

## 1章

### 【1.1】

---

```
1 module f(x,y,z,s,c);
2
3   input x,y,z;    // 入力ポートx,y,z
4   output s,c;    // 出力ポートs,c
5   wire u,v,w;    // 配線u,v,w
6
7
8   assign u=x&y;
9   assign u=x&y;
10  assign v=x&~z;
11  assign w=y|z;
12  assign s=u|v;
13  assign c=~(v&w);
14
15 endmodule
```

---

### 【1.2】

---

```
1 module inc(x,s);
2
3   input [3:0] x; // 4ビットの入力ポートx
4   output [4:0] s; // 5ビットの出力ポートs
5   wire [2:0] c; // 5ビットの配線c
6
7   ha ha0(x[0],1'b1,s[0],c[0]);
8   ha ha1(x[1],c[0],s[1],c[1]);
9   ha ha2(x[2],c[1],s[2],c[2]);
10  ha ha3(x[3],c[2],s[3],s[4]);
11
12 endmodule
```

---

### 【1.3】

---

```
1 module inc(x,s);
2   parameter N=8;
3
4   input [N-1:0] x; // Nビットの入力ポートx
5   output [N:0] s; // N+1ビットの出力ポートs
6   wire [N-2:0] c; // N-1ビットの配線c
```

```

7
8   ha ha0(x[0],1'b1,s[0],c[0]);
9
10  genvar i;
11  for(i=1;i<N-1;i=i+1)
12    ha ha1(x[i],c[i-1],s[i],c[i]);
13
14  ha ha2(x[N-1],c[N-2],s[N-1],s[N]);
15
16  endmodule

```

---

#### 【1.4】

```

1  module adder(x,y,s);
2    parameter N = 8;    // 既定値8のパラメタ
3
4    input [N-1:0] x,y;  // Nビットの入力ポートx,y
5    output [N:0] s;    // N+1ビットの出力ポートs
6    wire [N-2:0] c;    // N-1ビットの配線c
7
8    fa fa0(.x(1'b0),.y(x[0]),.z(y[0]),.s(s[0]),.c(c
9      [0]));
10
11   genvar i;
12   for(i=1;i<=N-2;i=i+1)
13     fa fa1(.x(c[i-1]),.y(x[i]),.z(y[i]),.s(s[i]),.c
14       (c[i]));
15
16   fa fa2(.x(c[N-2]),.y(x[N-1]),.z(y[N-1]),.s(s[N
17     -1]),.c(s[N]));
18
19  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,  $\overline{x_3} \cdot \overline{x_2} \cdot x_1 + \overline{x_3} \cdot x_2 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot x_0$ )

---

```

1 module p1(x,p);
2
3   input [3:0] x; // 4ビットの入力ポートx
4   output p; // 1ビットの出力ポートp
5
6   assign p = (~x[3])&(~x[2])&x[1]
7             | (~x[3])&x[1]&x[0]
8             | x[2]&(~x[1])&x[0]
9             | (~x[2])&x[1]&x[0];
10
11 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。

---

```

1 module p2(x,p);
2
3   input [3:0] x; // 4ビットの入力ポートx
4   output p;     // 1ビットの出力ポートp
5
6   assign p = (x[2] | x[1])
7             & ((~x[2]) | x[0])
8             & ((~x[3]) | x[0])
9             & ((~x[3]) | (~x[2]) | (~x[1]));
10
11 endmodule

```

---

(3)

---

```

1 module p_tb;
2
3   reg [3:0] x; // 4ビットの変数x
4   wire q1, q2; // 1ビットの配線q1, q2
5
6   p1 p1_0(x,q1);
7   p2 p2_0(x,q2);
8
9   initial begin
10    $dumpvars;
11    x = 0; #100
12    x = 1; #100
13    x = 2; #100
14    x = 3; #100
15    x = 4; #100
16    x = 5; #100
17    x = 6; #100
18    x = 7; #100
19    x = 8; #100
20    x = 9; #100
21    x = 10; #100
22    x = 11; #100
23    x = 12; #100
24    x = 13; #100
25    x = 14; #100
26    x = 15; #100
27    $finish;
28   end
29
30 endmodule

```

---

## 【1.6】

---

```
1 module seg7_tb;
2
3   reg [3:0] s; // 4ビットの変数s
4   wire [0:6] x; // 7ビットの配線x
5
6   seg7 seg7_0(s,x);
7
8   initial begin
9     $dumpvars;
10    s = 0; #100
11    s = 1; #100
12    s = 2; #100
13    s = 3; #100
14    s = 4; #100
15    s = 5; #100
16    s = 6; #100
17    s = 7; #100
18    s = 8; #100
19    s = 9; #100
20    $finish;
21  end
22
23 endmodule
```

---

## 【1.8】

---

```
1 module selector8(s,a,x);
2
3   input [2:0] s; // 3ビットの入力ポートs
4   input [0:7] a; // 8ビットの入力ポートa
5   output x; // 1ビットの出力ポートx
6   wire [0:3] c0; // 4ビットの配線c0
7   wire [0:1] c1; // 2ビットの配線c1
8
9   selector2 selector2_00(.s(s[0]),.a(a[0:1]),.x(c0
10    [0]));
11  selector2 selector2_01(.s(s[0]),.a(a[2:3]),.x(c0
12    [1]));
13  selector2 selector2_02(.s(s[0]),.a(a[4:5]),.x(c0
14    [2]));
15  selector2 selector2_03(.s(s[0]),.a(a[6:7]),.x(c0
16    [3]));
17
18 endmodule
```

```

14 selector2 selector2_10(.s(s[1]),.a(c0[0:1]),.x(c1
    [0]));
15 selector2 selector2_11(.s(s[1]),.a(c0[2:3]),.x(c1
    [1]));
16
17 selector2 selector2_2(.s(s[2]),.a(c1[0:1]),.x(x
    ));
18
19 endmodule

```

---

### [1.10]

```

                                alu.d.v
-----
1 'define ADD      5'b00000
2 'define SUB      5'b00001
3 'define MUL      5'b00010
4 'define SHL      5'b00011
5 'define SHR      5'b00100
6 'define BAND     5'b00101
7 'define BOR      5'b00110
8 'define BXOR     5'b00111
9 'define AND      5'b01000
10 'define OR       5'b01001
11 'define EQ       5'b01010
12 'define NE       5'b01011
13 'define GE       5'b01100
14 'define LE       5'b01101
15 'define GT       5'b01110
16 'define LT       5'b01111
17 'define NEG      5'b10000
18 'define BNOT     5'b10001
19 'define NOT      5'b10010
20 'define SQR      5'b11011
21 'define ABS      5'b11100
22 'define INC      5'b11101
23 'define DEC      5'b11110

```

---

```

                                alu.v
-----
1 'include "alu_d.v"
2 module alu(a,b,f,s);
3
4   input signed [15:0] a,b;    // 16ビットの入力ポートa,b

```

```

5   input [4:0] f; // 5ビットの入力ポートf
6   output signed [15:0] s; // 16ビットの出力ポートs
7   reg signed [15:0] s; // 16ビットの変数s
8
9   always @(a,b,f) // a,b,fが変化したときに実行
10      case(f) // fの値で演算が決定
11          'ADD : s = b + a;
12          'SUB : s = b - a;
13          'MUL : s = b * a;
14          'SHL : s = b << a;
15          'SHR : s = b >> a;
16          'BAND: s = b & a;
17          'BOR : s = b | a;
18          'BXOR: s = b ^ a;
19          'AND : s = b && a;
20          'OR  : s = b || a;
21          'EQ  : s = b == a;
22          'NE  : s = b != a;
23          'GE  : s = b >= a;
24          'LE  : s = b <= a;
25          'GT  : s = b > a;
26          'LT  : s = b < a;
27          'NEG : s = -a;
28          'BNOT: s = ~a;
29          'NOT : s = !a;
30          'SQR : s = a * a;
31          'ABS : s = (a>=0) ? a : -a;
32          'INC : s = a + 1;
33          'DEC : s = a - 1;
34          default : s = 16'hXXXX; //
              fが範囲外の場合は不定値
35      endcase
36
37 endmodule

