qtop!=0))) ? 1’b1 : 1’b0;
  assign alu2dbus = (cs==‘EXEC && irout[15:12]==‘OP) ? 1’b1 : 1’b0;
  assign ir2dbus = (cs==‘EXEC && irout[15:12]==‘PUSHI) ? 1’b1 : 1’b0;
  assign stack2dbus = (cs==‘EXEC && (irout[15:12]==‘POP || irout[15:12]==‘OUT)) ? 1’b1 : 1’b0;
  assign ram2dbus = ((cs==‘FETCH) || (cs==‘EXEC && irout[15:12]==‘PUSH)) ? 1’b1 : 1’b0;

endmodule
```

【4.6】

```
L1: PUSH n
    PUSHI 10
    NE
    JNZ L2
    HALT
L2: PUSH n
    OUT
    PUSH n
    PUSHI 1
    ADD
    POP n
    JMP L1
n:  1
```

【4.7】

```
     PUSH a1
     OUT
 L1: PUSH a2
     OUT
     PUSH a1
     PUSH a2
     ADD
     PUSH a2
     POP a1
     POP a2
     JMP L1
 a1: 1
 a2: 1
```

5章

【5.1】

```
     PUSH x
     PUSH y
     GT          // x>y
     JZ L1F
     PUSH x
     PUSH z
     GT          // x>z
     JZ L2F
     PUSH x
     POP t       // t=x
     JMP L2T
L1F: PUSH y
     PUSH z
     GT          // y>z
     JZ L2F
     PUSH y
     POP t       // t=y
     JMP L2T
L2F: PUSH z
     POP t       // t=z
L2T:
```

【5.2】

(1)

```
do{
  out(n);
  n=n-1;
}while(n>0);
halt;
int n=10;
```

(2)

```
do{
  if(n&1){
    out(n*3);
  }else{
    out(n>>1);
  }
  n=n-1;
}while(n>0);
halt;
int n=10;
```

(3)

```
m=0;
while(n<=3*m){
  m=m+1;
}
out(m);
n = n-3*m;
out(n);
halt;
int n=10;
int m;
```

(4)

```
do{
  if(a<b){b=a^b; a=a^b; b=a^b;}
  out(a);
  a=a-b;
}while(b);
halt;
int a=30,b=21;
```

【5.4】

for(A;B;C){D;} の場合、以下のようにアセンブリプログラムに変換できる

```
      Aの実行
L1T:  Bの計算
      JZ L1F
      Dの実行
      Cの実行
      JMP L1T
L1F:
```

```
// for(n=1;n<10;n=n+1){out(n);}の変換
      PUSHI 1
      POP n    // n=1
L1T:  PUSH n
      PUSHI 10
      LT       // n<10
      JZ L1F
      PUSH n
      OUT      // out(n)
      PUSH n
      PUSHI 1
      ADD
      POP n  // n=n+1
L1F:
```

【5.5】

式 1 ? 式 2 : 式 3 の場合、以下のようにアセンブリプログラムに変換できる

```
      式1の計算
      JZ L1F
      式2の計算
      JMP L1T
L1F:  式3の計算
L1T:
```

```
// x=x<0?-x:xの変換
      PUSH x
      PUSHI 0
```

```
        LT             // x<0
        JZ L1F
        PUSH x
        NEG            // -x
        JMP L1T
L1F: PUSH x      // x
L1T: POP x
```

【5.6】

式 1 && 式 2 の場合、以下のようにアセンブリプログラムに変換できる

```
        式1の 計 算
        JZ L1F
        式2の 計 算
        JMP L1T
L1F: PUSHI 0
L1T:
```

```
// if((5<x)&&(x<10)){out(n);}
        PUSHI 5
        PUSH x
        LT           // 5<x
        JZ L1F
        PUSH x
        PUSHI 10
        LT           // x<10
        JMP L1T
L1F: PUSHI 0
L1T: JZ L2F
        PUSH n
        OUT          // out(n)
L2F:
```

【5.7】

式 1 || 式 2 の場合、以下のようにアセンブリプログラムに変換できる

