Z80PE

LDLABO

2008 年 6 月 12 日

目次

図目次

表目次

第 1 章

論理譜

```
{ ========================================================================= }
{   Z80                                                    }
{                                                                           }
{ ========================================================================= }
logicname Z80

{ ========================================================================= }
{    実効譜                                                }
{ ========================================================================= }
entity main
{ ------------------------------------------------------------------------- }
{    入力                                                  }
{ ------------------------------------------------------------------------- }
input  RESET;
input  DATAIN[8];
input  NMI;
input  INT;
input  WAIT;

{ ------------------------------------------------------------------------- }
{    出力                                                  }
{ ------------------------------------------------------------------------- }
output Q[27];

bitn   ADDRESS[16];
bitn   DATAOUT[8];
bitn   DIR;
bitn   MIO;
bitn   INTA;

{ ------------------------------------------------------------------------- }
{    内部信号                                              }
{ ------------------------------------------------------------------------- }
bitr   regA[8];
bitr   regB[8];
bitr   regC[8];
bitr   regD[8];
bitr   regE[8];
bitr   regH[8];
bitr   regL[8];
bitr   regPC[16];
bitr   regIX[16];
bitr   regIY[16];
bitr   regI[8];
bitr   regR[8];
bitr   regSP[16];

bitr   regRA[8];   { 裏レジスタ }
```

```
bitr   regRB[8];
bitr   regRC[8];
bitr   regRD[8];
bitr   regRE[8];
bitr   regRH[8];
bitr   regRL[8];

bitr   flgS;
bitr   flgZ;
bitr   flgH;
bitr   flgPV;
bitr   flgN;
bitr   flgC;
bitr   flgEI;

bitr   flgRS;   { 裏フラグ }
bitr   flgRZ;
bitr   flgRH;
bitr   flgRPV;
bitr   flgRN;
bitr   flgRC;
bitr   flgREI;

bitr   reg0T[8];   { 補助レジスタ }
bitr   reg1T[8];

bitr   im[2];      { 割り込みモード }

bitr   code0d[8];
bitr   code1d[8];
bitr   code2d[8];
bitr   code3d[8];

bitr   anct[4];
bitr   inst[8];
bitr   expinst[4];
bitr   ccenm;
bitr   codesize[4];
bitr   nmiset[4];
bitr   intset[4];
bitr   intack0data[8];
bitr   intack1data[8];
bitr   intack2data[8];
bitr   inttableaddress[16];

bitn   regF[8];
bitn   dir;
bitn   dataout[8];
bitn   waitpc;
bitn   ancode0p;   { LD r,n }
bitn   ancode1p;   { LD ra,rb }
bitn   ancode2p;   { LD r,(HL) }
bitn   ancode3p;   { LD (HL),r }
bitn   ancode4p;   { LD IX[IY],n'n }
bitn   ancode5p;   { LD (IX[IY]+d),r }
bitn   ancode6p;   { LD r,(IX[IY]+d) }
bitn   ancode7p;   { LD (HL),n }
bitn   ancode8p;   { LD (IX[IY]+d),n }
bitn   ancode9p;   { LD A,(BC) }
bitn   ancode10p;  { LD (BC),A }
bitn   ancode11p;  { LD A,(DE) }
bitn   ancode12p;  { LD (DE),A }
bitn   ancode13p;  { LD A,(n'n) }
bitn   ancode14p;  { LD (n'n),A }
```

```
bitn   ancode15p;  { LD A,I }
bitn   ancode16p;  { LD I,A }
bitn   ancode17p;  { LD A,R }
bitn   ancode18p;  { LD R,A }
bitn   ancode19p;  { LD dd,n'n }
bitn   ancode20p;  { LD HL,(n'n) }
bitn   ancode21p;  { LD (n'n),HL }
bitn   ancode22p;  { LD dd,(n'n) }
bitn   ancode23p;  { LD (n'n),dd }
bitn   ancode24p;  { LD IX,(n'n) }
bitn   ancode25p;  { LD (n'n),IX }
bitn   ancode26p;  { LD IY,(n'n) }
bitn   ancode27p;  { LD (n'n),IY }
bitn   ancode28p;  { LD SP,HL }
bitn   ancode29p;  { LD SP,IX }
bitn   ancode30p;  { LD SP,IY }
bitn   ancode31p;  { PUSH qq }
bitn   ancode32p;  { POP qq }
bitn   ancode33p;  { PUSH IX }
bitn   ancode34p;  { PUSH IY }
bitn   ancode35p;  { POP IX }
bitn   ancode36p;  { POP IY }
bitn   ancode37p;  { EX DE,HL }
bitn   ancode38p;  { EX AF,AF' }
bitn   ancode39p;  { EXX }
bitn   ancode40p;  { EX (SP),HL }
bitn   ancode41p;  { EX (SP),IX }
bitn   ancode42p;  { EX (SP),IY }
bitn   ancode43p;  { LDI }
bitn   ancode44p;  { LDD }
bitn   ancode45p;  { LDIR }
bitn   ancode46p;  { LDDR }
bitn   ancode47p;  { CPI }
bitn   ancode48p;  { CPD }
bitn   ancode49p;  { CPIR }
bitn   ancode50p;  { CPDR }
bitn   ancode51p;  { ADD A,r }
bitn   ancode52p;  { ADD A,n }
bitn   ancode53p;  { ADD A,(HL) }
bitn   ancode54p;  { ADD A,(IX+d) }
bitn   ancode55p;  { ADD A,(IY+d) }
bitn   ancode56p;  { ADC A,r }
bitn   ancode57p;  { ADC A,n }
bitn   ancode58p;  { ADC A,(HL) }
bitn   ancode59p;  { ADC A,(IX+d) }
bitn   ancode60p;  { ADC A,(IY+d) }
bitn   ancode61p;  { SUB r }
bitn   ancode62p;  { SUB n }
bitn   ancode63p;  { SUB (HL) }
bitn   ancode64p;  { SUB (IX+d) }
bitn   ancode65p;  { SUB (IY+d) }
bitn   ancode66p;  { SBC A,r }
bitn   ancode67p;  { SBC A,n }
bitn   ancode68p;  { SBC A,(HL) }
bitn   ancode69p;  { SBC A,(IX+d) }
bitn   ancode70p;  { SBC A,(IY+d) }
bitn   ancode71p;  { AND r }
bitn   ancode72p;  { AND n }
bitn   ancode73p;  { AND (HL) }
bitn   ancode74p;  { AND (IX+d) }
```

```
bitn   ancode75p;  { AND (IY+d) }
bitn   ancode76p;  { OR r }
bitn   ancode77p;  { OR n }
bitn   ancode78p;  { OR (HL) }
bitn   ancode79p;  { OR (IX+d) }
bitn   ancode80p;  { OR (IY+d) }
bitn   ancode81p;  { XOR r }
bitn   ancode82p;  { XOR n }
bitn   ancode83p;  { XOR (HL) }
bitn   ancode84p;  { XOR (IX+d) }
bitn   ancode85p;  { XOR (IY+d) }
bitn   ancode86p;  { CP r }
bitn   ancode87p;  { CP n }
bitn   ancode88p;  { CP (HL) }
bitn   ancode89p;  { CP (IX+d) }
bitn   ancode90p;  { CP (IY+d) }
bitn   ancode91p;  { INC r }
bitn   ancode92p;  { INC (HL) }
bitn   ancode93p;  { INC (IX+d) }
bitn   ancode94p;  { INC (IY+d) }
bitn   ancode95p;  { DEC r }
bitn   ancode96p;  { DEC (HL) }
bitn   ancode97p;  { DEC (IX+d) }
bitn   ancode98p;  { DEC (IY+d) }
bitn   ancode99p;  { DAA }
bitn   ancode100p; { CPL }
bitn   ancode101p; { NEG }
bitn   ancode102p; { CCF }
bitn   ancode103p; { SCF }
bitn   ancode104p; { HALT }
bitn   ancode105p; { EI }
bitn   ancode106p; { DI }
bitn   ancode107p; { IM 0 }
bitn   ancode108p; { IM 1 }
bitn   ancode109p; { IM 2 }
bitn   ancode110p; { ADD HL,ss }
bitn   ancode111p; { ADC HL,ss }
bitn   ancode112p; { SBC HL,ss }
bitn   ancode113p; { ADD IX,pp }
bitn   ancode114p; { ADD IY,rr }
bitn   ancode115p; { INC ss }
bitn   ancode116p; { INC IX }
bitn   ancode117p; { INC IY }
bitn   ancode118p; { DEC ss }
bitn   ancode119p; { DEC IX }
bitn   ancode120p; { DEC IY }
bitn   ancode121p; { RLCA }
bitn   ancode122p; { RLA }
bitn   ancode123p; { RRCA }
bitn   ancode124p; { RRA }
bitn   ancode125p; { RLC r }
bitn   ancode126p; { RLC (HL) }
bitn   ancode127p; { RLC (IX+d) }
bitn   ancode128p; { RLC (IY+d) }
bitn   ancode129p; { RL r }
bitn   ancode130p; { RL (HL) }
bitn   ancode131p; { RL (IX+d) }
bitn   ancode132p; { RL (IY+d) }
bitn   ancode133p; { RRC r }
bitn   ancode134p; { RRC (HL) }
```

```
bitn   ancode135p; { RRC (IX+d) }
bitn   ancode136p; { RRC (IY+d) }
bitn   ancode137p; { RR r }
bitn   ancode138p; { RR (HL) }
bitn   ancode139p; { RR (IX+d) }
bitn   ancode140p; { RR (IY+d) }
bitn   ancode141p; { SLA r }
bitn   ancode142p; { SLA (HL) }
bitn   ancode143p; { SLA (IX+d) }
bitn   ancode144p; { SLA (IY+d) }
bitn   ancode145p; { SRL r }
bitn   ancode146p; { SRL (HL) }
bitn   ancode147p; { SRL (IX+d) }
bitn   ancode148p; { SRL (IY+d) }
bitn   ancode149p; { SRA r }
bitn   ancode150p; { SRA (HL) }
bitn   ancode151p; { SRA (IX+d) }
bitn   ancode152p; { SRA (IY+d) }
bitn   ancode153p; { RLD }
bitn   ancode154p; { RRD }
bitn   ancode155p; { BIT b,r }
bitn   ancode156p; { BIT b,(HL) }
bitn   ancode157p; { BIT b,(IX+d) }
bitn   ancode158p; { BIT b,(IY+d) }
bitn   ancode159p; { SET b,r }
bitn   ancode160p; { SET b,(HL) }
bitn   ancode161p; { SET b,(IX+d) }
bitn   ancode162p; { SET b,(IY+d) }
bitn   ancode163p; { RES b,r }
bitn   ancode164p; { RES b,(HL) }
bitn   ancode165p; { RES b,(IX+d) }
bitn   ancode166p; { RES b,(IY+d) }
bitn   ancode167p; { JP  n'n }
bitn   ancode168p; { JP  cc,n'n }
bitn   ancode169p; { JR  e }
bitn   ancode170p; { JR  C,e }
bitn   ancode171p; { JR  NC,e }
bitn   ancode172p; { JR  Z,e }
bitn   ancode173p; { JR  NZ,e }
bitn   ancode174p; { JP  (HL) }
bitn   ancode175p; { JP  (IX) }
bitn   ancode176p; { JP  (IY) }
bitn   ancode177p; { DJNZ e }
bitn   ancode178p; { CALL n'n }
bitn   ancode179p; { CALL cc,n'n }
bitn   ancode180p; { RET }
bitn   ancode181p; { RET cc }
bitn   ancode182p; { RETI }
bitn   ancode183p; { RETN }
bitn   ancode184p; { RST p }
bitn   ancode185p; { IN A,(n) }
bitn   ancode186p; { OUT (n),A }
bitn   ancode187p; { IN r,(C) }
bitn   ancode188p; { OUT (C),r }
bitn   ancode189p; { INI }
bitn   ancode190p; { IND }
bitn   ancode191p; { INIR }
bitn   ancode192p; { INDR }
bitn   ancode193p; { OUTI }
bitn   ancode194p; { OUTD }
```

```
bitn   ancode195p; { OTIR }
bitn   ancode196p; { OTDR }
bitn   z80code0p;  { IX 使用命令 }
bitn   z80code1p;  { IY 使用命令 }
bitn   z80code2p;  { I,R 使用命令 }
bitn   z80code3p;  { ビット操作命令 }
bitn   memoffset[16];
bitn   mempoint[16];
bitn   memreg[16];
bitn   mempoint1p[16];
bitn   mempoint1m[16];
bitn   alu16out[17];
bitn   alu16ina[17];
bitn   alu16inb[17];
bitn   alu8out[9];
bitn   alu8ina[9];
bitn   alu8inb[9];
bitn   alu4out[5];
bitn   alu4ina[5];
bitn   alu4inb[5];
bitn   carry[9];
bitn   wcarry[17];
bitn   alu4p[5];
bitn   alu4pc[5];
bitn   alu8p[9];
bitn   alu8pc[9];
bitn   alu16p[17];
bitn   alu16pc[17];
bitn   alu16m[17];
bitn   alu16mc[17];
bitn   alu4m[5];
bitn   alu4mc[5];
bitn   alu8m[9];
bitn   alu8mc[9];
bitn   alu8rss[9];
bitn   chgflgPV1p;
bitn   chgflgPV2p;
bitn   chgflgPV3p;
bitn   chgflgPV4p;
bitn   chgflgPV5p;
bitn   chgflgPV6p;
bitn   chgflgPV7p;
bitn   chgflgPV8p;
bitn   chgflgPV9p;
bitn   chgflgPV10p;
bitn   chgflgPV11p;
bitn   chgflgPV12p;
bitn   chgflgZ1p;
bitn   chgflgZ2p;
bitn   chgalu4in1p;
bitn   chgalu4in2p;
bitn   chgalu8ina1p;
bitn   chgalu8ina2p;
bitn   chgalu8ina3p;
bitn   chgalu8ina4p;
bitn   chgalu8ina5p;
bitn   chgregA1p;
bitn   chgregA2p;
bitn   chgregA3p;
bitn   chgregA4p;
bitn   chgregA5p;
bitn   chgregA6p;
bitn   chgregA7p;
bitn   chgregA9p;
bitn   chgregA10p;
```

```
bitn   chgregA11p;
bitn   chgregA12p;
bitn   chgregA13p;
bitn   chgregA14p;
bitn   chgregA15p;
bitn   chgregB1p;
bitn   chgregB2p;
bitn   chgregB3p;
bitn   chgregB4p;
bitn   chgregB5p;
bitn   chgregB6p;
bitn   chgregB7p;
bitn   chgregB9p;
bitn   chgregB10p;
bitn   chgregB12p;
bitn   chgregB13p;
bitn   chgregC1p;
bitn   chgregC2p;
bitn   chgregC3p;
bitn   chgregC4p;
bitn   chgregC5p;
bitn   chgregC6p;
bitn   chgregC8p;
bitn   chgregC9p;
bitn   chgregC10p;
bitn   chgregC11p;
bitn   chgregC12p;
bitn   chgregD1p;
bitn   chgregD2p;
bitn   chgregD3p;
bitn   chgregD4p;
bitn   chgregD5p;
bitn   chgregD6p;
bitn   chgregD7p;
bitn   chgregD9p;
bitn   chgregD10p;
bitn   chgregD11p;
bitn   chgregD12p;
bitn   chgregD13p;
bitn   chgregE1p;
bitn   chgregE2p;
bitn   chgregE3p;
bitn   chgregE4p;
bitn   chgregE5p;
bitn   chgregE6p;
bitn   chgregE7p;
bitn   chgregE9p;
bitn   chgregE10p;
bitn   chgregE11p;
bitn   chgregE12p;
bitn   chgregE13p;
bitn   chgregH1p;
bitn   chgregH2p;
bitn   chgregH3p;
bitn   chgregH4p;
bitn   chgregH5p;
bitn   chgregH6p;
bitn   chgregH7p;
bitn   chgregH8p;
bitn   chgregH10p;
bitn   chgregH11p;
bitn   chgregH12p;
bitn   chgregH13p;
bitn   chgregH14p;
bitn   chgregH15p;
bitn   chgregH16p;
```

```
bitn   chgregL1p;
bitn   chgregL2p;
bitn   chgregL3p;
bitn   chgregL4p;
bitn   chgregL5p;
bitn   chgregL6p;
bitn   chgregL7p;
bitn   chgregL8p;
bitn   chgregL10p;
bitn   chgregL11p;
bitn   chgregL12p;
bitn   chgregL13p;
bitn   chgregL14p;
bitn   chgregL15p;
bitn   chgregL16p;
bitn   chgADDRESS1p;
bitn   chgADDRESS2p;
bitn   chgADDRESS3p;
bitn   chgADDRESS4p;
bitn   chgADDRESS5p;
bitn   chgADDRESS6p;
bitn   chgmempoint1p;
bitn   chgmempoint2p;
bitn   chgmempoint3p;
bitn   chgmemoffset1p;
bitn   chgmemoffset2p;
bitn   chgmemoffset4p;
bitn   chgflgC2p;
bitn   chgflgC3p;
bitn   chgflgC4p;
bitn   chgflgC5p;
bitn   chgflgC6p;
bitn   chgflgC7p;
bitn   chgflgC8p;
bitn   chgflgC9p;
bitn   chgflgC11p;
bitn   chgflgC12p;
bitn   chgflgC13p;
bitn   parityl[3];
bitn   parityh[3];
bitn   parity;
bitn   alu8and[9];
bitn   alu8or[9];
bitn   alu8xor[9];
bitn   daalin[5];
bitn   daahin[5];
bitn   daalA[5];
bitn   daahA[5];
bitn   ab10lin;
bitn   ab10hin;
bitn   daacarry;
bitn   chgflgZ3p;
bitn   chgalu4in3p;
bitn   chgalu4in4p;
bitn   chgregH4p;
bitn   chgregL4p;
bitn   chgflgPV5p;
bitn   chgflgZ4p;
bitn   alu16ina1p;
bitn   alu16inb1p;
bitn   chgflgPV6p;
bitn   alu16ina2p;
bitn   alu16inb2p;
bitn   alu16ina3p;
bitn   alu16inb3p;
bitn   chgregB3p;
```

```
bitn   chgregC3p;
bitn   chgregD4p;
bitn   chgregE4p;
bitn   chgregH5p;
bitn   chgregL5p;
bitn   alu16ina4p;
bitn   chgregSP1p;
bitn   chgregSP2p;
bitn   chgregSP3p;
bitn   chgregSP4p;
bitn   chgregSP5p;
bitn   chgregSP6p;
bitn   chgregSP7p;
bitn   chgregSP8p;
bitn   chgregSP9p;
bitn   chgregSP10p;
bitn   chgregSP11p;
bitn   alu16inb4p;
bitn   alu16inb5p;
bitn   alu16ina5p;
bitn   alu16inb6p;
bitn   alu8ls[9];
bitn   alu8rs[9];
bitn   alu8lsc[9];
bitn   alu8rsc[9];
bitn   alu8lsz[9];
bitn   alu8rsz[9];
bitn   chgregA4p;
bitn   chgregA5p;
bitn   chgregB4p;
bitn   chgregC4p;
bitn   chgregD5p;
bitn   chgregE5p;
bitn   chgregH6p;
bitn   chgregL6p;
bitn   chgalu8inb1p;
bitn   chgalu8inb2p;