```

---

alu\_tb.v

---

```

1 'include "alu_d.v"
2 module alu_tb;
3
4   reg signed [15:0] a,b; // 16ビットの変数a,b
5   reg [4:0] f; // 4ビットの変数f
6   wire signed [15:0] s; // 16ビットの配線s
7
8   alu alu_0(a,b,f,s); // aluのインスタンス化
9

```

```

10  initial begin
11      $dumpvars;
12      a = 5;  f='SQR;  #100  // fの値を変える
13      a = -5; f='SQR;  #100
14      a = 5;  f='ABS;  #100  // fの値を変える
15      a = -5; f='ABS;  #100
16      a = 5;  f='INC;  #100  // fの値を変える
17      a = -5; f='INC;  #100
18      a = 5;  f='DEC;  #100  // fの値を変える
19      a = -5; f='DEC;  #100
20      $finish;
21  end
22
23  endmodule

```

---

### 【1.11】

(1)

```

1  module decoder1(s,e,d);
2
3      input s;
4      input e;
5      output [0:1] d;
6
7      assign d[0] = (s==1'b0) ? e : 1'b0;
8      assign d[1] = (s==1'b1) ? e : 1'b0;
9
10  endmodule

```

---

(2)

```

1  module decoder2(s,e,d);
2
3      input [1:0] s;
4      input e;
5      output [0:3] d;
6      wire [0:1] c;
7
8      decoder1 decoder1_0(s[1],e,c);
9      decoder1 decoder1_1(s[0],c[0],d[0:1]);
10     decoder1 decoder1_2(s[0],c[1],d[2:3]);
11
12  endmodule

```

---



(3)

---

```
1 module decoder3(s,e,d);
2
3   input [2:0] s;
4   input e;
5   output [0:7] d;
6   wire [0:1] c0;
7   wire [0:1] c1;
8   wire [0:1] c2;
9
10  decoder1 decoder1_0(s[2],e,c0);
11  decoder1 decoder1_1(s[1],c0[0],c1);
12  decoder1 decoder1_2(s[1],c0[1],c2);
13  decoder1 decoder1_3(s[0],c1[0],d[0:1]);
14  decoder1 decoder1_4(s[0],c1[1],d[2:3]);
15  decoder1 decoder1_5(s[0],c2[0],d[4:5]);
16  decoder1 decoder1_6(s[0],c2[1],d[6:7]);
17
18 endmodule
```

---

(4)

---

```
1 module decoder(s,e,d);
2   parameter M=4;
3   parameter N=2**M;
4
5   input [M-1:0] s;
6   input e;
7   output [0:N-1] d;
8
9   genvar i;
10  for(i=0;i<N;i=i+1)
11    assign d[i] = (s==i) ? e : 1'b0;
12
13 endmodule
```

---

## 2章

### 【2.1】

---

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

```

3
4   input clk,rst_n,load,inc,zero,dec;    // 1ビットの
      入力ポート
5   input [N-1:0] d;    // Nビットの入力ポートd
6   output [N-1:0] q;    // Nビットの出力ポートq
7   reg [N-1:0] q;    // Nビットの変数q
8
9   always @(posedge clk, negedge rst_n)
10      if(!rst_n) q<=0;    // 非同期リセット
11      else if(load) q<=d; // dの書き込み
12      else if(inc) q<=q+1; // qが1増える
13      else if(zero) q<=0; // qが0に
14      else if(dec) q<=q-1; // qが1減る
15
16 endmodule

```

---

## 【2.2】

```

1 module counter4(clk,rst_n,load,inc,d,q);
2
3   input clk,rst_n,load,inc;    // 1ビットの入力ポ一
      ト
4   input [3:0] d;    // 4ビットの入力ポートd
5   output [3:0] q;    // 4ビットの出力ポートq
6   wire [2:0] c;    // 3ビットの配線c
7   wire [4:0] s;    // 5ビットの配線s
8   wire [3:0] x;    // 4ビットの配線x
9
10  ha ha0(q[0],1'b1,s[0],c[0]);
11  ha ha1(q[1],c[0],s[1],c[1]);
12  ha ha2(q[2],c[1],s[2],c[2]);
13  ha ha3(q[3],c[2],s[3],s[4]);
14
15  selector2 selector2_0(load,{s[0],d[0]},x[0]);
16  selector2 selector2_1(load,{s[1],d[1]},x[1]);
17  selector2 selector2_2(load,{s[2],d[2]},x[2]);
18  selector2 selector2_3(load,{s[3],d[3]},x[3]);
19
20  ff ff0(clk,1'b1,rst_n,load|inc,x[0],q[0]);
21  ff ff1(clk,1'b1,rst_n,load|inc,x[1],q[1]);
22  ff ff2(clk,1'b1,rst_n,load|inc,x[2],q[2]);
23  ff ff3(clk,1'b1,rst_n,load|inc,x[3],q[3]);
24
25 endmodule

```

---

## 【2.4】

---

```
1 'define A 3'b000
2 'define B 3'b001
3 'define C 3'b010
4 'define D 3'b011
5 'define E 3'b100
6 'define F 3'b101
7
8 module state2(clk,rst_n,x,q);
9
10 input clk, rst_n, x; // 1ビットの入力ポート
11 output [2:0] q; // 3ビットの出力ポートq
12 reg [2:0] q; // 3ビットの変数q
13
14 always @(posedge clk, negedge rst_n)
15     if(!rst_n) q<='A; // 非同期リセット
16     else case(q) // 次の状態を決める
17         'A: if(x) q<='A; else q<='B;
18         'B: if(x) q<='A; else q<='C;
19         'C: if(x) q<='D; else q<='C;
20         'D: if(x) q<='E; else q<='B;
21         'E: if(x) q<='F; else q<='B;
22         'F: if(x) q<='A; else q<='B;
23         default: q<=3'bXXX;
24     endcase
25
26 endmodule
```

---

## 【2.5】

---

```
1 'define A 3'b000
2 'define B 3'b001
3 'define C 3'b010
4 'define D 3'b011
5 'define E 3'b100
6 'define F 3'b101
7
8 module state3(clk,rst_n,x,q);
9
10 input clk, rst_n, x; // 1ビットの入力ポート
11 output [2:0] q; // 3ビットの出力ポートq
12 reg [2:0] q; // 3ビットの変数q
13
14 always @(posedge clk, negedge rst_n)
```

```

15     if(!rst_n) q<='A;    // 非同期リセット
16     else case(q)    // 次の状態を決める
17         'A: if(x) q<='B; else q<='A;
18         'B: if(x) q<='B; else q<='C;
19         'C: if(x) q<='D; else q<='A;
20         'D: if(x) q<='B; else q<='E;
21         'E: if(x) q<='F; else q<='A;
22         'F: if(x) q<='B; else q<='E;
23         default: q<='3'bXXX;
24     endcase
25
26 endmodule

```

---

## [2.6]

```

1 module stack(clk,load,push,pop,d,qtop,qnext);
2     parameter N=16;
3
4     input clk,load,push,pop;    // 1ビットの入力ポート
5     input [N-1:0] d;    // Nビットの入力ポートd
6     output [N-1:0] qtop, qnext; //
7     // Nビットの出力ポートqtop, qnext
8     reg [N-1:0] qtop, qnext; // Nビットの変数qtop,
9     // qnext
10    reg [N-1:0] q [0:3];    // Nビットの変数q[0],q
11    // [1],q[2],q[3]
12
13    always @(q[0], q[1]) begin
14        qtop <= q[0];    // qtopにq[0]を出力
15        qnext <= q[1];    // qnextにq[1]を出力
16    end
17
18    always @(posedge clk) begin //
19        // clkの立ち上がりで動作
20        if(load) q[0]<=d; else if(pop) q[0]<=q[1];
21        if(push) q[1]<=q[0]; else if(pop) q[1]<=q[2];
22        if(push) q[2]<=q[1]; else if(pop) q[2]<=q[3];
23        if(push) q[3]<=q[2]; else if(pop) q[3]<={N-1'bX
24        };
25    end
26 endmodule