```
        式1の 計 算
        JNZ L1T
        式2の 計 算
        JMP L1F
L1T: PUSHI 1
L1F:
```

```
// if((x<5)||(10<x)){out(n);}
     PUSH x
     PUSHI 5
     LT          // x<5
     JNZ L1T
     PUSHI 10
     PUSH x
     LT          // 10<x
     JMP L1F
L1T: PUSHI 1
L1F: JZ L2F
     PUSH n
     OUT         // out(n)
L2F:
```

6章

【6.1】

```
#!/usr/bin/perl -W

%MCODE=(HALT=>0x0000,PUSHI=>0x1000,PUSH=>0x2000,POP
    =>0x3000,JMP=>0x4000,JZ=>0x5000,JNZ=>0x6000,IN
    =>0xD000,OUT=>0xE000,ADD=>0xF000,SUB=>0xF001,MUL
    =>0xF002,SHL=>0xF003,SHR=>0xF004,BAND=>0xF005,
    BOR=>0xF006,BXOR=>0xF007,AND=>0xF008,OR=>0xF009,
    EQ=>0xF00A,NE=>0xF00B,GE=>0xF00C,LE=>0xF00D,GT
    =>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT
    =>0xF012);

$adddr=0;
while(<>){
  push @source,$_;
  if(s/^\s*(\w+)://){
    $label{$1}=$addr;
    printf("%s:%03X\n",$1,$label{$1});#ラベルの表示
  }
  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
}

$addr=0;
foreach(@source){
  $m=$_;
  s/^\s*\w+://;
  if(/^\s*PUSHI\s+(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {PUSHI}+($1&0xfff));
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1}+$label{$2});
  }elsif(/^\s*([A-Z]+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1});
  }elsif(/^\s*(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$1&0
        xffff);
  }else{
    printf("%21s",$m);
  }
  printf(" // %s",$m);
}
```

【6.2】

```
#!/usr/bin/perl -W

%MCODE=(HALT=>0x0000,PUSHI=>0x1000,PUSH=>0x2000,POP
    =>0x3000,JMP=>0x4000,JZ=>0x5000,JNZ=>0x6000,IN
    =>0xD000,OUT=>0xE000,ADD=>0xF000,SUB=>0xF001,MUL
    =>0xF002,SHL=>0xF003,SHR=>0xF004,BAND=>0xF005,
    BOR=>0xF006,BXOR=>0xF007,AND=>0xF008,OR=>0xF009,
    EQ=>0xF00A,NE=>0xF00B,GE=>0xF00C,LE=>0xF00D,GT
    =>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT
    =>0xF012);

$adddr=0;
while(<>){
  push @source,$_;
  if(s/^\s*(\w+)://){$label{$1}=$addr;}
  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
}

$addr=0;
$line=0;
foreach(@source){
  $line++;
  $m=$_;
  s/^\s*\w+://;
  if(/^\s*PUSHI\s+(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {PUSHI}+($1&0xfff));
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1}+$label{$2});
  }elsif(/^\s*([A-Z]+)/){
    if(!defined($MCODE{$1})){
      printf("ERROR! line %d: instruction %s is not
          defined.\n",$line,$1);
      exit(1);
    }
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1});
  }elsif(/^\s*(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$1&0
        xffff);
  }else{
    printf("%21s",$m);
  }
  printf(" // %s",$m);
}
```

【6.3】

```
#!/usr/bin/perl -W

%MCODE=(HALT=>0x0000,PUSHI=>0x1000,PUSH=>0x2000,POP
    =>0x3000,JMP=>0x4000,JZ=>0x5000,JNZ=>0x6000,IN
    =>0xD000,OUT=>0xE000,ADD=>0xF000,SUB=>0xF001,MUL
    =>0xF002,SHL=>0xF003,SHR=>0xF004,BAND=>0xF005,
    BOR=>0xF006,BXOR=>0xF007,AND=>0xF008,OR=>0xF009,
    EQ=>0xF00A,NE=>0xF00B,GE=>0xF00C,LE=>0xF00D,GT
    =>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT
    =>0xF012);