bitn   chgalu8inb3p;
bitn   chgalu8inb4p;
bitn   chgalu8inb5p;
bitn   chgalu8inb6p;
bitn   chgalu8ina6p;
bitn   chgalu8inb7p;
bitn   chgregPC1p;
bitn   chgregPC2p;
bitn   chgregPC3p;
bitn   chgregPC7p;
bitn   chgregPC8p;
bitn   chgregPC10p;
bitn   chgregPC11p;
bitn   chgregPC12p;
bitn   chgregPC13p;
bitn   chgregPC14p;
bitn   chgregPC15p;
bitn   chgregPC16p;
bitn   chgregPC17p;
bitn   addpc[16];
bitn   regB1m[8];
bitn   chgregB5p;
bitn   regPC2p[16];
bitn   regPC1p[16];
bitn   ccen;
bitn   mio;
bitn   chgADDRESS7p;
bitn   chgregA6p;
bitn   chgADDRESS8p;
```

```
bitn   chgalu8ina7p;
bitn   chgalu8ina8p;
bitn   chgalu8ina9p;
bitn   chganct1p;
bitn   chganct2p;
bitn   chganct3p;
bitn   chgADDRESS9p;
bitn   intaddress[16];
bitn   intack;
bitn   chgADDRESS10p;
bitn   chgintackdata1p;
bitn   chgdataout1p;
bitn   chgdataout2p;
bitn   chgdataout3p;
bitn   chgdataout4p;
bitn   chgdataout5p;
bitn   chgdataout6p;
bitn   chgdataout7p;
bitn   chgdataout8p;
bitn   chgdataout9p;
bitn   chgdataout10p;
bitn   chgdataout11p;
bitn   chgdataout12p;
bitn   chgdataout13p;
bitn   chgdataout14p;
bitn   chgdataout15p;
bitn   chgreg0T1p;
bitn   chgreg0T2p;
bitn   chgreg1T1p;
bitn   chgreg1T2p;
bitn   chgalu8out1p;
bitn   chgalu8out2p;
bitn   chgalu8out3p;
bitn   chgalu8out4p;
bitn   chgalu8out5p;
bitn   chgalu8out6p;
bitn   chgalu8out7p;
bitn   chgalu8out8p;
bitn   chgalu8out9p;
bitn   chgalu8out10p;
bitn   chgalu8out11p;
bitn   chgalu8out12p;
bitn   chgalu8out13p;
bitn   chgalu8out14p;
bitn   chgalu8out15p;
bitn   chgalu8out16p;
bitn   chgalu16out1p;
bitn   chgalu16out2p;
bitn   chgalu4ina1p;
bitn   chgalu4ina2p;
bitn   chgalu4ina3p;
bitn   chgalu4ina4p;
bitn   chgalu4out1p;
bitn   chgalu4out2p;
bitn   chgalu4out3p;
bitn   chgalu4out4p;
bitn   instp[200];
bitn   anct0p;
bitn   anct1p;
bitn   anct2p;
bitn   anct3p;
bitn   anct4p;
bitn   anct5p;
bitn   anct10p;
bitn   anct11p;
bitn   chgflgEI0p;
```

```
bitn   chgflgEI1p;
bitn   chgflgEI2p;
bitn   chgflgN0p;
bitn   chgflgN1p;
bitn   chgflgN2p;
bitn   chgflgN3p;
bitn   chgflgH1p;
bitn   chgflgH2p;
bitn   chgflgH3p;
bitn   chgflgH4p;
bitn   chgflgH5p;
bitn   chgflgH6p;
bitn   chgflgZ5p;
bitn   chgflgZ6p;
bitn   chgflgZ7p;
bitn   chgflgZ8p;
bitn   chgflgZ9p;
bitn   chgflgZ10p;
bitn   chgflgZ11p;
bitn   chgflgS1p;
bitn   chgflgS2p;
bitn   chgflgS3p;
bitn   chgflgS4p;
bitn   chgflgS5p;
bitn   chgflgS6p;
bitn   chgflgS7p;
bitn   chgregIX1p;
bitn   chgregIX2p;
bitn   chgregIX3p;
bitn   chgregIX4p;
bitn   chgregIX5p;
bitn   chgregIX6p;
bitn   chgregIX7p;
bitn   chgregIY1p;
bitn   chgregIY2p;
bitn   chgregIY3p;
bitn   chgregIY4p;
bitn   chgregIY5p;
bitn   chgregIY6p;
bitn   chgregIY7p;

{ ------------------------------------------------------------------------ }
{    検査端子                                                            }
{ ------------------------------------------------------------------------ }

output T0P[8]; T0P=regA;
output T1P[8]; T1P=regB;
output T2P[8]; T2P=regC;
output T3P[8]; T3P=regD;
output T4P[8]; T4P=regE;
output T5P[8]; T5P=regH;
output T6P[8]; T6P=regL;
output T7P[4]; T7P=anct;
output T8P[8]; T8P=inst;
output T9P[8]; T9P=code0d;
output T10P[8]; T10P=code1d;
output T11P[8]; T11P=code2d;
output T12P[8]; T12P=code3d;
output T13P[16]; T13P=regPC;
output T14P[16]; T14P=regIX;
output T15P[16]; T15P=regIY;
output T16P[4];  T16P=expinst;
output T17P[8];  T17P=regI;
output T18P;     T18P=flgS;
output T19P;     T19P=flgZ;
output T20P;     T20P=flgH;
```

```
output T21P;     T21P=flgPV;
output T22P;     T22P=flgN;
output T23P;     T23P=flgC;
output T24P;     T24P=flgEI;
output T25P[8];  T25P=regR;
output T26P[16]; T26P=regSP;
output T27P[8];  T27P=regRA;
output T28P[8];  T28P=regRB;
output T29P[8];  T29P=regRC;
output T30P[8];  T30P=regRD;
output T31P[8];  T31P=regRE;
output T32P[8];  T32P=regRH;
output T33P[8];  T33P=regRL;
output T34P[8];  T34P=reg0T;
output T35P[8];  T35P=reg1T;
output T36P[9];  T36P=alu8out;
output T37P[9];  T37P=alu8m;
output T38P[4];  T38P=codesize;
output T39P[5];  T39P=alu4out;
output T40P[2];  T40P=im;
output T41P[17]; T41P=alu16out;
output T42P[17]; T42P=alu16ina;
output T43P[17]; T43P=alu16inb;
output T44P;     T44P=ancode143p;
output T45P;     T45P=ccenm;
output T46P[4];  T46P=nmiset;
output T47P[4];  T47P=intset;
output T48P;     T48P=ancode159p;
output T49P;     T49P=waitpc;
output T50P;     T50P=instp.132;
output T51P;     T51P=instp.23;
output T52P;     T52P=instp.179;
output T53P;     T53P=instp.180;
output T54P;     T54P=anct0p;
output T55P;     T55P=anct1p;
output T56P;     T56P=anct2p;
output T57P;     T57P=anct3p;
output T58P;     T58P=anct4p;
output T59P;     T59P=anct5p;
output T60P;     T60P=ancode175p;
output T61P[9];  T61P=alu8ina;
output T62P[9];  T62P=alu8inb;
output T63P;     T63P=chgdataout8p;
output T64P;     T64P=chgdataout9p;
output T65P[8];  T65P=regF;
output T66P;     T66P=ancode125p;              { RLC r }
output T67P;     T67P=ancode129p;              { RL r }
output T68P;     T68P=ancode133p;              { RRC r }
output T69P;     T69P=ancode137p;              { RR r }
output T70P;     T70P=ancode141p;              { SLA r }
output T71P;     T71P=ancode145p;              { SRL r }
output T72P;     T72P=ancode149p;              { SRA r }
output T73P;     T73P=ancode155p;              { BIT b,r }
output T74P;     T74P=ancode156p;              { BIT b,(HL) }
output T75P;     T75P=ancode159p;              { SET b,r }
output T76P;     T76P=ancode163p;              { RES b,r }
output T77P;     T77P=flgRS;
output T78P;     T78P=flgRZ;
output T79P;     T79P=flgRH;
output T80P;     T80P=flgRPV;
output T81P;     T81P=flgRN;
output T82P;     T82P=flgRC;
output T83P;     T83P=flgREI;
```

```
{ ------------------------------------------------------------------------ }
{    出力代入                                                            }
{ ------------------------------------------------------------------------ }
   Q.0:15=ADDRESS.0:15;
   Q.16:23=DATAOUT.0:7;
   Q.24=DIR;
   Q.25=MIO;
   Q.26=INTA;

   DIR=dir;
   MIO=mio;
   INTA=intack;
   DATAOUT=dataout;

{ ------------------------------------------------------------------------ }
{    番地選択                                                            }
{ ------------------------------------------------------------------------ }
   if (anct1p)
      if ( instp.21   { LD HL,(n'n)  }
         | instp.22   { LD (n'n),HL }
         | instp.32   { PUSH qq }
         | instp.33   { POP qq }
         | instp.41   { EX (SP),HL }
         | instp.7    { LD r,(IX[IY]+d) }
         | instp.8    { LD (IX[IY]+d),r }
         | instp.9    { LD (IX[IY]+d),n }
         | instp.23   { LD dd,(n'n) }
         | instp.24   { LD (n'n),dd }
         | instp.25   { LD IX,(n'n) }
         | instp.26   { LD (n'n),IX }
         | instp.27   { LD IY,(n'n) }
         | instp.28   { LD (n'n),IY }
         | instp.34   { PUSH IX }
         | instp.35   { PUSH IY }
         | instp.36   { POP IX }
         | instp.37   { POP IY }
         | instp.42   { EX (SP),IX }
         | instp.43   { EX (SP),IY }
         | instp.55   { ADD A,(IX+d) }
         | instp.56   { ADD A,(IY+d) }
         | instp.60   { ADC A,(IX+d) }
         | instp.61   { ADC A,(IY+d) }
         | instp.65   { SUB (IX+d) }
         | instp.66   { SUB (IY+d) }
         | instp.70   { SBC A,(IX+d) }
         | instp.71   { SBC A,(IY+d) }
         | instp.75   { AND (IX+d) }
         | instp.80   { OR (IX+d) }
         | instp.81   { OR (IY+d) }
         | instp.85   { XOR (IX+d) }
         | instp.86   { XOR (IY+d) }
         | instp.90   { CP (IX+d) }
         | instp.91   { CP (IY+d) }
         | instp.94   { INC (IX+d) }
         | instp.95   { INC (IY+d) }
         | instp.98   { DEC (IX+d) }
         | instp.99   { DEC (IY+d) }
         | instp.128  { RLC (IX+d) }
         | instp.129  { RLC (IY+d) }
         | instp.132  { RL (IX+d) }
         | instp.133  { RL (IY+d) }
```

```
        | instp.136  { RRC (IX+d) }
        | instp.137  { RRC (IY+d) }
        | instp.140  { RR (IX+d) }
        | instp.141  { RR (IY+d) }
        | instp.144  { SLA (IX+d) }
        | instp.145  { SLA (IY+d) }
        | instp.148  { SRL (IX+d) }
        | instp.149  { SRL (IY+d) }
        | instp.152  { SRA (IX+d) }
        | instp.153  { SRA (IY+d) }
        | instp.158  { BIT b,(IX+d) }
        | instp.159  { BIT b,(IY+d) }
        | instp.162  { SET b,(IX+d) }
        | instp.163  { SET b,(IY+d) }
        | instp.166  { RES b,(IX+d) }
        | instp.167  { RES b,(IY+d) }
        | instp.198  { NMI }
        | instp.199  { INT }
       )
        chgADDRESS1p=1;
    endif
endif

if (anct2p)
    if ( instp.21    { LD HL,(n'n) }
       | instp.22    { LD (n'n),HL }
       | instp.32    { PUSH qq }
       | instp.33    { POP qq }
       | instp.41    { EX (SP),HL }
       | instp.23    { LD dd,(n'n) }
       | instp.24    { LD (n'n),dd }
       | instp.25    { LD IX,(n'n) }
       | instp.26    { LD (n'n),IX }
       | instp.27    { LD IY,(n'n) }
       | instp.28    { LD (n'n),IY }
       | instp.34    { PUSH IX }
       | instp.35    { PUSH IY }
       | instp.36    { POP IX }
       | instp.37    { POP IY }
       | instp.42    { EX (SP),IX }
       | instp.43    { EX (SP),IY }
       | instp.94    { INC (IX+d) }
       | instp.95    { INC (IY+d) }
       | instp.98    { DEC (IX+d) }
       | instp.99    { DEC (IY+d) }
       | instp.128   { RLC (IX+d) }
       | instp.129   { RLC (IY+d) }
       | instp.132   { RL (IX+d) }
       | instp.133   { RL (IY+d) }
       | instp.136   { RRC (IX+d) }
       | instp.137   { RRC (IY+d) }
       | instp.140   { RR (IX+d) }
       | instp.141   { RR (IY+d) }
       | instp.144   { SLA (IX+d) }
       | instp.145   { SLA (IY+d) }
       | instp.148   { SRL (IX+d) }
       | instp.149   { SRL (IY+d) }
       | instp.152   { SRA (IX+d) }
       | instp.153   { SRA (IY+d) }
       | instp.162   { SET b,(IX+d) }
       | instp.163   { SET b,(IY+d) }
```

```
      | instp.166   { RES b,(IX+d) }
      | instp.167   { RES b,(IY+d) }
      | instp.198   { NMI }
      | instp.199
      )
       chgADDRESS1p=1;
   endif
endif

if (anct3p)
   if ( instp.41   { EX (SP),HL }
      | instp.42   { EX (SP),IX }
      | instp.43   { EX (SP),IY }
      )
       chgADDRESS1p=1;
   endif
endif

if (anct4p)
   if ( instp.41   { EX (SP),HL }
      | instp.42   { EX (SP),IX }
      | instp.43   { EX (SP),IY }
      )
       chgADDRESS1p=1;
   endif
endif

if (anct1p)
   if ( instp.3   { LD r,(HL) }
      | instp.4   { LD (HL),r }
      | instp.6   { LD (HL),n }
      | instp.48  { CPI }
      | instp.49  { CPD }
      | instp.50  { CPIR }
      | instp.51  { CPDR }
      | instp.54  { ADD A,(HL) }
      | instp.59  { ADC A,(HL) }
      | instp.64  { SUB (HL) }
      | instp.69  { SBC A,(HL) }
      | instp.74  { AND (HL) }
      | instp.79  { OR (HL) }
      | instp.84  { XOR (HL) }
      | instp.89  { CP (HL) }
      | instp.93  { INC (HL) }
      | instp.97  { DEC (HL) }
      | instp.127 { RLC (HL) }
      | instp.131 { RL (HL) }
      | instp.135 { RRC (HL) }
      | instp.139 { RR (HL) }
      | instp.143 { SLA (HL) }
      | instp.147 { SRL (HL) }
      | instp.151 { SRA (HL) }
      | instp.154 { RLD }
      | instp.155 { RRD }
      | instp.157 { BIT b,(HL) }
      | instp.161 { SET b,(HL) }
      | instp.165 { RES b,(HL) }
      | instp.190 { INI }
      | instp.191 { IND }
      | instp.192 { INIR }
      | instp.193 { INDR }
      )
```

```
      chgADDRESS2p=1;
   endif
endif

if (anct2p)
   if ( instp.44   { LDI }
      | instp.45   { LDD }
      | instp.46   { LDIR }
      | instp.47   { LDDR }
      | instp.69   { SBC A,(HL) }
      | instp.93   { INC (HL) }
      | instp.97   { DEC (HL) }
      | instp.127  { RLC (HL) }
      | instp.131  { RL (HL) }
      | instp.135  { RRC (HL) }
      | instp.139  { RR (HL) }
      | instp.143  { SLA (HL) }
      | instp.147  { SRL (HL) }
      | instp.151  { SRA (HL) }
      | instp.154  { RLD }
      | instp.155  { RRD }
      | instp.161  { SET b,(HL) }
      | instp.165  { RES b,(HL) }
      | instp.194   { OUTI }
      | instp.195   { OUTD }
      | instp.196   { OTIR }
      | instp.197   { OTDR }
      )
      chgADDRESS2p=1;
   endif
endif

if (anct1p)
   if ( instp.12   { LD A,(DE) }
      | instp.13   { LD (DE),A }
      | instp.44   { LDI }
      | instp.45   { LDD }
      | instp.46   { LDIR }
      | instp.47   { LDDR }
      )
      chgADDRESS3p=1;
   endif
endif

if (anct1p)
   if ( instp.10   { LD A,(BC) }
      | instp.11   { LD (BC),A }
      )
      chgADDRESS4p=1;
   endif
endif

if (anct1p)
   if ( instp.14   { LD A,(n'n) }
      | instp.15   { LD (n'n),A }
      )
      chgADDRESS5p=1;
   endif
endif

if (anct1p)
   if ( instp.179   { CALL n'n }
      | instp.180   { CALL cc,n'n }
```

```
      | instp.181   { RET }
      | instp.182   { RET cc }
      | instp.183   { RETI }
      | instp.184   { RETN }
      | instp.185   { RST }
      )
      chgADDRESS6p=1;
   endif
endif

if (anct2p)
   if ( instp.179   { CALL n'n }
      | instp.180   { CALL cc,n'n }
      | instp.181   { RET }
      | instp.182   { RET cc }