```

---

## 【2.7】

---

```
1 module stack8(clk,load,push,pop,d,qtop,qnext);
2   parameter N=16;
3
4   input clk,load,push,pop;    // 1ビットの入力ポート
5   input [N-1:0] d;           // Nビットの入力ポートd
6   output [N-1:0] qtop, qnext; //
7   // Nビットの出力ポートqtop, qnext
8   reg [N-1:0] q [0:7];       // Nビットの変数q[0],q
9   // [1],q[2],...,q[7]
10
11  assign qtop = q[0]; // qtopにq[0]を出力
12  assign qnext = q[1]; // qnextにq[1]を出力
13
14  always @(posedge clk) begin //
15    // clkの立ち上がりで動作
16    if(load) q [0]<=d; else if(pop) q[0]<=q[1];
17    if(push) q [1]<=q[0]; else if(pop) q[1]<=q[2];
18    if(push) q [2]<=q[1]; else if(pop) q[2]<=q[3];
19    if(push) q [3]<=q[2]; else if(pop) q[3]<=q[4];
20    if(push) q [4]<=q[3]; else if(pop) q[4]<=q[5];
21    if(push) q [5]<=q[4]; else if(pop) q[5]<=q[6];
22    if(push) q [6]<=q[5]; else if(pop) q[6]<=q[7];
23    if(push) q [7]<=q[6]; else if(pop) q[7]<={N{1'
24      bX}};
25  end
26 endmodule
```

---

## 【2.8】

---

```
1 module stackM(clk,load,push,pop,d,qtop,qnext);
2   parameter N=16,M=8;
3
4   input clk,load,push,pop;    // 1ビットの入力ポート
5   input [N-1:0] d;           // Nビットの入力ポートd
6   output [N-1:0] qtop, qnext; //
7   // Nビットの出力ポートqtop, qnext
8   reg [N-1:0] q [0:M-1];     // Nビットの変数q[0],q
9   // [1],q[2],...,q[M-1]
10
11  assign qtop = q[0]; // qtopにq[0]を出力
```

---

```

10 assign qnext = q[1]; // qnextにq[1]を出力
11
12 integer i;
13 always @(posedge clk) begin //
    clkの立ち上がりで動作
14     if(load) q [0]<=d; else if(pop) q[0]<=q[1];
15     for(i=1;i<M-1;i=i+1)
16         if(push) q[i]<=q[i-1]; else if(pop) q[i]<=q[i
            +1];
17     if(push) q[M-1]<=q[M-2]; else if(pop) q[M-1]<={
        N{1'bX}};
18 end
19
20 endmodule

```

---

### 3章

#### 【3.1】

(1)

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

(2)

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

(3)

```
[(a+b*c<d)&&!(a>d-e)] → [(a+b*c<d)] [!(a>d-e)]&&
→ [(a+b*c<d)] [(a>d-e)]!&&
→ [(a+b*c<d)] [a] [d-e]>!&&
→ [(a+b*c<d)] [a] [d] [e]->!&&
→ [a+b*c] [d]<[a] [d] [e]->!&&
→ [a] [b*c]+[d]<[a] [d] [e]->!&&
→ [a] [b] [c]*+[d]<[a] [d] [e]->!&&
→ abc*+d<ade->!&&
```

### 【3.2】

(1)

```
[1+(2+3*4-5)-6] → [1+(2+3*4-5)] [6]-
→ [1] [2+3*4-5]+[6]-
→ [1] [2+3*4] [5]-+[6]-
→ [1] [2] [3*4]+[5]-+[6]-
→ [1] [2] [3] [4]*+[5]-+[6]-
→ 1234*+5-+6-
```

---

```
1 'include "alu_d.v"
2 module opstack_tb1;
3
4 reg clk,num,op; // 1ビットの変数clk,num,op
5 reg [15:0] x; // 16ビットの変数x
6 wire [15:0] qtop; // 16ビットの配線qtop
7
8 opstack opstack0(clk,num,op,x,qtop);
9
10 initial clk=0;
11 always #50 clk=~clk;
12
13 initial begin // 後置記法のオペランド・演算を
14     xに代入
15     $dumpvars;
16     num = 1; op = 0; x = 1; #100
17     num = 1; op = 0; x = 2; #100
18     num = 1; op = 0; x = 3; #100
19     num = 1; op = 0; x = 4; #100
20     num = 0; op = 1; x = 'MUL; #100
```

```

20     num = 0; op = 1; x = 'ADD; #100
21     num = 1; op = 0; x = 5; #100
22     num = 0; op = 1; x = 'SUB; #100
23     num = 0; op = 1; x = 'ADD; #100
24     num = 1; op = 0; x = 6; #100
25     num = 0; op = 1; x = 'SUB; #100
26     $finish;
27     end
28
29 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>&&||

```

---

```

1  'include "alu_d.v"
2  module opstack_tb1;
3
4  reg clk,num,op; // 1ビットの変数clk,num,op
5  reg [15:0] x;   // 16ビットの変数x
6  wire [15:0] qtop; // 16ビットの配線qtop
7
8  opstack opstack0(clk,num,op,x,qtop);
9
10 initial clk=0;
11 always #50 clk=~clk;
12
13 initial begin // 後置記法のオペランド・演算を
                xに代入
14     $dumpvars;
15     num = 1; op = 0; x = 1; #100
16     num = 1; op = 0; x = 2; #100
17     num = 1; op = 0; x = 3; #100
18     num = 1; op = 0; x = 4; #100

```



```

19     num = 0; op = 1; x = 'MUL; #100
20     num = 0; op = 1; x = 'ADD; #100
21     num = 1; op = 0; x = 5; #100
22     num = 0; op = 1; x = 'SUB; #100
23     num = 0; op = 1; x = 'ADD; #100
24     num = 1; op = 0; x = 6; #100
25     num = 0; op = 1; x = 'SUB; #100
26     $finish;
27 end
28
29 endmodule

```

---

### 【3.4】

```

1 'include "state_d.v"
2 module fetch(clk,rst_n,run,irout);
3
4     input clk,rst_n,run;    // 1ビットの入力ポート
5     output [15:0] irout;    // 16ビットの出力ポート
6     irout
7     wire [1:0] cs; // 2ビットの配線cs
8     wire [11:0] pcout; // 12ビットの配線pcout
9     wire [15:0] ramout; // 16ビットの配線ramout
10
11     state state0(.clk(clk),.rst_n(rst_n),.run(run
12         ),.halt(cs=='EXEC && ramout==16'h0000),.q(cs
13         ));
14     counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load
15         (1'b0),.inc(cs=='FETCH),.q(pcout));
16     counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load
17         (cs=='FETCH),.inc(1'b0),.d(ramout),.q(irout
18         ));
19     ram ram0(.clk(clk),.load(1'b0),.addr(pcout),.q(
20         ramout));
21
22 endmodule

```

---

### 【3.6】

```

1 'include "state_d.v"
2 module formula(clk,rst_n,run,qtop);
3
4     input clk,rst_n,run;    // 1ビットの入力ポート

```

```

5     output [15:0] qtop;      // 16ビットの出力ポート
        qtop
6     wire [1:0] cs; // 2ビットの配線cs
7     wire [11:0] pcout; // 12ビットの配線pcout
8     wire [15:0] ramout, irout; // 16ビットの配線
        ramout, irout
9     wire num, op; // 1ビットの配線num, op
10
11     state state0(.clk(clk), .rst_n(rst_n), .run(run
        ), .halt(cs=='EXEC' && ramout==16'hFFFF), .q(cs
        ));
12     counter #(12) pc0(.clk(clk), .rst_n(rst_n), .load
        (1'b0), .inc(cs=='FETCH'), .q(pcout));
13     counter #(16) ir0(.clk(clk), .rst_n(rst_n), .load
        (cs=='FETCH'), .inc(1'b0), .d(ramout), .q(irout
        ));
14     ram ram0(.clk(clk), .load(1'b0), .addr(pcout), .q(
        ramout));
15     opstack opstack0(.clk(clk), .num(num), .op(op), .x
        (irout), .qtop(qtop));
16
17     assign num=(cs=='EXEC')&&~irout[15];
18     assign op=(cs=='EXEC')&&irout[15];
19
20 endmodule