$adddr=0;
while(<>){
  push @source,$_;
  if(s/^\s*(\w+)://){
    $label{$1}=$addr;
  }
  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
}

```

```
$addr=0;
$line=0;
foreach(@source){
  $line++;
  $m=$_;
  s/^\s*\w+://;
  if(/^\s*PUSHI\s+(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {PUSHI}+($1&0xfff));
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
    if(!defined($label{$2})){
      printf("ERROR! line %d: label %s is not
          defined.\n",$line,$2);
      exit(1);
    }
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1}+$label{$2});
  }elsif(/^\s*([A-Z]+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE
        {$1});
  }elsif(/^\s*(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$1&0
        xffff);
  }else{
    printf("%21s",$m);
  }
  printf(" // %s",$m);
}
```

【6.4】

```
#!/usr/bin/perl -W

%MCODE=(HALT=>0x0000,PUSHI=>0x1000,PUSH=>0x2000,POP
    =>0x3000,JMP=>0x4000,JZ=>0x5000,JNZ=>0x6000,IN
    =>0xD000,OUT=>0xE000,ADD=>0xF000,SUB=>0xF001,MUL
    =>0xF002,SHL=>0xF003,SHR=>0xF004,BAND=>0xF005,
    BOR=>0xF006,BXOR=>0xF007,AND=>0xF008,OR=>0xF009,
    EQ=>0xF00A,NE=>0xF00B,GE=>0xF00C,LE=>0xF00D,GT
    =>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT
    =>0xF012);

$adddr=0;
while(<>){
  push @source,$_;
  if(s/^\s*(\w+)://){
    $label{$1}=$addr;
  }
  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
}

$addr=0;
$line=0;
foreach(@source){
  $line++;
  $m=$_;
  s/^\s*\w+://;
  if(/^\s*PUSHI\s+(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{PUSHI}+($1&0xfff));
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{$1}+$label{$2});
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)/){
    printf("ERROR! line %d: operand for %s is missing.\n",$line,$1);
    exit(1);
  }elsif(/^\s*([A-Z]+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{$1});
  }elsif(/^\s*(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$1&0xffff);
  }else{
    printf("%21s",$m);
  }
  printf(" // %s",$m);
}
```

【6.5】

```
#!/usr/bin/perl -W

%MCODE=(HALT=>0x0000,PUSHI=>0x1000,PUSH=>0x2000,POP=>0x3000,JMP=>0x4000,JZ=>0x5000,JNZ=>0x6000,IN=>0xD000,OUT=>0xE000,ADD=>0xF000,SUB=>0xF001,MUL=>0xF002,SHL=>0xF003,SHR=>0xF004,BAND=>0xF005,BOR=>0xF006,BXOR=>0xF007,AND=>0xF008,OR=>0xF009,EQ=>0xF00A,NE=>0xF00B,GE=>0xF00C,LE=>0xF00D,GT
=>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT=>0xF012);

$adddr=0;
while(<>){
  push @source,$_;
  if(s/^\s*(\w+)://){
    $label{$1}=$addr;
  }
  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
}

$addr=0;
$line=0;
foreach(@source){
  $line++;
  $m=$_;
  s/^\s*\w+://;
  if(/^\s*PUSHI\s+(-?\d+)/){
    if(!(-2048<=$1 && $1<=2047)){
      printf("ERROR! line %d: operand %d of PUSHI is out of range.\n",$line,$1);
      exit(1);
    }
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{PUSHI}+($1&0xfff));
  }elsif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{$1}+$label{$2});
  }elsif(/^\s*([A-Z]+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$MCODE{$1});
  }elsif(/^\s*(-?\d+)/){
    printf("mem[12'h%03X]=16'h%04X;",$addr++,$1&0xffff);
  }else{
    printf("%21s",$m);
  }
  printf(" // %s",$m);
}
```

7章

【7.1】