      | instp.183   { RETI }
      | instp.184   { RETN }
      | instp.185   { RST }
      )
      chgADDRESS6p=1;
   endif
endif

if (anct1p)
   if ( instp.186   { IN A,(n) }
      | instp.187   { OUT (n),A }
      )
       chgADDRESS7p=1;
   endif
endif

if (anct1p)
   if ( instp.188   { IN r,(C) }
      | instp.189   { OUT (C),r }
      | instp.194   { OUTI }
      | instp.195   { OUTD }
      | instp.196   { OTIR }
      | instp.197   { OTDR }
      )
      chgADDRESS8p=1;
   endif
endif

if (anct2p)
   if ( instp.190   { INI }
      | instp.191   { IND }
      | instp.192   { INIR }
      | instp.193   { INDR }
      )
      chgADDRESS8p=1;
   endif
endif

if (anct5p)
   if (instp.199) chgADDRESS9p=1; endif  { INT }
endif

if (anct4p)
   if (instp.199)  { INT }
      switch(im)
         case 0: chgADDRESS9p=1;
         case 2: chgADDRESS10p=1;
      endswitch
   endif
```

```
endif

if (anct3p)
   if (instp.199)  { INT }
      switch(im)
         case 0: chgADDRESS9p=1;
         case 2: chgADDRESS10p=1;
      endswitch
   endif
endif

if ( chgADDRESS1p
   | chgADDRESS2p
   | chgADDRESS3p
   | chgADDRESS4p
   | chgADDRESS5p
   | chgADDRESS6p
   | chgADDRESS7p
   | chgADDRESS8p
   | chgADDRESS9p
   | chgADDRESS10p
   )
else
   ADDRESS=regPC;
endif

if (chgADDRESS1p) ADDRESS=mempoint; endif

if (chgADDRESS2p)
   ADDRESS.0:7=regL;
   ADDRESS.8:15=regH;
endif

if (chgADDRESS3p)
   ADDRESS.0:7=regE;
   ADDRESS.8:15=regD;
endif

if (chgADDRESS4p)
   ADDRESS.0:7=regC;
   ADDRESS.8:15=regB;
endif

if (chgADDRESS5p)
   ADDRESS.0:7=code2d;
   ADDRESS.8:15=code1d;
endif

if (chgADDRESS6p) ADDRESS=regSP; endif

if (chgADDRESS7p)
   ADDRESS.0:7=code1d;
   ADDRESS.8:15=regA;
endif

if (chgADDRESS8p)
   ADDRESS.0:7=regC;
   ADDRESS.8:15=regB;
endif

if (chgADDRESS9p)
   ADDRESS.0:3=anct.0:3;
   ADDRESS.4:15=0;
endif

if (chgADDRESS10p)
```

```
      ADDRESS=inttableaddress;
   endif

{ ------------------------------------------------------------------- }
{    割り込みテーブル番地                                  }
{ ------------------------------------------------------------------- }
   if (RESET)
      inttableaddress=0;
   else
      if (!WAIT)
         switch(instp.199,anct5p,anct4p)  { INT }
            case 1,1,0:
               inttableaddress.0:7=DATAIN;
               inttableaddress.8:15=regI;
            case 1,0,1:
               inttableaddress=inttableaddress+1;
            default:
               inttableaddress=inttableaddress;
         endswitch
      else
         inttableaddress=inttableaddress;
      endif
   endif

{ ------------------------------------------------------------------- }
{    NMI 記憶                                              }
{ ------------------------------------------------------------------- }
   if (RESET)
      nmiset=0;
   else
      if (!WAIT)
         switch(nmiset)
            case 0:
               if (NMI) nmiset=1; endif
            case 1:
               if (instp.198&anct1p)
                  nmiset=2;              NMI
               else
                  nmiset=nmiset;
               endif
            case 2:
               if (instp.184&anct1p)
                  nmiset=4;              RETN
               else
                  nmiset=nmiset;
               endif
            case 4:
               if (NMI) nmiset=nmiset; else nmiset=0; endif
         endswitch
      else
         nmiset=nmiset;
      endif
   endif

{ ------------------------------------------------------------------- }
{    INT 番地                                              }
{ ------------------------------------------------------------------- }
   switch(im)
      case 0:
         switch(intack0data)
            case 0xcd:
               intaddress.0:7=intack1data;
               intaddress.8:15=intack2data;
            default:
               switch(intack0data.6:7)
```

```
                  case 3:
                     switch(intack0data.0:2)
                        case 7:
                           switch(intack0data.3:5)
                              case 0: intaddress=0x00;
                              case 1: intaddress=0x08;
                              case 2: intaddress=0x10;
                              case 3: intaddress=0x18;
                              case 4: intaddress=0x20;
                              case 5: intaddress=0x28;
                              case 6: intaddress=0x30;
                              case 7: intaddress=0x38;
                           endswitch
                     endswitch
               endswitch
         endswitch
      case 2:
         intaddress.0:7=intack1data;
         intaddress.8:15=intack2data;
   endswitch

{ ------------------------------------------------------------------------ }
{    INT 応答コード                                          }
{ ------------------------------------------------------------------------ }
   if (RESET)
      intack0data=0;
      intack1data=0;
      intack2data=0;
   else
      if (!WAIT)
         switch(im)
            case 0: chgintackdata1p=1;
            case 2: chgintackdata1p=1;
         endswitch
      else
         intack0data=intack0data;
         intack1data=intack1data;
         intack2data=intack2data;
      endif
   endif

   if (chgintackdata1p)
      if (instp.199)
         switch(anct5p,anct4p,anct3p)
           case 1,0,0:
              intack0data=DATAIN;
              intack1data=intack1data;
              intack2data=intack2data;
           case 0,1,0:
              intack0data=intack0data;
              intack1data=DATAIN;
              intack2data=intack2data;
           case 0,0,1:
              intack0data=intack0data;
              intack1data=intack1data;
              intack2data=DATAIN;
           default:
              intack0data=intack0data;
              intack1data=intack1data;
              intack2data=intack2data;
         endswitch
      endif
   endif

{ ------------------------------------------------------------------------ }
{    INT 応答                                          }
```

```
{ ------------------------------------------------------------------------ }
   switch(im)
      case 0:
         switch(intset,anct)
            case 1,5: intack=1;   { INT CALL , RST }
            case 1,4: intack=1;   { n' }
            case 1,3: intack=1;   { n }
         endswitch
      case 2:
         switch(intset,anct)
            case 1,5: intack=1;   { INT 下位割り込みテーブル開始番地 }
         endswitch
   endswitch

{ ------------------------------------------------------------------------ }
{    INT 記憶                                                  }
{ ------------------------------------------------------------------------ }
   if (RESET)
      intset=0;
   else
      if (!WAIT)
            switch(intset)
               case 0:                          { 割り込みなし }
                  if (flgEI)
                     if (INT)
                        switch(nmiset)
                           case 0:
                              if (NMI)
                                 intset=6;   { 非マスク割り込みへ }
                              else
                                 intset=1;   { マスク可能割り込みへ }
                              endif
                           default: intset=6;
                        endswitch
                     endif
                  endif
               case 1:                          { マスク可能割り込み受付中 }
                  if (instp.199&anct1p)         { INT }
                     intset=4;
                  else
                     intset=intset;
                  endif
               case 4:                       { マスク可能割り込み発効 }
                  if (INT) intset=intset; else intset=0; endif
               case 6:                       { 非マスク割り込み }
                  switch(nmiset)
                     case 0:  intset=1;
                     case 4:  intset=1;
                     default: intset=intset;
                  endswitch
            endswitch
      else
         intset=intset;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    メモリ位置レジスタ                                    }
{ ------------------------------------------------------------------------ }
   switch(expinst)
      case 1: memreg=regIX;
      case 2: memreg=regIY;
      case 5: memreg=regIX;
      case 6: memreg=regIY;
   endswitch
```

```
{ ------------------------------------------------------------------------ }
{    メモリ位置 +1                                                        }
{ ------------------------------------------------------------------------ }
   mempoint1p=memoffset+1;

{ ------------------------------------------------------------------------ }
{    メモリ位置-1                                                         }
{ ------------------------------------------------------------------------ }
   mempoint1m=memoffset-1;

{ ------------------------------------------------------------------------ }
{    メモリ位置                                                           }
{ ------------------------------------------------------------------------ }
   if (anct1p)
      if ( instp.33   { POP qq }
         | instp.36   { POP IX }
         | instp.37   { POP IY }
         | instp.41   { EX (SP),HL }
         | instp.42   { EX (SP),IX }
         | instp.43   { EX (SP),IY }
         )
         chgmempoint1p=1;
      endif
   endif

   if (anct2p)
      if ( instp.21   { LD HL,(n'n) }
         | instp.22   { LD (n'n),HL }
         | instp.23   { LD dd,(n'n) }
         | instp.24   { LD (n'n),dd }
         | instp.25   { LD IX,(n'n) }
         | instp.26   { LD (n'n),IX }
         | instp.27   { LD IY,(n'n) }
         | instp.28   { LD (n'n),IY }
         | instp.33   { POP qq }
         | instp.36   { POP IX }
         | instp.37   { POP IY }
         )
         chgmempoint1p=1;
      endif
   endif

   if (anct3p)
      if ( instp.41   { EX (SP),HL }
         | instp.42   { EX (SP),IX }
         | instp.43   { EX (SP),IY }
         )
         chgmempoint1p=1;
      endif
   endif

   if (anct1p)
      if ( instp.21   { LD HL,(n'n) }
         | instp.22   { LD (n'n),HL }
         | instp.23   { LD dd,(n'n) }
         | instp.24   { LD (n'n),dd }
         | instp.25   { LD IX,(n'n) }
         | instp.26   { LD (n'n),IX }
         | instp.27   { LD IY,(n'n) }
         | instp.28   { LD (n'n),IY }
         )
         chgmempoint2p=1;
```

```
   endif
endif

if (anct2p)
   if ( instp.41   { EX (SP),HL }
      | instp.42   { EX (SP),IX }
      | instp.43   { EX (SP),IY }
      )
      chgmempoint2p=1;
   endif
endif

if (anct4p)
   if ( instp.41   { EX (SP),HL }
      | instp.42   { EX (SP),IX }
      | instp.43   { EX (SP),IY }
      )
      chgmempoint2p=1;
   endif
endif

if (instp.181)
   if (anct1p) chgmempoint2p=1; endif { RET }
   if (anct2p) chgmempoint2p=1; endif
endif

if (instp.182)
   if (anct1p) chgmempoint2p=1; endif { RET cc }
   if (anct2p) chgmempoint2p=1; endif
endif

if (instp.183)
   if (anct1p) chgmempoint2p=1; endif { RETI }
   if (anct2p) chgmempoint2p=1; endif
endif

if (instp.184)
   if (anct1p) chgmempoint2p=1; endif { RETN }
   if (anct2p) chgmempoint2p=1; endif
endif

if (instp.32)
   if (anct1p) chgmempoint3p=1; endif { PUSH qq }
   if (anct2p) chgmempoint3p=1; endif
endif

if (instp.34)
   if (anct1p) chgmempoint3p=1; endif { PUSH IX }
   if (anct2p) chgmempoint3p=1; endif
endif

if (instp.35)
   if (anct1p) chgmempoint3p=1; endif { PUSH IY }
   if (anct2p) chgmempoint3p=1; endif
endif

if (instp.179)
   if (anct1p) chgmempoint3p=1; endif { CALL n'n }
   if (anct2p) chgmempoint3p=1; endif
endif

if (instp.180)
   if (anct1p) chgmempoint3p=1; endif { CALL cc,n'n }
   if (anct2p) chgmempoint3p=1; endif
```

```
   endif

   if (instp.185&anct2p) chgmempoint3p=1; endif { RST p }

   if (instp.198)
      if (anct1p) chgmempoint3p=1; endif { NMI }
      if (anct2p) chgmempoint3p=1; endif
   endif

   if (instp.199)
      if (anct1p) chgmempoint3p=1; endif { INT }
      if (anct2p) chgmempoint3p=1; endif
   endif

   if (chgmempoint1p|chgmempoint2p|chgmempoint3p)
   else
      mempoint=memreg+memoffset;
   endif

   if (chgmempoint1p) mempoint=memoffset; endif
   if (chgmempoint2p) mempoint=mempoint1p; endif
   if (chgmempoint3p) mempoint=mempoint1m; endif

{ ------------------------------------------------------------------------ }
{    メモリオフセット                                          }
{ ------------------------------------------------------------------------ }
   if ( instp.21   { LD HL,(n'n) }
      | instp.22   { LD (n'n),HL }
      | instp.23   { LD dd,(n'n) }
      | instp.24   { LD (n'n),dd }
      | instp.25   { LD IX,(n'n) }
      | instp.26   { LD (n'n),IX }
      | instp.27   { LD IY,(n'n) }
      | instp.28   { LD (n'n),IY }
      )
      chgmemoffset1p=1;
   endif

   if ( instp.32      { PUSH qq }
      | instp.33      { POP qq }
      | instp.34      { PUSH IX }
      | instp.35      { PUSH IY }
      | instp.36      { POP IX }
      | instp.37      { POP IY }
      | instp.41      { EX (SP),HL }
      | instp.42      { EX (SP),IX }
      | instp.43      { EX (SP),IY }