```

---

### 【3.7】

```

1 reg [15:0] bus; // 16ビットの変数バス
2
3 always @(a,b,c,asel,bsel,csel) // busの値を決める
4     case({asel,bsel,csel})
5         3'b100: bus=a; // aselが1のときはa
6         3'b010: bus=b; // bselが1のときはb
7         3'b001: bus=c; // cselが1のときはc
8         default: bus=16'hXXXX;
9     endcase

```

---

### 【3.8】

```

1 'include "state_d.v"
2 module exfetch(clk,rst_n,run,halt);
3

```

```

4   input clk,rst_n,run,halt;    // 1ビットの入力ポート
5   wire [1:0] cs; // 2ビットの配線cs
6   wire [11:0] pcout; // 12ビットの配線pcout
7   wire [15:0] irout,ramout; // 16ビットの配線irout,
   ramout
8   reg [15:0] dbus;
9   wire [11:0] abus;
10
11  state state0(.clk(clk),.rst_n(rst_n),.run(run),.
   halt(halt),.q(cs));
12  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load
   (1'b0),.inc(cs=='FETCH'),.q(pcout));
13  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(
   cs=='FETCH'),.inc(1'b0),.d(ramout),.q(irout));
14  ram ram0(.clk(clk),.load(cs=='EXEC'),.addr(abus),.
   d(dbus),.q(ramout));
15
16
17  assign abus=((cs=='FETCH)?pcout:12'hZZZ);
18  assign abus=((cs=='EXEC'){4'h0,irout[15:8]}:12'
   hZZZ);
19
20  always @(cs,ramout,irout)
21    if(cs=='FETCH) dbus=ramout;
22    else if(cs=='EXEC) dbus={8'h00,irout[7:0]};
23    else dbus=12'hXXX;
24
25  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】

---

```

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

```

```

4 module tinycpu(clk,rst_n,run,out);
5
6   input clk,rst_n,run;    // 1ビットの入力ポート
7   output [15:0] out;    // 16ビットの出力ポートout
8   wire [1:0] cs;    // 2ビットの配線cs
9   wire [11:0] pcout;    // 12ビットの配線pcout
10  wire [15:0] irout,ramout,qtop,qnext,aluout;
11  reg [15:0] dbus;    // 12ビットの変数dbus
12  wire [11:0] abus;    // 12ビットの配線abus
13  reg halt,pcinc,push,pop,abus2pc,dbus2ir,
    dbus2stack,dbus2ram,dbus2out,pc2abus,ir2abus,
    alu2dbus,ir2dbus,stack2dbus,ram2dbus; //1ビッ
    トの変数
14
15  state state0(.clk(clk),.rst_n(rst_n),.run(run),.
    halt(halt),.q(cs));
16  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(
    abus2pc),.inc(pcinc),.d(abus),.q(pcout));
17  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(
    dbus2ir),.inc(1'b0),.d(dbus),.q(irout));
18  alu alu0(.a(qtop),.b(qnext),.f(irout[4:0]),.s(
    aluout));
19  stack stack0(.clk(clk),.load(dbus2stack),.push(
    push),.pop(pop),.d(dbus),.qtop(qtop),.qnext(
    qnext));
20  ram ram0(.clk(clk),.load(dbus2ram),.addr(abus),.d
    (dbus),.q(ramout));
21  counter #(16) out0(.clk(clk),.rst_n(rst_n),.load(
    dbus2out),.inc(1'b0),.d(dbus),.q(out));
22
23  assign abus = pc2abus?pcout:12'hZZZ;
24  assign abus = ir2abus?irout[11:0]:12'hZZZ;
25
26  always @(ram2dbus,ramout,ir2dbus,irout,stack2dbus
    ,qtop,alu2dbus,aluout)
27    if(ram2dbus) dbus = ramout;
28    else if(ir2dbus) dbus = {{4{irout[11]}},irout
    [11:0]};
29    else if(stack2dbus) dbus = qtop;
30    else if(alu2dbus) dbus = aluout;
31    else dbus = 16'hXXXX;
32
33  always @(cs,irout,qtop) // 制御線の値の決定
34  begin
35    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;
36     if(cs=='FETCH)
37     begin
38         pc2abus=1; ram2dbus=1; dbus2ir=1; pcinc=1;
39     end
40     else if(cs=='EXEC)
41         case(irout[15:12])
42             'HALT: halt=1;
43             'PUSHI: begin ir2dbus=1; dbus2stack=1; push
                    =1; end
44             'PUSH: begin ir2abus=1; ram2dbus=1;
                    dbus2stack=1; push=1; end
45             'POP: begin ir2abus=1; stack2dbus=1;
                    dbus2ram=1; pop=1; end
46             'JMP: begin ir2abus=1; abus2pc=1; end
47             'JZ: begin pop=1;
48                 if(qtop==0) begin ir2abus=1; abus2pc
                    =1; end
49                 end
50             'JNZ: begin pop=1;
51                 if(qtop!=0) begin ir2abus=1; abus2pc
                    =1; end
52                 end
53             'OUT: begin stack2dbus=1; dbus2out=1; pop
                    =1; end
54             'OP: begin alu2dbus=1; dbus2stack=1;
55                 if(irout[4]==0) pop=1;
56             end
57         endcase
58     end
59
60 endmodule

```

---

### 【4.3】

```

1  'include "state_d.v"    // ステートマシン回路の定数
    定義
2  'include "alu_d.v"     // 算術論理演算回路の定数定
    義
3  'include "inst.v"      // 機械語コードの定義
4  module tinycpu(clk,rst_n,run,in,out);
5
6      input clk,rst_n,run;    // 1ビットの入力ポート
7      input [15:0] in;      // 16ビットの入力ポート

```

```

8   output [15:0] out; // 16ビットの出力ポートout
9   wire [1:0] cs; // 2ビットの配線cs
10  wire [11:0] pcout; // 12ビットの配線pcout
11  wire [15:0] dbus,irout,ramout,qtop,qnext,aluout;
12  reg [11:0] abus; // 12ビットの変数abus
13  reg halt,pcinc,push,pop,abus2pc,dbus2ir,
    dbus2stack,dbus2ram,dbus2out,pc2abus,ir2abus,
    alu2dbus,ir2dbus,stack2dbus,ram2dbus; //1ビッ
    トの変数
14  reg in2dbus;
15
16  state state0(.clk(clk),.rst_n(rst_n),.run(run),.
    halt(halt),.q(cs));
17  counter #(12) pc0(.clk(clk),.rst_n(rst_n),.load(
    abus2pc),.inc(pcinc),.d(abus),.q(pcout));
18  counter #(16) ir0(.clk(clk),.rst_n(rst_n),.load(
    dbus2ir),.inc(1'b0),.d(dbus),.q(irout));
19  alu alu0(.a(qtop),.b(qnext),.f(irout[4:0]),.s(
    aluout));
20  stack stack0(.clk(clk),.load(dbus2stack),.push(
    push),.pop(pop),.d(dbus),.qtop(qtop),.qnext(
    qnext));
21  ram ram0(.clk(clk),.load(dbus2ram),.addr(abus),.d
    (dbus),.q(ramout));
22  counter #(16) out0(.clk(clk),.rst_n(rst_n),.load(
    dbus2out),.inc(1'b0),.d(dbus),.q(out));
23
24  always @ (pc2abus,ir2abus,pcout,irout)
25    if(pc2abus) abus = pcout;
26    else if(ir2abus) abus = irout[11:0];
27    else abus = 12'hXXX;
28
29  assign dbus = ram2dbus?ramout:16'hZZZZ;
30  assign dbus = ir2dbus?{4{irout[11]}},irout
    [11:0]:16'hZZZZ;
31  assign dbus = stack2dbus?qtop:16'hZZZZ;
32  assign dbus = alu2dbus?aluout:16'hZZZZ;
33  assign dbus = in2dbus?in:16'hZZZZ;
34
35  always @(cs,irout,qtop) // 制御線の値の決定
36  begin
37    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;
38    in2dbus=0;

```