```
%{
int s[4];
void printstack();
void push(int);
int pop();
void add();
void sub();
void mul();
void shl();
void shr();
void band();
void bor();
void bxor();
void and();
void or();
void eq();
void ne();
void ge();
void le();
void gt();
void lt();
void neg();
void bnot();
void not();
%}
%%
\n printf("result=%d\n",pop());
[0-9]+ {push(atoi(yytext)); printstack();}
\+ {add(); printstack();}
\- {sub(); printstack();}
\* {mul(); printstack();}
\>\> {shl(); printstack();}
\<\< {shr(); printstack();}
& {band(); printstack();}
\| {bor(); printstack();}
\^ {bxor(); printstack();}
\&\& {and(); printstack();}
\|\| {or(); printstack();}
== {eq(); printstack();}
!= {ne(); printstack();}
\>= {ge(); printstack();}
\<= {le(); printstack();}
\> {gt(); printstack();}
\< {lt(); printstack();}
\@ {neg(); printstack();}
```

```
\~ {bnot(); printstack();}
! {not(); printstack();}
.
%%
void printstack(){printf("%d %d %d %d\n",s[0],s[1],
    s[2],s[3]);}
void push(int x){s[3]=s[2];s[2]=s[1];s[1]=s[0];s
    [0]=x;}
int pop(){int x=s[0];s[0]=s[1];s[1]=s[2];s[2]=s[3];
    return(x);}
void add(){s[0]=s[1]+s[0];s[1]=s[2];s[2]=s[3];}
void sub(){s[0]=s[1]-s[0];s[1]=s[2];s[2]=s[3];}
void mul(){s[0]=s[1]*s[0];s[1]=s[2];s[2]=s[3];}
void shl(){s[0]=s[1]<<s[0];s[1]=s[2];s[2]=s[3];}
void shr(){s[0]=s[1]>>s[0];s[1]=s[2];s[2]=s[3];}
void band(){s[0]=s[1]&s[0];s[1]=s[2];s[2]=s[3];}
void bor(){s[0]=s[1]|s[0];s[1]=s[2];s[2]=s[3];}
void bxor(){s[0]=s[1]^s[0];s[1]=s[2];s[2]=s[3];}
void and(){s[0]=s[1]&&s[0];s[1]=s[2];s[2]=s[3];}
void or(){s[0]=s[1]||s[0];s[1]=s[2];s[2]=s[3];}
void eq(){s[0]=s[1]==s[0];s[1]=s[2];s[2]=s[3];}
void ne(){s[0]=s[1]!=s[0];s[1]=s[2];s[2]=s[3];}
void ge(){s[0]=s[1]>=s[0];s[1]=s[2];s[2]=s[3];}
void le(){s[0]=s[1]<=s[0];s[1]=s[2];s[2]=s[3];}
void gt(){s[0]=s[1]>s[0];s[1]=s[2];s[2]=s[3];}
void lt(){s[0]=s[1]<s[0];s[1]=s[2];s[2]=s[3];}
void neg(){s[0]=-s[0];}
void bnot(){s[0]=~s[0];}
void not(){s[0]=!s[0];}
```

【7.3】

(1)

操作	トークンと非終端記号の列	出力
入力	↓NAME[a]=NAME[b]+NAME[c]*NUM[3]-NAME[d]+NUM[5];	
還元 21	stmts↓NAME[a]=NAME[b]+NAME[c]*NUM[3]-NAME[d]+NUM[5];	
シフト	stmts NAME[a]=NAME[b]↓+NAME[c]*NUM[3]-NAME[d]+NUM[5];	
還元 26	stmts NAME[a]=expr↓+NAME[c]*NUM[3]-NAME[d]+NUM[5];	PUSH b
シフト	stmts NAME[a]=expr+NAME[c]↓*NUM[3]-NAME[d]+NUM[5];	
還元 26	stmts NAME[a]=expr+expr↓*NUM[3]-NAME[d]+NUM[5];	PUSH c
シフト	stmts NAME[a]=expr+expr*NUM[3]↓-NAME[d]+NUM[5];	
還元 27	stmts NAME[a]=expr+expr*expr↓-NAME[d]+NUM[5];	PUSHI 3
還元 33	stmts NAME[a]=expr+expr↓-NAME[d]+NUM[5];	MUL
還元 31	stmts NAME[a]=expr↓-NAME[d]+NUM[5];	ADD
シフト	stmts NAME[a]=expr-NAME[d]↓+NUM[5];	
還元 26	stmts NAME[a]=expr-expr↓+NUM[5];	PUSH d
還元 30	stmts NAME[a]=expr↓+NUM[5];	SUB
シフト	stmts NAME[a]=expr+NUM[5]↓;	
還元 27	stmts NAME[a]=expr+expr↓;	PUSHI 5
還元 31	stmts NAME[a]=expr↓;	ADD
シフト	stmts NAME[a]=expr;↓	
還元 24	stmts assign↓	POP a
還元 22	stmts ↓	

【7.6】

tinyc.l