      | instp.179      { CALL n'n }
      | instp.180      { CALL cc,n'n }
      | instp.181      { RET }
      | instp.182      { RET cc }
      | instp.183      { RETI }
      | instp.184      { RETN }
      | instp.185      { RST p }
      | instp.198      { NMI }
      | instp.199      { INT }
      )
      chgmemoffset2p=1;
   endif

   if ( instp.9        { LD (IX[IY]+d),n }
      | instp.128      { RLC (IX+d) }
      | instp.129      { RLC (IY+d) }
```

```
      | instp.132      { RL (IX+d) }
      | instp.133      { RL (IY+d) }
      | instp.136      { RRC (IX+d) }
      | instp.137      { RRC (IY+d) }
      | instp.140      { RR (IX+d) }
      | instp.141      { RR (IY+d) }
      | instp.144      { SLA (IX+d) }
      | instp.145      { SLA (IY+d) }
      | instp.148      { SRL (IX+d) }
      | instp.149      { SRL (IY+d) }
      | instp.152      { SRA (IX+d) }
      | instp.153      { SRA (IY+d) }
      | instp.158      { BIT b,(IX+d) }
      | instp.159      { BIT b,(IY+d) }
      | instp.162      { SET b,(IX+d) }
      | instp.163      { SET b,(IY+d) }
      | instp.166      { RES b,(IX+d) }
      | instp.167      { RES b,(IY+d) }
      )
      chgmemoffset4p=1;
    endif

   if (chgmemoffset1p|chgmemoffset2p|chgmemoffset4p)
   else
      memoffset.0:7=code1d;

      if (code1d.7)
         memoffset.8:15=0xff;
      else
         memoffset.8:15=0;
      endif
   endif

   if (chgmemoffset1p)
      memoffset.0:7=code2d;
      memoffset.8:15=code1d;
   endif

   if (chgmemoffset2p) memoffset=regSP; endif

   if (chgmemoffset4p)          { (IX[IY]+d) }
      memoffset.0:7=code2d;

      if (code2d.7)
         memoffset.8:15=0xff;   { 負の符号拡張 }
      else
         memoffset.8:15=0;      { 正の符号拡張 }
      endif
   endif

{ ------------------------------------------------------------------- }
{    データバスの方向                                                 }
{ ------------------------------------------------------------------- }
   if (anct1p)
      if ( instp.4    { LD (HL),r }
         | instp.6    { LD (HL),n }
         | instp.11   { LD (BC),A }
         | instp.13   { LD (DE),A }
         | instp.15   { LD (n'n),A }
         | instp.22   { LD (n'n),HL }
         | instp.32   { PUSH qq }
         | instp.41   { EX (SP),HL }
         | instp.8    { LD (IX[IY]+d),r }
```

```
| instp.9    { LD (IX[IY]+d),n }
| instp.24   { LD (n'n),dd }
| instp.26   { LD (n'n),IX }
| instp.28   { LD (n'n),IY }
| instp.34   { PUSH IX }
| instp.35   { PUSH IY }
| instp.42   { EX (SP),IX }
| instp.43   { EX (SP),IY }
| instp.44   { LDI }
| instp.45   { LDD }
| instp.46   { LDIR }
| instp.47   { LDDR }
| instp.93   { INC (HL) }
| instp.94   { INC (IX+d) }
| instp.95   { INC (IY+d) }
| instp.97   { DEC (HL) }
| instp.98   { DEC (IX+d) }
| instp.99   { DEC (IY+d) }
| instp.127  { RLC (HL) }
| instp.128  { RLC (IX+d) }
| instp.129  { RLC (IY+d) }
| instp.131  { RL (HL) }
| instp.132  { RL (IX+d) }
| instp.133  { RL (IY+d) }
| instp.135  { RRC (HL) }
| instp.136  { RRC (IX+d) }
| instp.137  { RRC (IY+d) }
| instp.139  { RR (HL) }
| instp.140  { RR (IX+d) }
| instp.141  { RR (IY+d) }
| instp.143  { SLA (HL) }
| instp.144  { SLA (IX+d) }
| instp.145  { SLA (IY+d) }
| instp.147  { SRL (HL) }
| instp.148  { SRL (IX+d) }
| instp.149  { SRL (IY+d) }
| instp.151  { SRA (HL) }
| instp.152  { SRA (IX+d) }
| instp.153  { SRA (IY+d) }
| instp.154  { RLD }
| instp.155  { RRD }
| instp.161  { BIT b,(HL) }
| instp.162  { SET b,(IX+d) }
| instp.163  { SET b,(IY+d) }
| instp.165  { RES b,(HL) }
| instp.166  { RES b,(IX+d) }
| instp.167  { RES b,(IY+d) }
| instp.179  { CALL n'n }
| instp.185  { RST p }
| instp.187  { OUT (n),A }
| instp.189  { OUT (C),r }
| instp.190  { INI }
| instp.191  { IND }
| instp.192  { INIR }
| instp.193  { INDR }
| instp.194  { OUTI }
| instp.195  { OUTD }
| instp.196  { OTIR }
| instp.197  { OTDR }
| instp.198  { NMI }
```

```
         | instp.199  { INT }
         )
         dir=1;
      endif
   endif

   if (instp.180&anct1p&ccenm) dir=1; endif { CALL cc,n'n }

   if (anct2p)
      if ( instp.22   { LD (n'n),HL }
         | instp.32   { PUSH qq }
         | instp.41   { EX (SP),HL }
         | instp.24   { LD (n'n),dd }
         | instp.26   { LD (n'n),IX }
         | instp.28   { LD (n'n),IY }
         | instp.34   { PUSH IX }
         | instp.35   { PUSH IY }
         | instp.42   { EX (SP),IX }
         | instp.43   { EX (SP),IY }
         | instp.179  { CALL n'n }
         | instp.185  { RST p }
         | instp.198  { NMI }
         | instp.199  { INT }
         )
         dir=1;
      endif
   endif

   if (instp.180&anct2p&ccenm) dir=1; endif   { CALL cc,n'n }

{ ------------------------------------------------------------------- }
{    I/O メモリ切り替え                                    }
{ ------------------------------------------------------------------- }
   if (anct1p)
      if ( instp.186   { IN A,(n) }
         | instp.187   { OUT (n),A }
         | instp.188   { IN r,(C) }
         | instp.189   { OUT (C),r }
         | instp.194   { OUTI }
         | instp.195   { OUTD }
         | instp.196   { OTIR }
         | instp.197   { OTDR }
         )
         mio=1;
      endif
   endif

   if (anct2p)
      if ( instp.190   { INI }
         | instp.191   { IND }
         | instp.192   { INIR }
         | instp.193   { INDR }
         )
         mio=1;
      endif
   endif

{ ------------------------------------------------------------------- }
{    データバスのデータ                                    }
{ ------------------------------------------------------------------- }
   if (instp.189&anct1p) chgdataout1p=1; endif { OUT (C),r }

   if (anct1p)
```

```
   if ( instp.44          { LDI }
      | instp.45          { LDD }
      | instp.46          { LDIR }
      | instp.47          { LDDR }
      | instp.179         { CALL n'n }
      | instp.180         { CALL cc,n'n }
      | instp.190         { INI }
      | instp.191         { IND }
      | instp.192         { INIR }
      | instp.193         { INDR }
      | instp.194         { OUTI }
      | instp.195         { OUTD }
      | instp.196         { OTIR }
      | instp.197         { OTDR }
      | instp.198         { NMI }
      | instp.199         { INT }
      )
      chgdataout2p=1;
   endif
endif

if (anct2p)
   if ( instp.179   { CALL n'n }
      | instp.180   { CALL cc,n'n }
      | instp.198   { NMI }
      | instp.199   { INT }
      )
      chgdataout3p=1;
   endif
endif

if (anct1p)
   if ( instp.93   { INC (HL)   }
      | instp.94   { INC (IX+d) }
      | instp.95   { INC (IY+d) }
      | instp.97   { DEC (HL) }
      | instp.98   { DEC (IX+d) }
      | instp.99   { DEC (IY+d) }
      | instp.127   { RLC (HL) }
      | instp.128   { RLC (IX+d) }
      | instp.129   { RLC (IY+d) }
      | instp.131   { RL (HL) }
      | instp.132   { RL (IX+d) }
      | instp.133   { RL (IY+d) }
      | instp.135   { RRC (HL) }
      | instp.136   { RRC (IX+d) }
      | instp.137   { RRC (IY+d) }
      | instp.139   { RR (HL) }
      | instp.140   { RR (IX+d) }
      | instp.141   { RR (IY+d) }
      | instp.143   { SLA (HL) }
      | instp.144   { SLA (IX+d) }
      | instp.145   { SLA (IY+d) }
      | instp.147   { SRL (HL) }
      | instp.148   { SRL (IX+d) }
      | instp.149   { SRL (IY+d) }
      | instp.151   { SRA (HL) }
      | instp.152   { SRA (IX+d) }
      | instp.153   { SRA (IY+d) }
      | instp.161   { BIT b,(HL) }
      | instp.162   { SET b,(IX+d) }
```

```
      | instp.163   { SET b,(IY+d) }
      | instp.165   { RES b,(HL) }
      | instp.166   { RES b,(IX+d) }
      | instp.167   { RES b,(IY+d) }
      )
      chgdataout4p=1;
   endif
endif

if (anct1p)
   if ( instp.4   { LD (HL),r }
      | instp.8   { LD (IX[IY]+d),r }
      )
      chgdataout5p=1;
   endif
endif

if (instp.24&anct1p) chgdataout6p=1; endif   { LD (n'n),dd }
if (instp.24&anct2p) chgdataout7p=1; endif
if (instp.32&anct1p) chgdataout8p=1; endif   { PUSH qq }
if (instp.32&anct2p) chgdataout9p=1; endif

if (anct1p)
   if ( instp.6   { LD (HL),n }
      | instp.9   { LD (IX[IY]+d),n }
      )
      chgdataout10p=1;
   endif
endif

if (anct1p)
   if ( instp.34   { PUSH IX }
      | instp.42   { EX (SP),IX }
      )
      chgdataout11p=1;
   endif
endif

if (instp.26&anct2p) chgdataout11p=1; endif { LD (n'n),IX }
if (instp.26&anct1p) chgdataout12p=1; endif

if (anct2p)
   if ( instp.34   { PUSH IX }
      | instp.42   { EX (SP),IX }
      )
      chgdataout12p=1;
   endif
endif

if (anct1p)
   if ( instp.35   { PUSH IY }
      | instp.43   { EX (SP),IY }
      )
      chgdataout13p=1;
   endif
endif

if (instp.28&anct2p) chgdataout13p=1; endif { LD (n'n),IY }
if (instp.28&anct1p) chgdataout14p=1; endif

if (anct2p)
   if ( instp.35   { PUSH IY }
      | instp.43   { EX (SP),IY }
      )
```

```
      chgdataout14p=1;
   endif
endif

if (anct1p)
   if ( instp.11      { LD (BC),A }
      | instp.13      { LD (DE),A }
      | instp.15      { LD (n'n),A }
      | instp.187     { OUT (n),a }
      )
      chgdataout15p=1;
   endif
endif

if (instp.22)                        { LD (n'n),HL }
   if (anct1p) dataout=regH; endif
   if (anct2p) dataout=regL; endif
endif

if (instp.41)                        { EX (SP),HL }
   if (anct1p) dataout=regL; endif
   if (anct2p) dataout=regH; endif
endif

if (instp.154&anct1p)  { RLD }
   dataout.0:3=regA.0:3;
   dataout.4:7=reg0T.0:3;
endif

if (instp.155&anct1p)  { RRD }
   dataout.0:3=reg0T.4:7;
   dataout.4:7=regA.0:3;
endif

if (instp.185) { RST p }
   if (anct1p) dataout=regPC.0:7; endif
   if (anct2p) dataout=regPC.8:15; endif
endif

if (chgdataout1p)
   switch(code0d.3:5)
      case 0: dataout=regB;
      case 1: dataout=regC;
      case 2: dataout=regD;
      case 3: dataout=regE;
      case 4: dataout=regH;
      case 5: dataout=regL;
      case 7: dataout=regA;
   endswitch
endif

if (chgdataout2p) dataout=reg0T; endif
if (chgdataout3p) dataout=reg1T; endif
if (chgdataout4p) dataout=alu8out.0:7; endif

if (chgdataout5p)
   switch(code0d.0:2)
      case 0: dataout=regB;
      case 1: dataout=regC;
      case 2: dataout=regD;
      case 3: dataout=regE;
      case 4: dataout=regH;
      case 5: dataout=regL;
      case 7: dataout=regA;
   endswitch
```

```
   endif

   if (chgdataout6p)
      switch(code0d.4:5)
         case 0: dataout=regB;
         case 1: dataout=regD;
         case 2: dataout=regH;
         case 3: dataout=regSP.8:15;
      endswitch
   endif

   if (chgdataout7p)
      switch(code0d.4:5)
         case 0: dataout=regC;
         case 1: dataout=regE;
         case 2: dataout=regL;
         case 3: dataout=regSP.0:7;
      endswitch
   endif

   if (chgdataout8p)
      switch(code0d.4:5)
         case 0: dataout=regC;
         case 1: dataout=regE;
         case 2: dataout=regL;
         case 3: dataout=regF;
      endswitch
   endif

   if (chgdataout9p)
      switch(code0d.4:5)
         case 0: dataout=regB;
         case 1: dataout=regD;
         case 2: dataout=regH;
         case 3: dataout=regA;
      endswitch
   endif

   if (chgdataout10p) dataout=code1d; endif

   if (chgdataout11p) dataout=regIX.0:7; endif
   if (chgdataout12p) dataout=regIX.8:15; endif
   if (chgdataout13p) dataout=regIY.0:7; endif
   if (chgdataout14p) dataout=regIY.8:15; endif
   if (chgdataout15p) dataout=regA; endif

{ ------------------------------------------------------------------- }
{    プログラムカウンタ停止                                  }
{ ------------------------------------------------------------------- }
   if (anct0p&ancode104p) waitpc=1; endif { HALT }

   if (anct1p)
      if ( instp.2    { LD ra,rb }
         | instp.3    { LD r,(HL) }
         | instp.4    { LD (HL),r }
         | instp.10   { LD A,(BC) }
         | instp.11   { LD (BC),A }
         | instp.12   { LD A,(DE) }
         | instp.13   { LD (DE),A }
         | instp.16   { LD A,I }
         | instp.17   { LD I,A }
         | instp.18   { LD A,R }
         | instp.19   { LD R,A }
         | instp.29   { LD SP,HL }
```

```
| instp.32   { PUSH qq }
| instp.33   { POP qq }
| instp.34   { PUSH IX }
| instp.35   { PUSH IY }
| instp.36   { POP IX }
| instp.37   { POP IY }
| instp.38   { EX DE,HL }
| instp.39   { EX AF,AF' }
| instp.40   { EXX }
| instp.41   { EX (SP),HL }
| instp.42   { EX (SP),IX }
| instp.43   { EX (SP),IY }
| instp.44   { LDI }
| instp.45   { LDD }
| instp.46   { LDIR }
| instp.47   { LDDR }
| instp.48   { CPI }
| instp.49   { CPD }
| instp.50   { CPIR }
| instp.51   { CPDR }
| instp.52   { ADD A,r }
| instp.54   { ADD A,(HL) }
| instp.57   { ADC A,r }
| instp.59   { ADC A,(HL) }
| instp.62   { SUB r }
| instp.64   { SUB (HL) }
| instp.67   { SBC A,r }
| instp.69   { SBC A,(HL) }
| instp.72   { AND r }
| instp.74   { AND (HL) }
| instp.77   { OR r }
| instp.79   { OR (HL) }
| instp.82   { XOR r }
| instp.84   { XOR (HL) }
| instp.87   { CP r }
| instp.89   { CP (HL) }
| instp.90   { CP (IX+d) }
| instp.91   { CP (IY+d) }
| instp.92   { INC r }
| instp.93   { INC (HL) }
| instp.94   { INC (IX+d) }
| instp.95   { INC (IY+d) }
| instp.96   { DEC r }
| instp.97   { DEC (HL) }
| instp.98   { DEC (IX+d) }