```

39     if(cs=='FETCH)
40     begin
41         pc2abus=1; ram2dbus=1; dbus2ir=1; pcinc=1;
42     end
43     else if(cs=='EXEC)
44         case(irout[15:12])
45             'HALT: halt=1;
46             'PUSHI: begin ir2dbus=1; dbus2stack=1; push
47                     =1; end
48             'PUSH: begin ir2abus=1; ram2dbus=1;
49                     dbus2stack=1; push=1; end
50             'POP: begin ir2abus=1; stack2dbus=1;
51                   dbus2ram=1; pop=1; end
52             'JMP: begin ir2abus=1; abus2pc=1; end
53             'JZ: begin pop=1;
54                   if(qtop==0) begin ir2abus=1; abus2pc
55                           =1; end
56                   end
57             'JNZ: begin pop=1;
58                   if(qtop!=0) begin ir2abus=1;
59                           abus2pc=1; end
60                   end
61             'OUT: begin stack2dbus=1; dbus2out=1; pop
62                   =1; end
63             'OP: begin alu2dbus=1; dbus2stack=1;
64                   if(irout[4]==0) pop=1;
65                   end
66             'IN: begin in2dbus=1; dbus2stack=1; push=1;
67                   end
68         endcase
69     end
70 endmodule

```

---

#### 【4.4】

```

1  'include "state_d.v"    // ステートマシン回路の定数
   定義
2  'include "alu_d.v"     // 算術論理演算回路の定数定
   義
3  'include "inst.v"     // 機械語コードの定義
4  module tinycpu(clk,rst_n,run,out);
5
6      input clk,rst_n,run;    // 1ビットの入力ポート
7      output [15:0] out;    // 16ビットの出力ポートout

```

```

8  wire [1:0] cs; // 2ビットの配線cs
9  wire [11:0] pcout; // 12ビットの配線pcout
10 wire [15:0] dbus, irout, ramout, qtop, qnext, aluout;
11 reg [11:0] abus; // 12ビットの変数abus
12 wire halt, pcinc, push, pop, abus2pc, dbus2ir,
    dbus2stack, dbus2ram, dbus2out, pc2abus, ir2abus,
    alu2dbus, ir2dbus, stack2dbus, ram2dbus; //1ビッ
    トの配線
13
14 state state0(.clk(clk), .rst_n(rst_n), .run(run), .
    halt(halt), .q(cs));
15 counter #(12) pc0(.clk(clk), .rst_n(rst_n), .load(
    abus2pc), .inc(pcinc), .d(abus), .q(pcout));
16 counter #(16) ir0(.clk(clk), .rst_n(rst_n), .load(
    dbus2ir), .inc(1'b0), .d(dbus), .q(irout));
17 alu alu0(.a(qtop), .b(qnext), .f(irout[4:0]), .s(
    aluout));
18 stack stack0(.clk(clk), .load(dbus2stack), .push(
    push), .pop(pop), .d(dbus), .qtop(qtop), .qnext(
    qnext));
19 ram ram0(.clk(clk), .load(dbus2ram), .addr(abus), .d
    (dbus), .q(ramout));
20 counter #(16) out0(.clk(clk), .rst_n(rst_n), .load(
    dbus2out), .inc(1'b0), .d(dbus), .q(out));
21
22 always @ (pc2abus, ir2abus, pcout, irout)
23     if(pc2abus) abus = pcout;
24     else if(ir2abus) abus = irout[11:0];
25     else abus = 12'hXXX;
26
27 assign dbus = ram2dbus?ramout:16'hZZZZ;
28 assign dbus = ir2dbus?{{4{irout[11]}} , irout
    [11:0]}:16'hZZZZ;
29 assign dbus = stack2dbus?qtop:16'hZZZZ;
30 assign dbus = alu2dbus?aluout:16'hZZZZ;
31
32 assign halt = (cs=='EXEC && irout[15:12]=='HALT)
    ? 1'b1 : 1'b0;
33 assign pcinc = (cs=='FETCH) ? 1'b1 : 1'b0;
34 assign push = (cs=='EXEC && (irout[15:12]=='PUSHI
    || irout[15:12]=='PUSH)) ? 1'b1 : 1'b0;
35 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;

```



```

36  assign abus2pc = (cs=='EXEC && (irout[15:12]==
      'JMP || (irout[15:12]=='JZ && qtop==0) || (
        irout[15:12]=='JNZ && qtop!=0))) ? 1'b1 : 1'b0
      ;
37  assign dbus2ir = (cs=='FETCH) ? 1'b1 : 1'b0;
38  assign dbus2stack = (cs=='EXEC && (irout[15:12]==
      'PUSHI || irout[15:12]=='PUSH || irout
        [15:12]=='OP)) ? 1'b1 : 1'b0;
39  assign dbus2ram = (cs=='EXEC && irout[15:12]==
      'POP) ? 1'b1 : 1'b0;
40  assign dbus2out = (cs=='EXEC && irout[15:12]==
      'OUT) ? 1'b1 : 1'b0;
41  assign pc2abus = (cs=='FETCH) ? 1'b1 : 1'b0;
42  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
      ;
43  assign alu2dbus = (cs=='EXEC && irout[15:12]=='OP
      ) ? 1'b1 : 1'b0;
44  assign ir2dbus = (cs=='EXEC && irout[15:12]==
      'PUSHI) ? 1'b1 : 1'b0;
45  assign stack2dbus = (cs=='EXEC && (irout[15:12]==
      'POP || irout[15:12]=='OUT)) ? 1'b1 : 1'b0;
46  assign ram2dbus = ((cs=='FETCH) || (cs=='EXEC &&
      irout[15:12]=='PUSH)) ? 1'b1 : 1'b0;
47
48  endmodule

```

---

#### 【4.6】

---

```

1  L1: PUSH n
2     PUSHI 10
3     NE
4     JNZ L2
5     HALT
6  L2: PUSH n
7     OUT
8     PUSH n
9     PUSHI 1
10    ADD
11    POP n
12    JMP L1
13  n: 1

```

---

### 【4.7】

---

```
1      PUSH a1
2      OUT
3  L1:  PUSH a2
4      OUT
5      PUSH a1
6      PUSH a2
7      ADD
8      PUSH a2
9      POP a1
10     POP a2
11     JMP L1
12     a1: 1
13     a2: 1
```

---

## 5章

### 【5.1】

---

```
1      PUSH x
2      PUSH y
3      GT          // x>y
4      JZ L1F
5      PUSH x
6      PUSH z
7      GT          // x>z
8      JZ L2F
9      PUSH x
10     POP t       // t=x
11     JMP L2T
12  L1F: PUSH y
13     PUSH z
14     GT          // y>z
15     JZ L2F
16     PUSH y
17     POP t       // t=y
18     JMP L2T
19  L2F: PUSH z
20     POP t       // t=z
21  L2T:
```

---

## **[5.2]**

(1)

---

```
1 do{
2   out(n);
3   n=n-1;
4 }while(n>0);
5 halt;
6 int n=10;
```

---

(2)

---

```
1 do{
2   if(n&1){
3     out(n*3);
4   }else{
5     out(n>>1);
6   }
7   n=n-1;
8 }while(n>0);
9 halt;
10 int n=10;
```

---

(3)

---

```
1 m=0;
2 while(n<=3*m){
3   m=m+1;
4 }
5 out(m);
6 n = n-3*m;
7 out(n);
8 halt;
9 int n=10;
10 int m;
```

---

(4)

---

```
1 do{
2   if(a<b){b=a^b; a=a^b; b=a^b;}
3   out(a);
4   a=a-b;
```

```
5 }while(b);
6 halt;
7 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

---

```
1     A の実行
2 L1T: B の計算
3     JZ L1F
4     D の実行
5     C の実行
6     JMP L1T
7 L1F:
```

---

```
1 // for(n=1;n<10;n=n+1){out(n);} の変換
2     PUSHI 1
3     POP n    // n=1
4 L1T: PUSH n
5     PUSHI 10
6     LT      // n<10
7     JZ L1F
8     PUSH n
9     OUT     // out(n)
10    PUSH n
11    PUSHI 1
12    ADD
13    POP n   // n=n+1
14 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

---

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

---

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

---