```
%{
#include <string.h>
#include "y.tab.h"
int n=0;
%}
%%
[ \t\r\n]
&&     {return(AND);}
\|\|   {return(OR);}
==     {return(EQ);}
!=     {return(NE);}
\>=    {return(GE);}
\<=    {return(LE);}
\<\<   {return(SHL);}
\>\>   {return(SHR);}
if     {yylval.n=++n;return(IF);}
while {yylval.n=++n;return(WHILE);}
do     {yylval.n=++n;return(DO);}
```

```
int   {return(INT);}
else  {return(ELSE);}
halt  {return(HALT);}
out   {return(OUT);}
\+\+  {return(INC);}
\-\-  {return(DEC);}
[0-9]+ {yylval.n=atoi(yytext);return(NUM);}
[a-zA-Z][a-zA-Z0-9]* {yylval.s=strdup(yytext);return(NAME);}
.     {return(yytext[0]);}
%%
```

tinyc.y

```
%{
#include <stdio.h>
extern int yylex();
int yyerror(const char *s);
%}
%union {int n; char *s;}
%token <s> NAME
%token <n> NUM IF WHILE DO
%token INT ELSE HALT OUT
%type <n> if
%left OR
%left AND
%left '|'
%left '^'
%left '&'
%left EQ NE
%left GE LE '<' '>'
%left SHL SHR
%left '+' '-'
%left '*'
%right '!' '~' NEG
%left INC DEC
%%
stmts: |stmts stmt
;
stmt: intdef|ifelse|while|do|halt|out|assign|inc|dec
;
intdef: INT intlist ';'
;
intlist: integer
| intlist ',' integer
```

```
;
integer: NAME {printf("%s:\t0\n",$1);free($1);}
| NAME '=' NUM {printf("%s:\t%d\n",$1,$3);free($1);}
| NAME '=' '-' NUM {printf("%s:\t%d\n",$1,-$4);free($1);}
;
ifelse: if {printf("L%dF:\n",$1);}
| if {printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);} ELSE '{' stmts '}' {printf("L%0dT:\n",$1);}
;
if: IF '(' expr ')' {printf("\tJZ L%0dF\n",$1);} '{' stmts '}' {$$=$1;}
;
while: WHILE {printf("L%0dT:\n",$1);} '(' expr ')' {printf("\tJZ L%0dF\n",$1);} '{' stmts '}' {printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);}
;
do: DO {printf("L%0dT:\n",$1);} '{' stmts '}' WHILE '(' expr ')' ';' {printf("\tJNZ L%0dT\n",$1);}
;
halt: HALT ';' {printf("\tHALT\n");}
;
out: OUT '(' expr ')' ';' {printf("\tOUT\n");}
;
assign: NAME '=' expr ';' {printf("\tPOP %s\n",$1);free($1);}
;
inc: NAME INC ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tADD\n\tPOP %s\n",$1,$1);free($1);}
| INC NAME ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tADD\n\tPOP %s\n",$2,$2);free($2);}
;
dec: NAME DEC ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tSUB\n\tPOP %s\n",$1,$1);free($1);}
| DEC NAME ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tSUB\n\tPOP %s\n",$2,$2);free($2);}
;
expr: NAME {printf("\tPUSH %s\n",$1);free($1);}
| NUM {printf("\tPUSHI %d\n",$1);}
| '!' expr {printf("\tNOT\n");}
| '~' expr {printf("\tNBNOT\n");}
| '-' expr %prec NEG {printf("\tNEG\n");}
| expr '+' expr {printf("\tADD\n");}
| expr '-' expr {printf("\tSUB\n");}
| expr '*' expr {printf("\tMUL\n");}
| expr AND expr {printf("\tAND\n");}
| expr OR expr {printf("\tOR\n");}
| expr '&' expr {printf("\tBAND\n");}
| expr '|' expr {printf("\tBOR\n");}
| expr '^' expr {printf("\tBXOR\n");}
| expr SHL expr {printf("\tSHL\n");}
| expr SHR expr {printf("\tSHR\n");}
| expr EQ expr {printf("\tEQ\n");}
| expr NE expr {printf("\tNE\n");}
| expr GE expr {printf("\tGE\n");}
| expr LE expr {printf("\tLE\n");}
| expr '<' expr {printf("\tLT\n");}
| expr '>' expr {printf("\tGT\n");}
| '(' expr ')'
;
%%
int yyerror(const char *s){printf("%s\n",s);}
int main(){return(yyparse());}
```

【7.7】

tinyc.l はリスト 7.7 と同じ。

tinyc.y