| instp.99   { DEC (IY+d) }
| instp.100  { DAA }
| instp.101  { CPL }
| instp.102  { NEG }
| instp.103  { CCF }
| instp.104  { SCF }
| instp.105  { HALT }
| instp.106  { EI }
| instp.107  { DI }
| instp.108  { IM 0 }
| instp.109  { IM 1 }
| instp.110  { IM 2 }
| instp.111  { ADD HL,ss }
| instp.112  { ADC HL,ss }
| instp.113  { SBC HL,ss }
```

```
| instp.114  { ADD IX,pp }
| instp.115  { ADD IY,rr }
| instp.116  { INC ss }
| instp.117  { INC IX }
| instp.118  { INC IY }
| instp.119  { DEC ss }
| instp.120  { DEC IX }
| instp.121  { DEC IY }
| instp.122  { RLCA }
| instp.123  { RLA }
| instp.124  { RRCA }
| instp.125  { RRA }
| instp.126  { RLC r }
| instp.127  { RLC (HL) }
| instp.128  { RLC (IX+d) }
| instp.129  { RLC (IY+d) }
| instp.130  { RL r }
| instp.131  { RL (HL) }
| instp.132  { RL (IX+d) }
| instp.133  { RL (IY+d) }
| instp.134  { RRC r }
| instp.135  { RRC (HL) }
| instp.136  { RRC (IX+d) }
| instp.137  { RRC (IY+d) }
| instp.138  { RR r }
| instp.139  { RR (HL) }
| instp.140  { RR (IX+d) }
| instp.141  { RR (IY+d) }
| instp.142  { SLA r }
| instp.143  { SLA (HL) }
| instp.144  { SLA (IX+d) }
| instp.145  { SLA (IY+d) }
| instp.146  { SRL r }
| instp.147  { SRL (HL) }
| instp.148  { SRL (IX+d) }
| instp.149  { SRL (IY+d) }
| instp.150  { SRA r }
| instp.151  { SRA (HL) }
| instp.152  { SRA (IX+d) }
| instp.153  { SRA (IY+d) }
| instp.154  { RLD }
| instp.155  { RRD }
| instp.156  { BIT b,r }
| instp.157  { BIT b,(HL) }
| instp.158  { BIT b,(IX+d) }
| instp.159  { BIT b,(IY+d) }
| instp.160  { SET b,r }
| instp.161  { SET b,(HL) }
| instp.162  { SET b,(IX+d) }
| instp.163  { SET b,(IY+d) }
| instp.164  { RES b,r }
| instp.165  { RES b,(HL) }
| instp.166  { RES b,(IX+d) }
| instp.167  { RES b,(IY+d) }
| instp.188  { IN r,(C) }
| instp.189  { OUT (C),r }
| instp.192  { INIR }
| instp.193  { INDR }
| instp.196  { OTIR }
| instp.197  { OTDR }
```

```
      )
      waitpc=1;
   endif
endif

if (anct2p)
   if ( instp.168          { JP n'n }
      | instp.169          { JP cc,n'n }
      | instp.179          { CALL n'n }
      | instp.180          { CALL cc,n'n }
      )
   else
      waitpc=1;
   endif
endif

if (anct3p)
   if ( instp.21  { LD HL,(n'n) }
      | instp.22  { LD (n'n),HL }
      | instp.23  { LD dd,(n'n) }
      | instp.24  { LD (n'n),dd }
      | instp.25  { LD IX,(n'n) }
      | instp.26  { LD (n'n),IX }
      | instp.27  { LD IY,(n'n) }
      | instp.28  { LD (n'n),IY }
      | instp.41  { EX (SP),HL }
      | instp.42  { EX (SP),IX }
      | instp.43  { EX (SP),IY }
      | instp.198 { NMI }
      | instp.199 { INT }
      )
      waitpc=1;
   endif
endif

if (anct4p)
   if ( instp.21  { LD HL,(n'n) }
      | instp.22  { LD (n'n),HL }
      | instp.23  { LD dd,(n'n) }
      | instp.24  { LD (n'n),dd }
      | instp.25  { LD IX,(n'n) }
      | instp.26  { LD (n'n),IX }
      | instp.27  { LD IY,(n'n) }
      | instp.28  { LD (n'n),IY }
      | instp.179 { CALL n'n }
      | instp.180 { CALL cc,n'n }
      )
   else
      waitpc=1;
   endif
endif

if (anct10p)
   if (ancode175p)
      switch(expinst)
         case 1: waitpc=1;         { JP (IX) }
      endswitch
   endif

   if (ancode176p)
      switch(expinst)
         case 2: waitpc=1;         { JP (IY) }
      endswitch
```

```
      endif

      switch(expinst)
         case 3:
            if (ancode189p) waitpc=1; endif { INI }
            if (ancode190p) waitpc=1; endif { IND }
            if (ancode193p) waitpc=1; endif { OUTI }
            if (ancode194p) waitpc=1; endif { OUTD }
      endswitch
   endif

{ ------------------------------------------------------------------- }
{    スタックポインタ                                    }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.20&anct1p) chgregSP5p=1; endif  { LD SP,n'n }
         if (instp.23&anct1p) chgregSP6p=1; endif  { LD SP,(n'n) }
         if (instp.23&anct2p) chgregSP7p=1; endif  { LD SP,(n'n) }
         if (instp.29&anct1p) chgregSP8p=1; endif  { LD SP,HL }
         if (instp.116&anct1p) chgregSP1p=1; endif { INC SP }
         if (instp.119&anct1p) chgregSP1p=1; endif { DEC SP }

         if (anct1p)
            if ( instp.32   { PUSH qq }
               | instp.34   { PUSH IX }
               | instp.35   { PUSH IY }
               | instp.198  { NMI }
               | instp.199  { INT }
               )
               chgregSP2p=1;
            endif
         endif

         if (anct2p)
            if ( instp.32   { PUSH qq }
               | instp.34   { PUSH IX }
               | instp.35   { PUSH IY }
               | instp.179  { CALL n'n }
               | instp.185  { RST p }
               | instp.198  { NMI }
               | instp.199  { INT }
               )
               chgregSP2p=1;
            endif
         endif

         if (instp.179&anct3p) chgregSP2p=1; endif  { CALL n'n }

         if (anct1p)
            if ( instp.36   { POP IX }
               | instp.37   { POP IY }
               | instp.33   { POP qq }
               | instp.181  { RET }
               | instp.182  { RET cc }
               | instp.183  { RETI }
               | instp.184  { RETN }
               )
               chgregSP3p=1;
            endif
         endif

         if (anct2p)
```

```
            if ( instp.36   { POP IX }
               | instp.37   { POP IY }
               | instp.181  { RET }
               | instp.182  { RET cc }
               | instp.183  { RETI }
               | instp.184  { RETN }
               | instp.33   { POP qq }
               )
               chgregSP3p=1;
            endif
         endif

         if (instp.30&anct1p) chgregSP9p=1; endif   { LD SP,IX }
         if (instp.31&anct1p) chgregSP10p=1; endif  { LD SP,IY }

         if (instp.180) { CALL cc,n'n }
            if (anct3p)
               if (ccenm)
                  chgregSP2p=1;
               else
                  chgregSP4p=1;
               endif
            endif

            if (anct2p)
               if (ccenm)
                  chgregSP2p=1;
               else
                  chgregSP4p=1;
               endif
            endif
         endif
      else
         chgregSP4p=1;
      endif
   endif

   if (chgregSP1p)
      switch(code0d.4:5)
         case 3:  regSP=alu16out.0:15;
         default: regSP=regSP;
      endswitch
   endif

   if (chgregSP2p) regSP=mempoint1m; endif
   if (chgregSP3p) regSP=mempoint1p; endif
   if (chgregSP4p|chgregSP11p) regSP=regSP; endif

   if (chgregSP5p)
      switch(code0d.4:5)
         case 3: regSP.0:7=code2d;  { LD SP,n'n }
                 regSP.8:15=code1d;
         default:chgregSP4p=1;
      endswitch
   endif

   if (chgregSP6p)
      switch(code0d.4:5)
         case 3: regSP.8:15=DATAIN;  { LD SP,(n'n) }
                 regSP.0:7=regSP.0:7;
         default:chgregSP4p=1;
      endswitch
   endif

   if (chgregSP7p)
```

```
      switch(code0d.4:5)
         case 3: regSP.0:7=DATAIN;  { LD SP,(n'n) }
                 regSP.8:15=regSP.8:15;
         default:chgregSP4p=1;
      endswitch
   endif

   if (chgregSP8p)
      regSP.0:7=regL;
      regSP.8:15=regH;
   endif

   if (chgregSP9p) regSP=regIX; endif
   if (chgregSP10p) regSP=regIY; endif

   if ( chgregSP1p
      | chgregSP2p
      | chgregSP3p
      | chgregSP4p
      | chgregSP5p
      | chgregSP6p
      | chgregSP7p
      | chgregSP8p
      | chgregSP9p
      | chgregSP10p
      )
   else
      if (RESET)
         regSP=0;
      else
         if (ancode184p)      { RST p }
            if (anct0p)
               regSP=regSP-1;
            else
               chgregSP11p=1;
            endif
         else
            chgregSP11p=1;
         endif
      endif
   endif

{ ------------------------------------------------------------------- }
{    プログラムカウンタ                                  }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (!waitpc)
            if (instp.168&anct1p) chgregPC1p=1; endif { JP n'n }

            if (instp.169&anct1p)
               switch(code0d.3:5)
                  case 0: if (!flgZ)  chgregPC1p=1; else chgregPC2p=1; endif { JP NZ,n'n }
                  case 1: if (flgZ)   chgregPC1p=1; else chgregPC2p=1; endif { JP Z,n'n }
                  case 2: if (!flgC)  chgregPC1p=1; else chgregPC2p=1; endif { JP NC,n'n }
                  case 3: if (flgC)   chgregPC1p=1; else chgregPC2p=1; endif { JP C,n'n }
                  case 4: if (!flgPV) chgregPC1p=1; else chgregPC2p=1; endif { JP PO,n'n }
                  case 5: if (flgPV)  chgregPC1p=1; else chgregPC2p=1; endif { JP PE,n'n }
                  case 6: if (!flgS)  chgregPC1p=1; else chgregPC2p=1; endif { JP P,n'n }
                  case 7: if (flgS)   chgregPC1p=1; else chgregPC2p=1; endif { JP M,n'n }
               endswitch
            endif

            if (instp.170&anct1p) chgregPC3p=1; endif { JR e }
```

```
            if (instp.171&anct1p)                                    { JR C,e }
               if (flgC)  chgregPC3p=1; else chgregPC2p=1; endif
            endif

            if (instp.172&anct1p)                                    { JR NC,e }
               if (!flgC) chgregPC3p=1; else chgregPC2p=1; endif
            endif

            if (instp.173&anct1p)                                    { JR Z,e }
               if (flgZ)  chgregPC3p=1; else chgregPC2p=1; endif
            endif

            if (instp.174&anct1p)                                    { JR NZ,e }
               if (!flgZ) chgregPC3p=1; else chgregPC2p=1; endif
            endif

            if (instp.178&anct1p)                                    { DJNZ e }
               if (regB1m!=0) chgregPC3p=1; else chgregPC2p=1; endif
            endif

            if (anct1p)
               if ( instp.179 { CALL n'n }
                  | instp.180 { CALL cc,n'n }
                  )
                  chgregPC7p=1;
               endif
            endif

            if (anct1p)
               if ( instp.181 { RET }
                  | instp.182 { RET cc }
                  | instp.183 { RETI }
                  | instp.184 { RETN }
                  )
                  chgregPC8p=1;
               endif
            endif

            if (instp.182&anct2p) chgregPC8p=1; endif
            if (instp.185&anct1p) chgregPC10p=1; endif { RST p }
            if (instp.198&anct1p) chgregPC11p=1; endif { NMI }
            if (instp.199&anct1p) chgregPC12p=1; endif { INT }
         endif
      endif
   endif


   if (chgregPC1p)
      regPC.0:7=code1d;
      regPC.8:15=DATAIN;
   endif

   if (chgregPC2p|chgregPC14p) regPC=regPC+1; endif
   if (chgregPC3p) regPC=regPC+addpc+1; endif

   if (chgregPC13p)
      regPC.0:7=regL;
      regPC.8:15=regH;
   endif

   if (chgregPC15p) regPC=regIX; endif
   if (chgregPC16p) regPC=regIY; endif

   if (chgregPC7p)
      regPC.0:7=code2d;
```

```
   regPC.8:15=code1d;
endif

if (chgregPC8p)
   regPC.0:7=reg0T;
   regPC.8:15=DATAIN;
endif

if (chgregPC17p) regPC=regPC+3; endif

if (chgregPC10p)
   switch(code0d.3:5)
      case 0: regPC=0x0000;
      case 1: regPC=0x0008;
      case 2: regPC=0x0010;
      case 3: regPC=0x0018;
      case 4: regPC=0x0020;
      case 5: regPC=0x0028;
      case 6: regPC=0x0030;
      case 7: regPC=0x0038;
   endswitch
endif

if (chgregPC11p) regPC=0x66; endif

if (chgregPC12p)
   switch(im)
      case 0: regPC=intaddress;
      case 1: regPC=0x38;
      case 2: regPC=intaddress;
   endswitch
endif

if ( chgregPC1p
   | chgregPC2p
   | chgregPC3p
   | chgregPC7p
   | chgregPC8p
   | chgregPC10p
   | chgregPC11p
   | chgregPC12p
   )
else
   if (RESET)
      regPC=0;
   else
      if (WAIT|waitpc)
         regPC=regPC;
      else
         if (anct0p)
            switch(expinst)
               case 0:
                  if (ancode174p) chgregPC13p=1; endif                { JP (HL) }
                  if (ancode179p)                                     { CALL cc,n'n }
                     if (ccen) chgregPC14p=1; else chgregPC17p=1; endif
                  endif

                  if (!ancode174p&!ancode179p) chgregPC14p=1; endif
               case 1:
                  if (ancode175p) chgregPC15p=1; else  chgregPC14p=1; endif { JP (IX) }
               case 2:
                  if (ancode176p) chgregPC16p=1; else  chgregPC14p=1; endif { JP (IY) }
               default: chgregPC14p=1;
            endswitch
         else
            chgregPC14p=1;
```

```
            endif
         endif
      endif
   endif

{ ------------------------------------------------------------------- }
{    条件呼び出し復帰判断                                  }
{ ------------------------------------------------------------------- }
   switch(DATAIN.3:5)
      case 0: if (!flgZ)  ccen=1; endif { NZ }
      case 1: if (flgZ)   ccen=1; endif { Z }
      case 2: if (!flgC)  ccen=1; endif { NC }
      case 3: if (flgC)   ccen=1; endif { C }
      case 4: if (!flgPV) ccen=1; endif { PO }
      case 5: if (flgPV)  ccen=1; endif { PE }
      case 6: if (!flgS)  ccen=1; endif { P }
      case 7: if (flgS)   ccen=1; endif { M }
   endswitch

{ ------------------------------------------------------------------- }