```

4      LT          // x<0
5      JZ L1F
6      PUSH x
7      NEG         // -x
8      JMP L1T
9 L1F: PUSH x     // x
10 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

```

1      式1 の 計算
2      JZ L1F
3      式2 の 計算
4      JMP L1T
5 L1F: PUSHI 0
6 L1T:

```

---

```

1 // if((5<x)&&(x<10)){out(n);}
2   PUSHI 5
3   PUSH x
4   LT          // 5<x
5   JZ L1F
6   PUSH x
7   PUSHI 10
8   LT          // x<10
9   JMP L1T
10 L1F: PUSHI 0
11 L1T: JZ L2F
12   PUSH n
13   OUT        // out(n)
14 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

```

1      式1 の 計算
2      JNZ L1T
3      式2 の 計算
4      JMP L1F
5 L1T: PUSHI 1
6 L1F:

```

---

---

```

1 // if((x<5)||(10<x)){out(n);}
2     PUSH x
3     PUSHI 5
4     LT          // x<5
5     JNZ L1T
6     PUSHI 10
7     PUSH x
8     LT          // 10<x
9     JMP L1F
10 L1T: PUSHI 1
11 L1F: JZ L2F
12     PUSH n
13     OUT          // out(n)
14 L2F:

```

---

## 6章

### 【6.1】

---

```

1 #!/usr/bin/perl -W
2
3 %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);
4
5 $addr=0;
6 while(<>){
7     push @source,$_;
8     if(s/^\s*(\w+)://){
9         $label{$1}=$addr;
10        printf("%s:%03X\n",$1,$label{$1});#ラベルの表示
11    }
12    if(/^\s*(-?\d+[A-Z]+)/){$addr++;}
13 }
14
15 $addr=0;
16 foreach(@source){
17     $m=$_;

```

```

18 s/^\s*\w+://;
19 if(/^\s*PUSHI\s+(-?\d+)/){
20     printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {PUSHI}+($1&0xff));
21 }elseif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
22     printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {$1}+$label{$2});
23 }elseif(/^\s*([A-Z]+)/){
24     printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {$1});
25 }elseif(/^\s*(-?\d+)/){
26     printf("mem[12'h%03X]=16'h%04X;", $addr++, $1&0
           xffff);
27 }else{
28     printf("%21s", $m);
29 }
30 printf(" // %s", $m);
31 }

```

---

## [6.2]

---

```

1 #!/usr/bin/perl -W
2
3 %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);
4
5 $addr=0;
6 while(<>){
7     push @source, $_;
8     if(s/^\s*(\w+)://){$label{$1}=$addr;}
9     if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
10 }
11
12 $addr=0;
13 $line=0;
14 foreach(@source){
15     $line++;
16     $m=$_;
17     s/^\s*\w+://;

```

```

18  if (/^\s*PUSHI\s+(-?\d+)/){
19      printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {PUSHI}+($1&0xff));
20  }elseif (/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
21      printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {$1}+$label{$2});
22  }elseif (/^\s*([A-Z]+)/){
23      if(!defined($MCODE{$1}))){
24          printf("ERROR! line %d: instruction %s is not
           defined.\n", $line, $1);
25          exit(1);
26      }
27      printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
           {$1});
28  }elseif (/^\s*(-?\d+)/){
29      printf("mem[12'h%03X]=16'h%04X;", $addr++, $1&0
           xffff);
30  }else{
31      printf("%21s", $m);
32  }
33  printf(" // %s", $m);
34  }

```

---

### [6.3]

---

```

1  #!/usr/bin/perl -W
2
3  %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);
4
5  $addr=0;
6  while(<>){
7      push @source, $_;
8      if(s/^\s*(\w+)://){
9          $label{$1}=$addr;
10     }
11     if (/^\s*(-?\d+|[A-Z]+)/){$addr++;}
12 }
13

```



```

14 $addr=0;
15 $line=0;
16 foreach(@source){
17     $line++;
18     $m=$_;
19     s/^\s*\w+://;
20     if(/^\s*PUSHI\s+(-?\d+)/){
21         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {PUSHI}+($1&0xff));
22     }elseif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
23         if(!defined($label{$2})) {
24             printf("ERROR! line %d: label %s is not
                    defined.\n", $line, $2);
25             exit(1);
26         }
27         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {$1}+$label{$2});
28     }elseif(/^\s*([A-Z]+)/){
29         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {$1});
30     }elseif(/^\s*(-?\d+)/){
31         printf("mem[12'h%03X]=16'h%04X;", $addr++, $1&0
                xffff);
32     }else{
33         printf("%21s", $m);
34     }
35     printf(" // %s", $m);
36 }

```

---

## 【6.4】

---

```

1 #!/usr/bin/perl -W
2
3 %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);
4
5 $addr=0;
6 while(<>){
7     push @source, $_;

```

```

8   if(s/^\s*(\w+)://){
9       $label{$1}=$addr;
10  }
11  if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
12 }
13
14 $addr=0;
15 $line=0;
16 foreach(@source){
17     $line++;
18     $m=$_;
19     s/^\s*\w+://;
20     if(/^\s*PUSHI\s+(-?\d+)/){
21         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
22             {PUSHI}+($1&0xfff));
23     }elseif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
24         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
25             {$1}+$label{$2});
26     }elseif(/^\s*(PUSH|POP|JMP|JZ|JNZ)/){
27         printf("ERROR! line %d: operand for %s is
28             missing.\n", $line, $1);
29         exit(1);
30     }elseif(/^\s*([A-Z]+)/){
31         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
32             {$1});
33     }elseif(/^\s*(-?\d+)/){
34         printf("mem[12'h%03X]=16'h%04X;", $addr++, $1&0
35             xffff);
36     }else{
37         printf("%21s", $m);
38     }
39     printf(" // %s", $m);
40 }

```

---

## [6.5]

```

1  #!/usr/bin/perl -W
2
3  %MCODE=(HALT=>0x0000 , PUSHI=>0x1000 , PUSH=>0x2000 , POP
4      =>0x3000 , JMP=>0x4000 , JZ=>0x5000 , JNZ=>0x6000 , IN
5      =>0xD000 , OUT=>0xE000 , ADD=>0xF000 , SUB=>0xF001 , MUL
6      =>0xF002 , SHL=>0xF003 , SHR=>0xF004 , BAND=>0xF005 ,
7      BOR=>0xF006 , BXOR=>0xF007 , AND=>0xF008 , OR=>0xF009 ,
8      EQ=>0xF00A , NE=>0xF00B , GE=>0xF00C , LE=>0xF00D , GT

```

```

=>0xF00E,LE=>0xF00F,NEG=>0xF010,BNOT=>0xF011,NOT
=>0xF012);
4
5 $addr=0;
6 while(<>){
7     push @source,$_;
8     if(s/^\s*(\w+)://){
9         $label{$1}=$addr;
10    }
11    if(/^\s*(-?\d+|[A-Z]+)/){$addr++;}
12 }
13
14 $addr=0;
15 $line=0;
16 foreach(@source){
17     $line++;
18     $m=$_;
19     s/^\s*\w+://;
20     if(/^\s*PUSHI\s+(-?\d+)/){
21         if(!(-2048<=$1 && $1<=2047)){
22             printf("ERROR! line %d: operand %d of PUSHI
                is out of range.\n",$line,$1);
23             exit(1);
24         }
25         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {PUSHI}+($1&0xfff));
26     }elseif(/^\s*(PUSH|POP|JMP|JZ|JNZ)\s+(\w+)/){
27         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {$1}+$label{$2});
28     }elseif(/^\s*([A-Z]+)/){
29         printf("mem[12'h%03X]=16'h%04X;", $addr++, $MCODE
                {$1});
30     }elseif(/^\s*(-?\d+)/){
31         printf("mem[12'h%03X]=16'h%04X;", $addr++, $1&0
                xffff);
32     }else{
33         printf("%21s", $m);
34     }
35     printf(" // %s", $m);
36 }

```

---

## 7章

### 【7.1】

---