```
%{
#include <stdio.h>
extern int yylex();
int yyerror(const char *s);
%}
%union {int n; char *s;}
%token <s> NAME
%token <n> NUM IF WHILE DO AND
%token INT ELSE HALT OUT
%type <n> if
%left OR
%left AND
%left '|'
%left '^'
%left '&'
%left EQ NE
%left GE LE '<' '>'
%left SHL SHR
%left '+' '-'
%left '*'
%right '!' '~' NEG
```

```
%%
stmts: |stmts stmt
;
stmt: intdef|ifelse|while|do|halt|out|assign
;
intdef: INT intlist ';'
;
intlist: integer
| intlist ',' integer
;
integer: NAME {printf("%s:\t0\n",$1);free($1);}
| NAME '=' NUM {printf("%s:\t%d\n",$1,$3);free($1
    );}
| NAME '=' '-' NUM {printf("%s:\t%d\n",$1,-$4);free
    ($1);}
;
ifelse: if {printf("L%dF:\n",$1);}
| if {printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);} ELSE
    '{' stmts '}' {printf("L%0dT:\n",$1);}
;
if: IF '(' expr ')' {printf("\tJZ L%0dF\n",$1);}
    '{' stmts '}' {$$=$1;}
;
while: WHILE {printf("L%0dT:\n",$1);} '(' expr ')'
    {printf("\tJZ L%0dF\n",$1);} '{' stmts '}' {
    printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);}
;
do: DO {printf("L%0dT:\n",$1);} '{' stmts '}' WHILE
     '(' expr ')' ';' {printf("\tJNZ L%0dT\n",$1);}
;
halt: HALT ';' {printf("\tHALT\n");}
;
out: OUT '(' expr ')' ';' {printf("\tOUT\n");}
;
assign: NAME '=' expr ';' {printf("\tPOP %s\n",$1);
     free($1);}
;
expr: NAME {printf("\tPUSH %s\n",$1);free($1);}
| NUM {printf("\tPUSHI %d\n",$1);}
| '!' expr {printf("\tNOT\n");}
| '~' expr {printf("\tNBNOT\n");}
| '-' expr %prec NEG {printf("\tNEG\n");}
| expr '+' expr {printf("\tADD\n");}
| expr '-' expr {printf("\tSUB\n");}
| expr '*' expr {printf("\tMUL\n");}
| expr OR expr {printf("\tOR\n");}
| expr '&' expr {printf("\tBAND\n");}
| expr '|' expr {printf("\tBOR\n");}
| expr '^' expr {printf("\tBXOR\n");}
| expr SHL expr {printf("\tSHL\n");}
| expr SHR expr {printf("\tSHR\n");}
| expr EQ expr {printf("\tEQ\n");}
| expr NE expr {printf("\tNE\n");}
| expr GE expr {printf("\tGE\n");}
| expr LE expr {printf("\tLE\n");}
| expr '<' expr {printf("\tLT\n");}
| expr '>' expr {printf("\tGT\n");}
| '(' expr ')'
| and
;
and: expr AND {printf("\tJZ L%dF\n",$2);} expr {
    printf("\tJMP L%dT\nL%dF:\n\tPUSHI 0\nL%dT:\n",
    $2,$2,$2,$2);}
;
%%
int yyerror(const char *s){printf("%s\n",s);}
int main(){return(yyparse());}
```

【7.8】

tinyc.l はリスト 7.7 と同じ。

tinyc.y