{    条件呼び出し復帰判断記憶                                }
{ ------------------------------------------------------------------- }
   if (!WAIT)
      if (anct0p)
         if (ancode179p|ancode181p)    { CALL n'n }
            ccenm=ccen;                { RET }
         endif
      else
         ccenm=ccenm;
      endif
   else
      ccenm=ccenm;
   endif

{ ------------------------------------------------------------------- }
{    レジスタ B 減算値                                    }
{ ------------------------------------------------------------------- }
   regB1m=regB-1;

{ ------------------------------------------------------------------- }
{    相対分岐加算値                                       }
{ ------------------------------------------------------------------- }
   addpc.0:7=DATAIN;

   if (DATAIN.7)
      addpc.8:15=0xff;
   else
      addpc.8:15=0;
   endif

{ ------------------------------------------------------------------- }
{    レジスタ F                                           }
{ ------------------------------------------------------------------- }
   regF.0=flgC;
   regF.1=flgN;
   regF.2=flgPV;
   regF.4=flgH;
   regF.6=flgZ;
   regF.7=flgS;

{ ------------------------------------------------------------------- }
{    割り込みモード                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
```

```
      im=0;
   else
      if (!WAIT)
         if (anct1p)
            if (instp.108) im=0; endif      { IM 0 }
            if (instp.109) im=1; endif      { IM 1 }
            if (instp.110) im=2; endif      { IM 2 }

            if (instp.108|instp.109|instp.110)
            else
               im=im;
            endif
         else
            im=im;
         endif
      else
         im=im;
      endif
   endif

{ ------------------------------------------------------------------- }
{    フラグ EI                                          }
{ ------------------------------------------------------------------- }
   if (RESET)
      flgEI=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)  chgflgEI2p=1; endif    { EX AF,AF' }
         if (instp.106&anct0p) chgflgEI1p=1; endif    { EI }
         if (instp.107&anct1p) chgflgEI0p=1; endif    { DI }
         if (instp.199&anct1p) chgflgEI0p=1; endif    { INT }

         if (chgflgEI0p|chgflgEI1p|chgflgEI2p)
         else
            flgEI=flgEI;
         endif
      else
         flgEI=flgEI;
      endif
   endif

   if (chgflgEI0p) flgEI=0; endif
   if (chgflgEI1p) flgEI=1; endif
   if (chgflgEI2p) flgEI=flgREI; endif

{ ------------------------------------------------------------------- }
{    フラグ N                                           }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
               if (instp.33&anct2p) { POP AF }
                  switch(code0d.4:5)
                     case 3: chgflgN2p=1;
                  endswitch
               endif

               if (instp.39&anct1p) chgflgN3p=1; endif { EX  AF,AF' }

               if (anct1p)
                  if ( instp.44 { LDI }
                     | instp.45 { LDD }
                     | instp.46 { LDIR }
                     | instp.47 { LDDR }
                     | instp.52 { ADD A,r }
```

```
| instp.53 { ADD A,n }
| instp.54 { ADD A,(HL) }
| instp.55 { ADD A,(IX+d) }
| instp.56 { ADD A,(IY+d) }
| instp.57 { ADC A,r }
| instp.58 { ADC A,n }
| instp.59 { ADC A,(HL) }
| instp.60 { ADC A,(IX+d) }
| instp.61 { ADC A,(IY+d) }
| instp.72 { AND r }
| instp.73 { AND n }
| instp.74 { AND (HL) }
| instp.75 { AND (IX[IY]+d) }
| instp.77 { OR r }
| instp.78 { OR n }
| instp.79 { OR (HL) }
| instp.80 { OR (IX+d) }
| instp.81 { OR (IY+d) }
| instp.82 { XOR r }
| instp.83 { XOR n }
| instp.84 { XOR (HL) }
| instp.85 { XOR (IX+d) }
| instp.86 { XOR (IY+d) }
| instp.92 { INC r }
| instp.93 { INC (HL) }
| instp.94 { INC (IX+d) }
| instp.95 { INC (IY+d) }
| instp.103 { CCF }
| instp.104 { SCF }
| instp.111 { ADD HL,ss }
| instp.112 { ADC HL,ss }
| instp.114 { ADD IX,pp }
| instp.115  { ADD IY,rr }
| instp.122 { RLCA }
| instp.123 { RLA }
| instp.124 { RRCA }
| instp.125 { RRA }
| instp.126 { RLC r }
| instp.127 { RLC (HL) }
| instp.128 { RLC (IX+d) }
| instp.129 { RLC (IY+d) }
| instp.130 { RL r }
| instp.131 { RL (HL) }
| instp.132 { RL (IX+d) }
| instp.133 { RL (IY+d) }
| instp.134 { RRC r }
| instp.135 { RRC (HL) }
| instp.136 { RRC (IX+d) }
| instp.137 { RRC (IY+d) }
| instp.138 { RR r }
| instp.139 { RR (HL) }
| instp.140 { RR (IX+d) }
| instp.141 { RR (IY+d) }
| instp.142 { SLA r }
| instp.143 { SLA (HL) }
| instp.144 { SLA (IX+d) }
| instp.145 { SLA (IY+d) }
| instp.146 { SRL r }
| instp.147 { SRL (HL) }
| instp.148 { SRL (IX+d) }
```

```
                  | instp.149 { SRL (IY+d) }
                  | instp.150 { SRA r }
                  | instp.151 { SRA (HL) }
                  | instp.152 { SRA (IX+d) }
                  | instp.153 { SRA (IY+d) }
                  | instp.154 { RLD }
                  | instp.155 { RRD }
                  | instp.156 { BIT b,r }
                  | instp.157 { BIT b,(HL) }
                  | instp.158 { BIT b,(IX+d) }
                  | instp.159 { BIT b,(IY+d) }
                  | instp.188 { IN r,(C) }
                  | instp.16  { LD A,I }
                  | instp.18  { LD A,R }
                  )
                   chgflgN0p=1;
               endif
            endif

            if (anct1p)
               if ( instp.48 { CPI }
                  | instp.49 { CPD }
                  | instp.50 { CPIR }
                  | instp.51 { CPDR }
                  | instp.62 { SUB r }
                  | instp.63 { SUB n }
                  | instp.64 { SUB (HL) }
                  | instp.65 { SUB (IX+d) }
                  | instp.66 { SUB (IY+d) }
                  | instp.67 { SBC A,r }
                  | instp.68 { SBC A,n }
                  | instp.69 { SBC A,(HL) }
                  | instp.70 { SBC A,(IX+d) }
                  | instp.71 { SBC A,(IY+d) }
                  | instp.87 { CP r }
                  | instp.88 { CP n }
                  | instp.89 { CP (HL) }
                  | instp.90 { CP (IX+d) }
                  | instp.91 { CP (IY+d) }
                  | instp.96 { DEC r }
                  | instp.97 { DEC (HL) }
                  | instp.98 { DEC (IX+d) }
                  | instp.99 { DEC (IY+d) }
                  | instp.101 { CPL }
                  | instp.102 { NEG }
                  | instp.113 { SBC HL,ss }
                  | instp.190 { INI }
                  | instp.191 { IND }
                  | instp.192 { INIR }
                  | instp.193 { INDR }
                  | instp.194 { OUTI }
                  | instp.195 { OUTD }
                  | instp.196 { OTIR }
                  | instp.197 { OTDR }
                   )
                  chgflgN1p=1;
               endif
            endif
      endif
   endif
```

```
   if (chgflgN0p) flgN=0; endif
   if (chgflgN1p) flgN=1; endif
   if (chgflgN2p) flgN=DATAIN.1; endif
   if (chgflgN3p) flgN=flgRN; endif

   if (chgflgN0p|chgflgN1p|chgflgN2p|chgflgN3p)
   else
      if (RESET)
         flgN=0;
      else
         flgN=flgN;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    パリティ計算                                                         }
{ ------------------------------------------------------------------------ }
   switch(alu8out.0:3)
      case 0b0000: parityl=0;
      case 0b0001: parityl=1;
      case 0b0010: parityl=1;
      case 0b0011: parityl=2;
      case 0b0100: parityl=1;
      case 0b0101: parityl=2;
      case 0b0110: parityl=2;
      case 0b0111: parityl=3;
      case 0b1000: parityl=1;
      case 0b1001: parityl=2;
      case 0b1010: parityl=2;
      case 0b1011: parityl=3;
      case 0b1100: parityl=2;
      case 0b1101: parityl=3;
      case 0b1110: parityl=3;
      case 0b1111: parityl=4;
   endswitch

   switch(alu8out.4:7)
      case 0b0000: parityh=0;
      case 0b0001: parityh=1;
      case 0b0010: parityh=1;
      case 0b0011: parityh=2;
      case 0b0100: parityh=1;
      case 0b0101: parityh=2;
      case 0b0110: parityh=2;
      case 0b0111: parityh=3;
      case 0b1000: parityh=1;
      case 0b1001: parityh=2;
      case 0b1010: parityh=2;
      case 0b1011: parityh=3;
      case 0b1100: parityh=2;
      case 0b1101: parityh=3;
      case 0b1110: parityh=3;
      case 0b1111: parityh=4;
   endswitch

   switch(parityl.0,parityh.0)
      case 0,0: parity=1;
      case 0,1: parity=0;
      case 1,0: parity=0;
      case 1,1: parity=1;
   endswitch

{ ------------------------------------------------------------------------ }
{    フラグ PV                                                            }
{ ------------------------------------------------------------------------ }
   if (!RESET)
```

```
if (!WAIT)
   if (instp.33&anct2p) { POP AF }
      switch(code0d.4:5)
         case 3: chgflgPV7p=1;
      endswitch
   endif

   if (instp.16&anct1p) chgflgPV8p=1; endif { LD A,I }
   if (instp.18&anct1p) chgflgPV9p=1; endif { LD A,R }
   if (instp.39&anct1p) chgflgPV10p=1; endif { EX  AF,AF' }

   if (anct1p)
      if ( instp.46 { LDIR }
         | instp.47 { LDDR }
         | instp.190 { INI }
         | instp.191 { IND }
         | instp.192 { INIR }
         | instp.193 { INDR }
         | instp.194 { OUTI }
         | instp.195 { OUTD }
         | instp.196 { OTIR }
         | instp.197 { OTDR }
         | instp.156 { BIT b,r }
         | instp.157 { BIT b,(HL) }
         | instp.158 { BIT b,(IX+d) }
         | instp.159 { BIT b,(IY+d) }
         )
         chgflgPV11p=1;
      endif
   endif

   if (anct1p)
      if ( instp.67 { SBC A,r (V) }
         | instp.68 { SBC A,n (V) }
         | instp.69 { SBC A,(HL) (V) }
         | instp.70 { SBC A,(IX+d) (V) }
         | instp.71 { SBC A,(IY+d) (V) }
         )
         chgflgPV1p=1;
      endif
   endif

   if (anct1p)
      if ( instp.62 { SUB r (V) }
         | instp.63 { SUB n (V) }
         | instp.64 { SUB (HL) (V) }
         | instp.65 { SUB (IX+d) (V) }
         | instp.66 { SUB (IY+d) (V) }
         | instp.87 { CP r (V) }
         | instp.88 { CP n (V) }
         | instp.89 { CP (HL) (V) }
         | instp.90 { CP (IX+d) (V) }
         | instp.91 { CP (IY+d) (V) }
         )
         chgflgPV2p=1;
      endif
   endif

   if (anct1p)
      if ( instp.52 { ADD A,r (V) }
         | instp.53 { ADD A,n (V) }
         | instp.54 { ADD A,(HL) (V) }
         | instp.55 { ADD A,(IX+d) (V) }
```

```
      | instp.56 { ADD A,(IY+d) (V) }
      | instp.57 { ADC A,r (V) }
      | instp.58 { ADC A,n (V) }
      | instp.59 { ADC A,(HL) (V) }
      | instp.60 { ADC A,(IX+d) (V) }
      | instp.61 { ADC A,(IY+d) (V) }
      | instp.92 { INC r (V) }
      | instp.93 { INC (HL) (V) }
      | instp.94 { INC (IX+d) (V) }
      | instp.95 { INC (IY+d) (V) }
      | instp.96 { DEC r (V) }
      | instp.97 { DEC (HL) (V) }
      | instp.98 { DEC (IX+d) (V) }
      | instp.99 { DEC (IY+d) (V) }
      | instp.102 { NEG (P) }
      )
      chgflgPV3p=1;
   endif
endif

if (anct1p)
   if ( instp.44 { LDI }
      | instp.45 { LDD }
      | instp.48 { CPI }
      | instp.49 { CPD }
      | instp.50 { CPIR }
      | instp.51 { CPDR }
      )
      chgflgPV4p=1;
   endif
endif

if (instp.112&anct1p) chgflgPV5p=1; endif { ADC HL,ss }
if (instp.113&anct1p) chgflgPV6p=1; endif { SBC HL,ss }

if (anct1p)
   if ( instp.72 { AND r (P) }
      | instp.73 { AND n (P) }
      | instp.74 { AND (HL) (P) }
      | instp.75 { AND (IX[IY]+d) (P) }
      | instp.77 { OR r (P) }
      | instp.78 { OR n (P) }
      | instp.79 { OR (HL) (P) }
      | instp.80 { OR (IX+d) (P) }
      | instp.81 { OR (IY+d) (P) }
      | instp.82 { XOR r (P) }
      | instp.83 { XOR n (P) }
      | instp.84 { XOR (HL) (P) }
      | instp.85 { XOR (IX+d) (P) }
      | instp.86 { XOR (IY+d) (P) }
      | instp.100 { DAA (P) }
      | instp.126 { RLC r }
      | instp.127 { RLC (HL) }
      | instp.128 { RLC (IX+d) }
      | instp.129 { RLC (IY+d) }
      | instp.130 { RL r }
      | instp.131 { RL (HL) }
      | instp.132 { RL (IX+d) }
      | instp.133 { RL (IY+d) }
      | instp.134 { RRC r }
      | instp.135 { RRC (HL) }
```

```
            | instp.136 { RRC (IX+d) }
            | instp.137 { RRC (IY+d) }
            | instp.138 { RR r }
            | instp.139 { RR (HL) }
            | instp.140 { RR (IX+d) }
            | instp.141 { RR (IY+d) }
            | instp.142 { SLA r }
            | instp.143 { SLA (HL) }
            | instp.144 { SLA (IX+d) }
            | instp.145 { SLA (IY+d) }
            | instp.146 { SRL r }
            | instp.147 { SRL (HL) }
            | instp.148 { SRL (IX+d) }
            | instp.149 { SRL (IY+d) }
            | instp.150 { SRA r }
            | instp.151 { SRA (HL) }
            | instp.152 { SRA (IX+d) }
            | instp.153 { SRA (IY+d) }
            | instp.154 { RLD }
            | instp.155 { RRD }
            | instp.188 { IN r,(C) }
            )
            chgflgPV12p=1;
         endif
      endif
   endif
endif

if ( chgflgPV1p
   | chgflgPV2p
   | chgflgPV3p
   | chgflgPV4p
   | chgflgPV5p
   | chgflgPV6p