```

1  %{
2  int s[4];
3  void printstack();
4  void push(int);
5  int pop();
6  void add();
7  void sub();
8  void mul();
9  void shl();
10 void shr();
11 void band();
12 void bor();
13 void bxor();
14 void and();
15 void or();
16 void eq();
17 void ne();
18 void ge();
19 void le();
20 void gt();
21 void lt();
22 void neg();
23 void bnot();
24 void not();
25 %}
26 %%
27 \n printf("result=%d\n",pop());
28 [0-9]+ {push(atoi(yytext)); printstack();}
29 \+ {add(); printstack();}
30 \- {sub(); printstack();}
31 \* {mul(); printstack();}
32 \>\> {shl(); printstack();}
33 \<\< {shr(); printstack();}
34 & {band(); printstack();}
35 \| {bor(); printstack();}
36 \^ {bxor(); printstack();}
37 \&\& {and(); printstack();}
38 \|\|\| {or(); printstack();}
39 == {eq(); printstack();}
40 != {ne(); printstack();}
41 \>= {ge(); printstack();}
42 \<= {le(); printstack();}
43 \> {gt(); printstack();}
44 \< {lt(); printstack();}
45 \@ {neg(); printstack();}

```

```

46  ~ {bnot(); printstack();}
47  ! {not(); printstack();}
48  .
49  %%
50  void printstack(){printf("%d %d %d %d\n",s[0],s[1],
    s[2],s[3]);}
51  void push(int x){s[3]=s[2];s[2]=s[1];s[1]=s[0];s
    [0]=x;}
52  int pop(){int x=s[0];s[0]=s[1];s[1]=s[2];s[2]=s[3];
    return(x);}
53  void add(){s[0]=s[1]+s[0];s[1]=s[2];s[2]=s[3];}
54  void sub(){s[0]=s[1]-s[0];s[1]=s[2];s[2]=s[3];}
55  void mul(){s[0]=s[1]*s[0];s[1]=s[2];s[2]=s[3];}
56  void shl(){s[0]=s[1]<<s[0];s[1]=s[2];s[2]=s[3];}
57  void shr(){s[0]=s[1]>>s[0];s[1]=s[2];s[2]=s[3];}
58  void band(){s[0]=s[1]&s[0];s[1]=s[2];s[2]=s[3];}
59  void bor(){s[0]=s[1]|s[0];s[1]=s[2];s[2]=s[3];}
60  void bxor(){s[0]=s[1]^s[0];s[1]=s[2];s[2]=s[3];}
61  void and(){s[0]=s[1]&& s[0];s[1]=s[2];s[2]=s[3];}
62  void or(){s[0]=s[1]||s[0];s[1]=s[2];s[2]=s[3];}
63  void eq(){s[0]=s[1]==s[0];s[1]=s[2];s[2]=s[3];}
64  void ne(){s[0]=s[1]!=s[0];s[1]=s[2];s[2]=s[3];}
65  void ge(){s[0]=s[1]>=s[0];s[1]=s[2];s[2]=s[3];}
66  void le(){s[0]=s[1]<=s[0];s[1]=s[2];s[2]=s[3];}
67  void gt(){s[0]=s[1]>s[0];s[1]=s[2];s[2]=s[3];}
68  void lt(){s[0]=s[1]<s[0];s[1]=s[2];s[2]=s[3];}
69  void neg(){s[0]=-s[0];}
70  void bnot(){s[0]=~s[0];}
71  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

```

1  %{
2  #include <string.h>
3  #include "y.tab.h"
4  int n=0;
5  %}
6  %%
7  [ \t\r\n]
8  &&    {return(AND);}
9  \|\|  {return(OR);}
10 ==    {return(EQ);}
11 !=    {return(NE);}
12 \>=   {return(GE);}
13 \<=   {return(LE);}
14 \<<   {return(SHL);}
15 \>>   {return(SHR);}
16 if    {yylval.n=++n;return(IF);}
17 while {yylval.n=++n;return(WHILE);}
18 do    {yylval.n=++n;return(DO);}

```

```

19 int    {return(INT);}
20 else  {return(ELSE);}
21 halt  {return(HALT);}
22 out   {return(OUT);}
23 \+\+  {return(INC);}
24 \-\-  {return(DEC);}
25 [0-9]+ {yyval.n=atoi(yytext);return(NUM);}
26 [a-zA-Z][a-zA-Z0-9]* {yyval.s=strdup(yytext);
    return(NAME);}
27 .      {return(yytext[0]);}
28 %%

```

---

tinyc.y

---

```

1  %{
2  #include <stdio.h>
3  extern int yylex();
4  int yyerror(const char *s);
5  %}
6  %union {int n; char *s;}
7  %token <s> NAME
8  %token <n> NUM IF WHILE DO
9  %token INT ELSE HALT OUT
10 %type <n> if
11 %left OR
12 %left AND
13 %left '|'
14 %left '^'
15 %left '&'
16 %left EQ NE
17 %left GE LE '<' '>'
18 %left SHL SHR
19 %left '+' '-'
20 %left '*'
21 %right '!' '~' NEG
22 %left INC DEC
23 %%
24 stmts: |stmts stmt
25 ;
26 stmt: intdef|ifelse|while|do|halt|out|assign|inc|
    dec
27 ;
28 intdef: INT intlist ','
29 ;
30 intlist: integer
31 | intlist ',' integer

```

```

32 ;
33 integer: NAME {printf("%s:\t0\n", $1); free($1);}
34 | NAME '=' NUM {printf("%s:\t%d\n", $1, $3); free($1
    );}
35 | NAME '=' '-' NUM {printf("%s:\t%d\n", $1, -$4); free
    ($1);}
36 ;
37 ifelse: if {printf("L%dF:\n", $1);}
38 | if {printf("\tJMP L%OdT\nL%OdF:\n", $1, $1);} ELSE
    '{' stmts '}' {printf("L%OdT:\n", $1);}
39 ;
40 if: IF '(' expr ')' {printf("\tJZ L%OdF\n", $1);}
    '{' stmts '}' {$$=$1;}
41 ;
42 while: WHILE {printf("L%OdT:\n", $1);} '(' expr ')'
    {printf("\tJZ L%OdF\n", $1);} '{' stmts '}' {
    printf("\tJMP L%OdT\nL%OdF:\n", $1, $1);}
43 ;
44 do: DO {printf("L%OdT:\n", $1);} '{' stmts '}' WHILE
    '(' expr ')' ';' {printf("\tJNZ L%OdT\n", $1);}
45 ;
46 halt: HALT ';' {printf("\tHALT\n");}
47 ;
48 out: OUT '(' expr ')' ';' {printf("\tOUT\n");}
49 ;
50 assign: NAME '=' expr ';' {printf("\tPOP %s\n", $1);
    free($1);}
51 ;
52 inc: NAME INC ';' {printf("\tPUSH %s\n\tPUSHI 1\n\
    tADD\n\tPOP %s\n", $1, $1); free($1);}
53 | INC NAME ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tADD
    \n\tPOP %s\n", $2, $2); free($2);}
54 ;
55 dec: NAME DEC ';' {printf("\tPUSH %s\n\tPUSHI 1\n\
    tSUB\n\tPOP %s\n", $1, $1); free($1);}
56 | DEC NAME ';' {printf("\tPUSH %s\n\tPUSHI 1\n\tSUB
    \n\tPOP %s\n", $2, $2); free($2);}
57 ;
58 expr: NAME {printf("\tPUSH %s\n", $1); free($1);}
59 | NUM {printf("\tPUSHI %d\n", $1);}
60 | '!' expr {printf("\tNOT\n");}
61 | '~' expr {printf("\tNBNOT\n");}
62 | '-' expr %prec NEG {printf("\tNEG\n");}
63 | expr '+' expr {printf("\tADD\n");}
64 | expr '-' expr {printf("\tSUB\n");}
65 | expr '*' expr {printf("\tMUL\n");}