```
%{
#include <stdio.h>
extern int yylex();
int yyerror(const char *s);
%}
%union {int n; char *s;}
%token <s> NAME
%token <n> NUM IF WHILE DO OR
%token INT ELSE HALT OUT
%type <n> if
%left OR
%left AND
%left '|'
%left '^'
%left '&'
%left EQ NE
%left GE LE '<' '>'
%left SHL SHR
```

```
%left ’+’ ’-’
%left ’*’
%right ’!’ ’˜’ NEG
%%
stmts: |stmts stmt
;
stmt: intdef|ifelse|while|do|halt|out|assign
;
intdef: INT intlist ’;’
;
intlist: integer
| intlist ’,’ integer
;
integer: NAME {printf("%s:\t0\n",$1);free($1);}
| NAME ’=’ NUM {printf("%s:\t%d\n",$1,$3);free($1);}
| NAME ’=’ ’-’ NUM {printf("%s:\t%d\n",$1,-$4);free($1);}
;
ifelse: if {printf("L%dF:\n",$1);}
| if {printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);} ELSE ’{’ stmts ’}’ {printf("L%0dT:\n",$1);}
;
if: IF ’(’ expr ’)’ {printf("\tJZ L%0dF\n",$1);} ’{’ stmts ’}’ {$$=$1;}
;
while: WHILE {printf("L%0dT:\n",$1);} ’(’ expr ’)’ {printf("\tJZ L%0dF\n",$1);} ’{’ stmts ’}’ {printf("\tJMP L%0dT\nL%0dF:\n",$1,$1);}
;
do: DO {printf("L%0dT:\n",$1);} ’{’ stmts ’}’ WHILE ’(’ expr ’)’ ’;’ {printf("\tJNZ L%0dT\n",$1);}
;
halt: HALT ’;’ {printf("\tHALT\n");}
;
out: OUT ’(’ expr ’)’ ’;’ {printf("\tOUT\n");}
;
assign: NAME ’=’ expr ’;’ {printf("\tPOP %s\n",$1); free($1);}
;
expr: NAME {printf("\tPUSH %s\n",$1);free($1);}
| NUM {printf("\tPUSHI %d\n",$1);}
| ’!’ expr {printf("\tNOT\n");}
| ’˜’ expr {printf("\tNBNOT\n");}
| ’-’ expr %prec NEG {printf("\tNEG\n");}
| expr ’+’ expr {printf("\tADD\n");}
| expr '-' expr {printf("\tSUB\n");}
| expr '*' expr {printf("\tMUL\n");}
| expr AND expr {printf("\tAND\n");}
| expr '&' expr {printf("\tBAND\n");}
| expr '|' expr {printf("\tBOR\n");}
| expr '^' expr {printf("\tBXOR\n");}
| expr SHL expr {printf("\tSHL\n");}
| expr SHR expr {printf("\tSHR\n");}
| expr EQ expr {printf("\tEQ\n");}
| expr NE expr {printf("\tNE\n");}
| expr GE expr {printf("\tGE\n");}
| expr LE expr {printf("\tLE\n");}
| expr '<' expr {printf("\tLT\n");}
| expr '>' expr {printf("\tGT\n");}
| '(' expr ')'
| or
;
or: expr OR {printf("\tJNZ L%dT\n",$2);} expr {
    printf("\tJMP L%dF\nL%dT:\n\tPUSHI 1\nL%dF:\n",
    $2,$2,$2,$2);}
;
%%
int yyerror(const char *s){printf("%s\n",s);}
int main(){return(yyparse());}
```