   | chgflgPV7p
   | chgflgPV8p
   | chgflgPV9p
   | chgflgPV10p
   | chgflgPV11p
   | chgflgPV12p
   )
else
   if (RESET)
      flgPV=0;
   else
      flgPV=flgPV;
   endif
endif

if (chgflgPV12p) flgPV=parity; endif
if (chgflgPV11p) flgPV=0; endif
if (chgflgPV10p) flgPV=flgRPV; endif
if (chgflgPV9p) flgPV=flgEI; endif
if (chgflgPV8p) flgPV=flgEI; endif
if (chgflgPV7p) flgPV=DATAIN.2; endif

if (chgflgPV6p)
   switch(alu16ina.15,alu16inb.15,alu16m.16,alu16m.15)
      case 0,1,1,1: flgPV=1;       { 1-(-32767)=32768 }
      case 1,0,0,0: flgPV=1;       { -32768-1=-32769 }
      default:
         switch(alu16ina.15,alu16inb.15,alu16out.16,alu16out.15)
            case 0,1,1,1: flgPV=1; { 1-(-32768)=32768 }
            case 1,0,0,0: flgPV=1; { -32768-1=-32769 }
```

```
            endswitch
         endswitch
      endif

      if (chgflgPV5p)
         switch(alu16ina.15,alu16inb.15,alu16out.16,alu16out.15)
            case 0,0,0,1: flgPV=1;       { 7fffh+1=8000h  }
            case 1,1,1,0: flgPV=1;       { 8000h+8000h=1.0 }
         endswitch
      endif

      if (chgflgPV4p)
         switch(regB)
            case 0:
               switch(regC)
                  case 1:  flgPV=0;
                  default: flgPV=1;
               endswitch
            default: flgPV=1;
         endswitch
      endif

      if (chgflgPV3p)
         switch(alu8ina.7,alu8inb.7,alu8out.8,alu8out.7)
            case 0,0,0,1: flgPV=1;       { 127+1=128  }
            case 1,1,1,0: flgPV=1;       { -128+(-128)=256 }
         endswitch
      endif

      if (chgflgPV2p)
         switch(alu8ina.7,alu8inb.7,alu8out.8,alu8out.7)
            case 0,1,1,1: flgPV=1;       { 1-(-128)=129 }
            case 1,0,0,0: flgPV=1;       { -128-1=-129 }
         endswitch
      endif

      if (chgflgPV1p)
         switch(alu8ina.7,alu8inb.7,alu8m.8,alu8m.7)
            case 0,1,1,1: flgPV=1;       { 1-(-128)=129 }
            case 1,0,0,0: flgPV=1;       { -128-1=-129 }
            default:
               switch(alu8ina.7,alu8inb.7,alu8out.8,alu8out.7)
                  case 0,1,1,1: flgPV=1; { 1-(-128)=129 }
                  case 1,0,0,0: flgPV=1; { -128-1=-129 }
               endswitch
         endswitch
      endif

{ ------------------------------------------------------------------- }
{    フラグ H                                          }
{ ------------------------------------------------------------------- }
      if (!RESET)
         if (!WAIT)
            if (instp.33&anct2p) { POP AF }
               switch(code0d.4:5)
                  case 3: chgflgH1p=1;
               endswitch
            endif

            if (instp.39&anct1p) chgflgH2p=1; endif { EX  AF,AF' }
            if (instp.100&anct1p) chgflgH3p=1; endif { DAA }

            if (anct1p)
               if ( instp.16 { LD A,I }
                  | instp.18 { LD A,R }
```

```
| instp.44 { LDI }
| instp.45 { LDD }
| instp.46 { LDIR }
| instp.47 { LDDR }
| instp.77 { OR r }
| instp.78 { OR n }
| instp.79 { OR (HL) }
| instp.80 { OR (IX+d) }
| instp.81 { OR (IY+d) }
| instp.82 { XOR r }
| instp.83 { XOR n }
| instp.84 { XOR (HL) }
| instp.85 { XOR (IX+d) }
| instp.86 { XOR (IY+d) }
| instp.103 { CCF }
| instp.104 { SCF }
| instp.111 { ADD HL,ss }
| instp.112 { ADC HL,ss }
| instp.113 { SBC HL,ss }
| instp.114 { ADD IX,pp }
| instp.115 { ADD IY,rr }
| instp.122 { RLCA }
| instp.123 { RLA }
| instp.124 { RRCA }
| instp.125 { RRA }
| instp.126 { RLC r }
| instp.127 { RLC (HL) }
| instp.128 { RLC (IX+d) }
| instp.129 { RLC (IY+d) }
| instp.130 { RL r }
| instp.131 { RL (HL) }
| instp.132 { RL (IX+d) }
| instp.133 { RL (IY+d) }
| instp.134 { RRC r }
| instp.135 { RRC (HL) }
| instp.136 { RRC (IX+d) }
| instp.137 { RRC (IY+d) }
| instp.138 { RR r }
| instp.139 { RR (HL) }
| instp.140 { RR (IX+d) }
| instp.141 { RR (IY+d) }
| instp.142 { SLA r }
| instp.143 { SLA (HL) }
| instp.144 { SLA (IX+d) }
| instp.145 { SLA (IY+d) }
| instp.146 { SRL r }
| instp.147 { SRL (HL) }
| instp.148 { SRL (IX+d) }
| instp.149 { SRL (IY+d) }
| instp.150 { SRA r }
| instp.151 { SRA (HL) }
| instp.152 { SRA (IX+d) }
| instp.153 { SRA (IY+d) }
| instp.154 { RLD }
| instp.155 { RRD }
| instp.188 { IN r,(C) }
| instp.190 { INI }
| instp.191 { IND }
| instp.192 { INIR }
| instp.193 { INDR }
```

```
      | instp.194 { OUTI }
      | instp.195 { OUTD }
      | instp.196 { OTIR }
      | instp.197 { OTDR }
      )
      chgflgH4p=1;
   endif
endif

if (anct1p)
   if ( instp.72 { AND r }
      | instp.73 { AND n }
      | instp.74 { AND (HL) }
      | instp.75 { AND (IX[IY]+d) }
      | instp.101 { CPL }
      | instp.156 { BIT b,r }
      | instp.157 { BIT b,(HL) }
      | instp.158 { BIT b,(IX+d) }
      | instp.159 { BIT b,(IY+d) }
      )
      chgflgH5p=1;
   endif
endif

if (anct1p)
   if ( instp.62  { SUB r }
      | instp.63  { SUB n }
      | instp.64  { SUB (HL) }
      | instp.65  { SUB (IX+d) }
      | instp.66  { SUB (IY+d) }
      | instp.67  { SBC A,r }
      | instp.68  { SBC A,n }
      | instp.69  { SBC A,(HL) }
      | instp.70  { SBC A,(IX+d) }
      | instp.71  { SBC A,(IY+d) }
      | instp.87  { CP r }
      | instp.88  { CP n }
      | instp.89  { CP (HL) }
      | instp.90  { CP (IX+d) }
      | instp.91  { CP (IY+d) }
      | instp.96  { DEC r }
      | instp.52  { ADD A,r }
      | instp.53  { ADD A,n }
      | instp.54  { ADD A,(HL) }
      | instp.55  { ADD A,(IX+d) }
      | instp.56  { ADD A,(IY+d) }
      | instp.57  { ADC A,r }
      | instp.58  { ADC A,n }
      | instp.59  { ADC A,(HL) }
      | instp.60  { ADC A,(IX+d) }
      | instp.61  { ADC A,(IY+d) }
      | instp.92  { INC r }
      | instp.102 { NEG }
      )
      chgflgH6p=1;
   endif
endif

if (anct2p)
   if ( instp.93  { INC (HL) }
      | instp.94  { INC (IX+d) }
      | instp.95  { INC (IY+d) }
```

```
               | instp.97  { DEC (HL) }
               | instp.98  { DEC (IX+d) }
               | instp.99  { DEC (IY+d) }
               )
               chgflgH6p=1;
            endif
         endif

      endif
   endif

   if (chgflgH1p) flgH=DATAIN.4; endif
   if (chgflgH2p) flgH=flgRH; endif
   if (chgflgH3p) flgH=daalA.4; endif
   if (chgflgH4p) flgH=0; endif
   if (chgflgH5p) flgH=1; endif
   if (chgflgH6p) flgH=alu4out.4; endif

   if ( chgflgH1p
      | chgflgH2p
      | chgflgH3p
      | chgflgH4p
      | chgflgH5p
      | chgflgH6p
      )
   else
      if (RESET)
         flgH=0;
      else
         flgH=flgH;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    フラグ Z                                                  }
{ ------------------------------------------------------------------------ }
   if (!RESET)
      if (!WAIT)
         if (instp.33&anct2p) { POP AF }
            switch(code0d.4:5)
               case 3: chgflgZ5p=1;
            endswitch
         endif

         if (instp.39&anct1p) chgflgZ6p=1; endif { EX  AF,AF' }
         if (instp.16&anct1p) chgflgZ7p=1; endif { LD A,I }
         if (instp.18&anct1p) chgflgZ10p=1; endif { LD A,R }
         if (instp.33&anct1p) chgflgZ5p=1; endif { POP AF }

         if (anct1p)
            if ( instp.48    { CPI }
               | instp.49    { CPD }
               | instp.50    { CPIR }
               | instp.51    { CPDR }
               )
               chgflgZ1p=1;
            endif
         endif

         if (anct1p)
            if ( instp.52    { ADD A,r }
               | instp.53    { ADD A,n }
               | instp.54    { ADD A,(HL) }
               | instp.55    { ADD A,(IX+d) }
```

```
| instp.56    { ADD A,(IY+d) }
| instp.57    { ADC A,r }
| instp.58    { ADC A,n }
| instp.59    { ADC A,(HL) }
| instp.60    { ADC A,(IX+d) }
| instp.61    { ADC A,(IY+d) }
| instp.62    { SUB r }
| instp.63    { SUB n }
| instp.64    { SUB (HL) }
| instp.65    { SUB (IX+d) }
| instp.66    { SUB (IY+d) }
| instp.67    { SBC A,r }
| instp.68    { SBC A,n }
| instp.69    { SBC A,(HL) }
| instp.70    { SBC A,(IX+d) }
| instp.71    { SBC A,(IY+d) }
| instp.72    { AND r }
| instp.73    { AND n }
| instp.74    { AND (HL) }
| instp.75    { AND (IX[IY]+d) }
| instp.77    { OR r }
| instp.78    { OR n }
| instp.79    { OR (HL) }
| instp.80    { OR (IX+d) }
| instp.81    { OR (IY+d) }
| instp.82    { XOR r }
| instp.83    { XOR n }
| instp.84    { XOR (HL) }
| instp.85    { XOR (IX+d) }
| instp.86    { XOR (IY+d) }
| instp.87    { CP r }
| instp.88    { CP n }
| instp.89    { CP (HL) }
| instp.90    { CP (IX+d) }
| instp.91    { CP (IY+d) }
| instp.92    { INC r }
| instp.93    { INC (HL) }
| instp.94    { INC (IX+d) }
| instp.95    { INC (IY+d) }
| instp.96    { DEC r }
| instp.97    { DEC (HL) }
| instp.98    { DEC (IX+d) }
| instp.99    { DEC (IY+d) }
| instp.102   { NEG }
| instp.126   { RLC r }
| instp.127   { RLC (HL) }
| instp.128   { RLC (IX+d) }
| instp.129   { RLC (IY+d) }
| instp.130   { RL r }
| instp.131   { RL (HL) }
| instp.132   { RL (IX+d) }
| instp.133   { RL (IY+d) }
| instp.134   { RRC r }
| instp.135   { RRC (HL) }
| instp.136   { RRC (IX+d) }
| instp.137   { RRC (IY+d) }
| instp.138   { RR r }
| instp.139   { RR (HL) }
| instp.140   { RR (IX+d) }
| instp.141   { RR (IY+d) }
```

```
            | instp.142   { SLA r }
            | instp.143   { SLA (HL) }
            | instp.144   { SLA (IX+d) }
            | instp.145   { SLA (IY+d) }
            | instp.146   { SRL r }
            | instp.147   { SRL (HL) }
            | instp.148   { SRL (IX+d) }
            | instp.149   { SRL (IY+d) }
            | instp.150   { SRA r }
            | instp.151   { SRA (HL) }
            | instp.152   { SRA (IX+d) }
            | instp.153   { SRA (IY+d) }
            | instp.156   { BIT b,r }
            | instp.157   { BIT b,(HL) }
            | instp.158   { BIT b,(IX+d) }
            | instp.159   { BIT b,(IY+d) }
            | instp.188   { IN r,(C) }
            | instp.190   { INI }
            | instp.191   { IND }
            | instp.194   { OUTI }
            | instp.195   { OUTD }
            )
            chgflgZ2p=1;
         endif
      endif

      if (anct1p)
         if ( instp.192   { INIR }
            | instp.193   { INDR }
            | instp.196   { OTIR }
            | instp.197   { OTDR }
            )
            chgflgZ11p=1;
         endif
      endif

      if (anct1p)
         if (instp.154)   { RLD }
            chgflgZ8p=1;
         endif
      endif

      if (anct1p)
         if (instp.155)   { RRD }
            chgflgZ9p=1;
         endif
      endif

      if (instp.100&anct1p) chgflgZ3p=1; endif { DAA }

      if (anct1p)
         if ( instp.112   { ADC HL,ss }
            | instp.113   { SBC HL,ss }
            )
            chgflgZ4p=1;
         endif
      endif
   endif
endif

if (chgflgZ1p)
   if (regA==DATAIN)
       flgZ=1;
```

```
   else
       flgZ=0;
   endif
endif

if (chgflgZ2p)
   switch(alu8out.0:7)
      case 0: flgZ=1;
   endswitch
endif

if (chgflgZ3p)
   switch(daahA.3:0,daalA.3:0)
      case 0,0: flgZ=1;
   endswitch
endif

if (chgflgZ4p)
   switch(alu16out.0:15)
      case 0: flgZ=1;
   endswitch
endif

if (chgflgZ5p) flgZ=DATAIN.6; endif
if (chgflgZ6p) flgZ=flgRZ; endif
if (chgflgZ7p) flgZ=regI==0; endif
if (chgflgZ10p) flgZ=regR==0; endif
if (chgflgZ11p) flgZ=1; endif

if (chgflgZ8p)
   switch(regA.4:7)
      case 0:
         switch(reg0T.4:7)
            case 0: flgZ=1;
         endswitch
   endswitch
endif

if (chgflgZ9p)
   switch(regA.4:7)
      case 0:
         switch(reg0T.0:3)
            case 0: flgZ=1;
         endswitch
   endswitch
endif

if ( chgflgZ1p
   | chgflgZ2p
   | chgflgZ3p
   | chgflgZ4p
   | chgflgZ5p
   | chgflgZ6p
   | chgflgZ7p
   | chgflgZ8p
   | chgflgZ9p
   | chgflgZ10p
   | chgflgZ11p
   )
else
   if (RESET)
      flgZ=0;
   else
      flgZ=flgZ;
   endif
endif
```

```
{ ------------------------------------------------------------------- }
{    フラグ S                                                }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.33&anct2p) { POP AF }
            switch(code0d.4:5)
               case 3: chgflgS1p=1;
            endswitch
         endif

         if (instp.33&anct1p) chgflgS1p=1; endif { LD A,R }
         if (instp.39&anct1p) chgflgS2p=1; endif { EX  AF,AF' }

         if (anct1p)
            if ( instp.16  { LD A,I }
               | instp.18  { LD A,R }
               | instp.154 { RLD }
               | instp.155 { RRD }
               )
               chgflgS3p=1;
            endif
         endif

         if (instp.100&anct1p) chgflgS4p=1; endif { DAA }

         if (anct1p)
            if ( instp.112 { ADC HL,ss }
               | instp.113 { SBC HL,ss }
               )
               chgflgS5p=1;
            endif
         endif

         if (anct1p)
            if ( instp.156 { BIT b,r }
               | instp.157 { BIT b,(HL) }
               | instp.158 { BIT b,(IX+d) }
               | instp.159 { BIT b,(IY+d) }
               | instp.190 { INI }
               | instp.191 { IND }
               | instp.192 { INIR }
               | instp.193 { INDR }
               | instp.194 { OUTI }
               | instp.195 { OUTD }
               | instp.196 { OTIR }