```



```

66 | expr AND expr {printf("\tAND\n");}
67 | expr OR expr {printf("\tOR\n");}
68 | expr '&' expr {printf("\tBAND\n");}
69 | expr '|' expr {printf("\tBOR\n");}
70 | expr '^' expr {printf("\tBXOR\n");}
71 | expr SHL expr {printf("\tSHL\n");}
72 | expr SHR expr {printf("\tSHR\n");}
73 | expr EQ expr {printf("\tEQ\n");}
74 | expr NE expr {printf("\tNE\n");}
75 | expr GE expr {printf("\tGE\n");}
76 | expr LE expr {printf("\tLE\n");}
77 | expr '<' expr {printf("\tLT\n");}
78 | expr '>' expr {printf("\tGT\n");}
79 | '(' expr ')'
80 ;
81 %%
82 int yyerror(const char *s){printf("%s\n",s);}
83 int main(){return(yyparse());}

```

---

## [7.7]

tinyc.1 はリスト 7.7 と同じ。

tinyc.y

---

```

1  %{
2  #include <stdio.h>
3  extern int yylex();
4  int yyerror(const char *s);
5  %}
6  %union {int n; char *s;}
7  %token <s> NAME
8  %token <n> NUM IF WHILE DO AND
9  %token INT ELSE HALT OUT
10 %type <n> if
11 %left OR
12 %left AND
13 %left '|'
14 %left '^'
15 %left '&'
16 %left EQ NE
17 %left GE LE '<' '>'
18 %left SHL SHR
19 %left '+' '-'
20 %left '*'
21 %right '!' '~' NEG

```

```

22 %%
23 stmts: |stmts stmt
24 ;
25 stmt: intdef|ifelse|while|do|halt|out|assign
26 ;
27 intdef: INT intlist ','
28 ;
29 intlist: integer
30 | intlist ',' integer
31 ;
32 integer: NAME {printf("%s:\t0\n", $1); free($1);}
33 | NAME '=' NUM {printf("%s:\t%d\n", $1, $3); free($1
    );}
34 | NAME '=' '-' NUM {printf("%s:\t%d\n", $1, -$4); free
    ($1);}
35 ;
36 ifelse: if {printf("L%dF:\n", $1);}
37 | if {printf("\tJMP L%0dT\nL%0dF:\n", $1, $1);} ELSE
    '{' stmts '}' {printf("L%0dT:\n", $1);}
38 ;
39 if: IF '(' expr ')' {printf("\tJZ L%0dF\n", $1);}
    '{' stmts '}' {'$$=$1;}
40 ;
41 while: WHILE {printf("L%0dT:\n", $1);} '(' expr ')'
    {printf("\tJZ L%0dF\n", $1);} '{' stmts '}' {
    printf("\tJMP L%0dT\nL%0dF:\n", $1, $1);}
42 ;
43 do: DO {printf("L%0dT:\n", $1);} '{' stmts '}' WHILE
    '(' expr ')' ';' {printf("\tJNZ L%0dT\n", $1);}
44 ;
45 halt: HALT ';' {printf("\tHALT\n");}
46 ;
47 out: OUT '(' expr ')' ';' {printf("\tOUT\n");}
48 ;
49 assign: NAME '=' expr ';' {printf("\tPOP %s\n", $1);
    free($1);}
50 ;
51 expr: NAME {printf("\tPUSH %s\n", $1); free($1);}
52 | NUM {printf("\tPUSHI %d\n", $1);}
53 | '!' expr {printf("\tNOT\n");}
54 | '~' expr {printf("\tNBNOT\n");}
55 | '-' expr %prec NEG {printf("\tNEG\n");}
56 | expr '+' expr {printf("\tADD\n");}
57 | expr '-' expr {printf("\tSUB\n");}
58 | expr '*' expr {printf("\tMUL\n");}
59 | expr OR expr {printf("\tOR\n");}

```

```

60 | expr '&' expr {printf("\tBAND\n");}
61 | expr '|' expr {printf("\tBOR\n");}
62 | expr '^' expr {printf("\tBXOR\n");}
63 | expr SHL expr {printf("\tSHL\n");}
64 | expr SHR expr {printf("\tSHR\n");}
65 | expr EQ expr {printf("\tEQ\n");}
66 | expr NE expr {printf("\tNE\n");}
67 | expr GE expr {printf("\tGE\n");}
68 | expr LE expr {printf("\tLE\n");}
69 | expr '<' expr {printf("\tLT\n");}
70 | expr '>' expr {printf("\tGT\n");}
71 | '(' expr ')'
72 | and
73 ;
74 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);}
75 ;
76 %%
77 int yyerror(const char *s){printf("%s\n", s);}
78 int main(){return(yyparse());}

```

---

## [7.8]

tinyc.1 はリスト 7.7 と同じ。

tinyc.y

---

```

1 %{
2 #include <stdio.h>
3 extern int yylex();
4 int yyerror(const char *s);
5 %}
6 %union {int n; char *s;}
7 %token <s> NAME
8 %token <n> NUM IF WHILE DO OR
9 %token INT ELSE HALT OUT
10 %type <n> if
11 %left OR
12 %left AND
13 %left '|'
14 %left '^'
15 %left '&'
16 %left EQ NE
17 %left GE LE '<' '>'
18 %left SHL SHR

```

```

19 %left '+' '-'
20 %left '*'
21 %right '!' '~' NEG
22 %%
23 stmts: |stmts stmt
24 ;
25 stmt: intdef|ifelse|while|do|halt|out|assign
26 ;
27 intdef: INT intlist ','
28 ;
29 intlist: integer
30 | intlist ',' integer
31 ;
32 integer: NAME {printf("%s:\t0\n", $1); free($1);}
33 | NAME '=' NUM {printf("%s:\t%d\n", $1, $3); free($1
    );}
34 | NAME '=' '-' NUM {printf("%s:\t%d\n", $1, -$4); free
    ($1);}
35 ;
36 ifelse: if {printf("L%dF:\n", $1);}
37 | if {printf("\tJMP L%dT\nL%dF:\n", $1, $1);} ELSE
    '{' stmts '}' {printf("L%dT:\n", $1);}
38 ;
39 if: IF '(' expr ')' {printf("\tJZ L%dF\n", $1);}
    '{' stmts '}' {'$$=$1;}
40 ;
41 while: WHILE {printf("L%dT:\n", $1);} '(' expr ')'
    {printf("\tJZ L%dF\n", $1);} '{' stmts '}' {
    printf("\tJMP L%dT\nL%dF:\n", $1, $1);}
42 ;
43 do: DO {printf("L%dT:\n", $1);} '{' stmts '}' WHILE
    '(' expr ')' ';' {printf("\tJNZ L%dT\n", $1);}
44 ;
45 halt: HALT ';' {printf("\tHALT\n");}
46 ;
47 out: OUT '(' expr ')' ';' {printf("\tOUT\n");}
48 ;
49 assign: NAME '=' expr ';' {printf("\tPOP %s\n", $1);
    free($1);}
50 ;
51 expr: NAME {printf("\tPUSH %s\n", $1); free($1);}
52 | NUM {printf("\tPUSHI %d\n", $1);}
53 | '!' expr {printf("\tNOT\n");}
54 | '~' expr {printf("\tNBNOT\n");}
55 | '-' expr %prec NEG {printf("\tNEG\n");}
56 | expr '+' expr {printf("\tADD\n");}

```

```

57 | expr '-' expr {printf("\tSUB\n");}
58 | expr '*' expr {printf("\tMUL\n");}
59 | expr AND expr {printf("\tAND\n");}
60 | expr '&' expr {printf("\tBAND\n");}
61 | expr '|' expr {printf("\tBOR\n");}
62 | expr '^' expr {printf("\tBXOR\n");}
63 | expr SHL expr {printf("\tSHL\n");}
64 | expr SHR expr {printf("\tSHR\n");}
65 | expr EQ expr {printf("\tEQ\n");}
66 | expr NE expr {printf("\tNE\n");}
67 | expr GE expr {printf("\tGE\n");}
68 | expr LE expr {printf("\tLE\n");}
69 | expr '<' expr {printf("\tLT\n");}
70 | expr '>' expr {printf("\tGT\n");}
71 | '(' expr ')'
72 | or
73 ;
74 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);}
75 ;
76 %%
77 int yyerror(const char *s){printf("%s\n", s);}
78 int main(){return(yyparse());}

```

---