               | instp.197 { OTIR }
               )
               chgflgS6p=1;
            endif
         endif

         if (anct1p)
            if ( instp.52 { ADD A,r }
               | instp.53 { ADD A,n }
               | instp.54 { ADD A,(HL) }
               | instp.55 { ADD A,(IX+d) }
               | instp.56 { ADD A,(IY+d) }
               | instp.57 { ADC A,r }
               | instp.58 { ADC A,n }
               | instp.59 { ADC A,(HL) }
               | instp.60 { ADC A,(IX+d) }
```

```
| instp.61 { ADC A,(IY+d) }
| instp.62 { SUB r }
| instp.63 { SUB n }
| instp.64 { SUB (HL) }
| instp.65 { SUB (IX+d) }
| instp.66 { SUB (IY+d) }
| instp.67 { SBC A,r }
| instp.68 { SBC A,n }
| instp.69 { SBC A,(HL) }
| instp.70 { SBC A,(IX+d) }
| instp.71 { SBC A,(IY+d) }
| instp.72 { AND r }
| instp.73 { AND n }
| instp.74 { AND (HL) }
| instp.75 { AND (IX[IY]+d) }
| instp.77 { OR r }
| instp.78 { OR n }
| instp.79 { OR (HL) }
| instp.80 { OR (IX+d) }
| instp.81 { OR (IY+d) }
| instp.82 { XOR r }
| instp.83 { XOR n }
| instp.84 { XOR (HL) }
| instp.85 { XOR (IX+d) }
| instp.86 { XOR (IY+d) }
| instp.87 { CP r }
| instp.88 { CP n }
| instp.89 { CP (HL) }
| instp.90 { CP (IX+d) }
| instp.91 { CP (IY+d) }
| instp.92 { INC r }
| instp.93 { INC (HL) }
| instp.94 { INC (IX+d) }
| instp.95 { INC (IY+d) }
| instp.96 { DEC r }
| instp.97 { DEC (HL) }
| instp.98 { DEC (IX+d) }
| instp.99 { DEC (IY+d) }
| instp.102 { NEG }
| instp.126 { RLC r }
| instp.127 { RLC (HL) }
| instp.128 { RLC (IX+d) }
| instp.129 { RLC (IY+d) }
| instp.130 { RL r }
| instp.131 { RL (HL) }
| instp.132 { RL (IX+d) }
| instp.133 { RL (IY+d) }
| instp.134 { RRC r }
| instp.135 { RRC (HL) }
| instp.136 { RRC (IX+d) }
| instp.137 { RRC (IY+d) }
| instp.138 { RR r }
| instp.139 { RR (HL) }
| instp.140 { RR (IX+d) }
| instp.141 { RR (IY+d) }
| instp.142 { SLA r }
| instp.143 { SLA (HL) }
| instp.144 { SLA (IX+d) }
| instp.145 { SLA (IY+d) }
| instp.146 { SRL r }
```

```
               | instp.147 { SRL (HL) }
               | instp.148 { SRL (IX+d) }
               | instp.149 { SRL (IY+d) }
               | instp.150 { SRA r }
               | instp.151 { SRA (HL) }
               | instp.152 { SRA (IX+d) }
               | instp.153 { SRA (IY+d) }
               | instp.188 { IN r,(C) }
               )
               chgflgS7p=1;
            endif
         endif
      endif
   endif

   if (chgflgS1p) flgS=DATAIN.7; endif
   if (chgflgS2p) flgS=flgRS; endif
   if (chgflgS3p) flgS=regA.7; endif
   if (chgflgS4p) flgS=daahA.3; endif
   if (chgflgS5p) flgS=alu16out.15; endif
   if (chgflgS6p) flgS=0; endif
   if (chgflgS7p) flgS=alu8out.7; endif

   if ( chgflgS1p
      | chgflgS2p
      | chgflgS3p
      | chgflgS4p
      | chgflgS5p
      | chgflgS6p
      | chgflgS7p
      )
   else
      if (RESET)
         flgS=0;
      else
         flgS=flgS;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    フラグ C                                                 }
{ ------------------------------------------------------------------------ }
   if (!RESET)
      if (!WAIT)
         if (instp.33&anct2p) { POP AF }
            switch(code0d.4:5)
               case 3: chgflgC12p=1;
            endswitch
         endif

         if (instp.39&anct1p)  chgflgC13p=1; endif { EX  AF,AF' }
         if (instp.100&anct1p) chgflgC3p=1; endif  { DAA }
         if (instp.103&anct1p) chgflgC4p=1; endif  { CCF }
         if (instp.104&anct1p) chgflgC5p=1; endif  { SCF }

         if (anct1p)
            if ( instp.122    { RLCA }
               | instp.123    { RLA }
               )
               chgflgC6p=1;
            endif
         endif

         if (anct1p)
```

```
   if ( instp.124       { RRCA }
      | instp.125       { RRA }
      )
      chgflgC7p=1;
   endif
endif

if (instp.126&anct1p) chgflgC2p=1; endif { RLC r }

if (anct1p)
   if ( instp.113  { SBC HL,ss }
      | instp.111  { ADD HL,ss }
      | instp.112  { ADC HL,ss }
      | instp.114  { ADD IX,pp }
      | instp.115  { ADD IY,rr }
      )
      chgflgC8p=1;
   endif
endif

if (anct1p)
   if ( instp.52  { ADD A,r }
      | instp.53  { ADD A,n }
      | instp.54  { ADD A,(HL) }
      | instp.55  { ADD A,(IX+d) }
      | instp.56  { ADD A,(IY+d) }
      | instp.57  { ADC A,r }
      | instp.58  { ADC A,n }
      | instp.59  { ADC A,(HL) }
      | instp.60  { ADC A,(IX+d) }
      | instp.61  { ADC A,(IY+d) }
      | instp.127 { RLC (HL) }
      | instp.128 { RLC (IX+d) }
      | instp.129 { RLC (IY+d) }
      | instp.130 { RL r }
      | instp.131 { RL (HL) }
      | instp.132 { RL (IX+d) }
      | instp.133 { RL (IY+d) }
      | instp.134 { RRC r }
      | instp.135 { RRC (HL) }
      | instp.136 { RRC (IX+d) }
      | instp.137 { RRC (IY+d) }
      | instp.138 { RR r }
      | instp.139 { RR (HL) }
      | instp.140 { RR (IX+d) }
      | instp.141 { RR (IY+d) }
      | instp.142 { SLA r }
      | instp.143 { SLA (HL) }
      | instp.144 { SLA (IX+d) }
      | instp.145 { SLA (IY+d) }
      | instp.146 { SRL r }
      | instp.147 { SRL (HL) }
      | instp.148 { SRL (IX+d) }
      | instp.149 { SRL (IY+d) }
      | instp.150 { SRA r }
      | instp.151 { SRA (HL) }
      | instp.152 { SRA (IX+d) }
      | instp.153 { SRA (IY+d) }
      | instp.102 { NEG }
      | instp.62  { SUB r }
      | instp.63  { SUB n }
```

```
            | instp.64  { SUB (HL) }
            | instp.65  { SUB (IX+d) }
            | instp.66  { SUB (IY+d) }
            | instp.87  { CP r }
            | instp.88  { CP n }
            | instp.89  { CP (HL) }
            | instp.90  { CP (IX+d) }
            | instp.91  { CP (IY+d) }
            | instp.67  { SBC A,r }
            | instp.68  { SBC A,n }
            | instp.69  { SBC A,(HL) }
            | instp.70  { SBC A,(IX+d) }
            | instp.71  { SBC A,(IY+d) }
            )
            chgflgC9p=1;
         endif
      endif

      if (anct1p)
         if ( instp.72          { AND r }
            | instp.73          { AND n }
            | instp.74          { AND (HL) }
            | instp.75          { AND (IX[IY]+d) }
            | instp.77          { OR r }
            | instp.78          { OR n }
            | instp.79          { OR (HL) }
            | instp.80          { OR (IX+d) }
            | instp.81          { OR (IY+d) }
            | instp.82          { XOR r }
            | instp.83          { XOR n }
            | instp.84          { XOR (HL) }
            | instp.85          { XOR (IX+d) }
            | instp.86          { XOR (IY+d) }
            | instp.190         { INI }
            | instp.191         { IND }
            | instp.192         { INIR }
            | instp.193         { INDR }
            | instp.194         { OUTI }
            | instp.195         { OUTD }
            | instp.196         { OTIR }
            | instp.197         { OTDR }
            )
            chgflgC11p=1;
         endif
      endif

   endif
endif

if (chgflgC2p)
   switch(code0d.0:2)
      case 0: flgC=regB.7;
      case 1: flgC=regC.7;
      case 2: flgC=regD.7;
      case 3: flgC=regE.7;
      case 4: flgC=regH.7;
      case 5: flgC=regL.7;
      case 7: flgC=regA.7;
   endswitch
endif

if (chgflgC3p) flgC=daacarry; endif
if (chgflgC4p) flgC=!flgC; endif
```

```
   if (chgflgC5p) flgC=1; endif
   if (chgflgC6p) flgC=regA.7; endif
   if (chgflgC7p) flgC=regA.0; endif
   if (chgflgC8p) flgC=alu16out.16; endif
   if (chgflgC9p) flgC=alu8out.8; endif
   if (chgflgC11p) flgC=0; endif
   if (chgflgC12p) flgC=DATAIN.0; endif
   if (chgflgC13p) flgC=flgRC; endif

   if ( chgflgC2p
      | chgflgC3p
      | chgflgC4p
      | chgflgC5p
      | chgflgC6p
      | chgflgC7p
      | chgflgC8p
      | chgflgC9p
      | chgflgC11p
      | chgflgC12p
      | chgflgC13p
      )
   else
      if (RESET)
         flgC=0;
      else
         flgC=flgC;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    補助レジスタ T0                                          }
{ ------------------------------------------------------------------------ }
   if (!RESET)
      if (!WAIT)
         if (nmiset.0|intset.0)      { 割り込み }
            if (instp.105)
               reg0T=regPC1p.0:7;    { HALT に割り込み }
            else
               if (anct0p)
                  reg0T=regPC.0:7;
               else
                  reg0T=reg0T;
               endif
            endif
         else
            if (anct4p)
               if ( instp.179    { CALL n'n }
                  | instp.180    { CALL cc,n'n }
                  )
                  chgreg0T2p=1;
               endif
            endif

            if (anct3p)
               if ( instp.41    { EX (SP),HL }
                  | instp.42    { EX (SP),IX }
                  | instp.43    { EX (SP),IY }
                  )
                  chgreg0T1p=1;
               endif
            endif

            if (anct2p)
               if ( instp.44    { LDI }
                  | instp.45    { LDD }
```

```
                | instp.46   { LDIR }
                | instp.47   { LDDR }
                | instp.93   { INC (HL) }
                | instp.94   { INC (IX+d) }
                | instp.95   { INC (IY+d) }
                | instp.97   { DEC (HL) }
                | instp.98   { DEC (IX+d) }
                | instp.99   { DEC (IY+d) }
                | instp.127   { RLC (HL) }
                | instp.128   { RLC (IX+d) }
                | instp.129   { RLC (IY+d) }
                | instp.131   { RL (HL) }
                | instp.132   { RL (IX+d) }
                | instp.133   { RL (IY+d) }
                | instp.135   { RRC (HL) }
                | instp.136   { RRC (IX+d) }
                | instp.137   { RRC (IY+d) }
                | instp.139   { RR (HL) }
                | instp.140   { RR (IX+d) }
                | instp.141   { RR (IY+d) }
                | instp.143   { SLA (HL) }
                | instp.144   { SLA (IX+d) }
                | instp.145   { SLA (IY+d) }
                | instp.147   { SLA (HL) }
                | instp.148   { SRL (IX+d) }
                | instp.149   { SRL (IY+d) }
                | instp.151   { SRA (HL) }
                | instp.152   { SRA (IX+d) }
                | instp.153   { SRA (IY+d) }
                | instp.154   { RLD }
                | instp.155   { RRD }
                | instp.161   { SET b,(HL) }
                | instp.162   { SET b,(IX+d) }
                | instp.163   { SET b,(IY+d) }
                | instp.165   { RES b,(HL) }
                | instp.166   { RES b,(IX+d) }
                | instp.167   { RES b,(IY+d) }
                | instp.181   { RET }
                | instp.182   { RET cc }
                | instp.183   { RETI }
                | instp.184   { RETN }
                | instp.190   { INI }
                | instp.191   { IND }
                | instp.192   { INIR }
                | instp.193   { INDR }
                | instp.194   { OUTI }
                | instp.195   { OUTD }
                | instp.196   { OTIR }
                | instp.197   { OTDR }
               )
               chgreg0T1p=1;
            endif
         endif
      endif
   endif
endif

if (chgreg0T1p) reg0T=DATAIN; endif
if (chgreg0T2p) reg0T=regPC2p.0:7; endif

if ( nmiset.0
```

```
        | intset.0
        | chgreg0T1p
        | chgreg0T2p
        )
    else
        if (RESET)
            reg0T=0;
        else
            reg0T=reg0T;
        endif
    endif

{ ------------------------------------------------------------------- }
{     補助レジスタ T1                                                 }
{ ------------------------------------------------------------------- }
    if (!RESET)
        if (!WAIT)
            if (nmiset.0|intset.0)          { 割り込み }
                if (instp.105)
                    reg1T=regPC1p.8:15;     { HALT に割り込み }
                else
                    if (anct0p)
                        reg1T=regPC.8:15;
                    else
                        reg1T=reg1T;
                    endif
                endif
            else
                if (anct4p)
                    if ( instp.179    { CALL n'n }
                        | instp.180   { CALL cc,n'n }
                        )
                        chgreg1T2p=1;
                    endif
                endif

                if (anct4p)
                    if ( instp.41
                        | instp.42
                        | instp.43
                        )
                        chgreg1T1p=1;
                    endif
                endif

                { RLC (IX+d) }
                { RL (IX+d) }
                { RL (IY+d) }
                { RRC (IX+d) }
                { RRC (IY+d) }
                { RR (IX+d) }
                { RR (IY+d) }
                { SLA (IX+d) }
                { SLA (IY+d) }
                { SRL (IX+d) }
                { SRL (IY+d) }
                { SRA (IX+d) }
                { BIT b,(IX+d) }
                { BIT b,(IY+d) }
                { SET b,(IX+d) }
                { SET b,(IY+d) }
                { RES b,(IX+d) }
                { RES b,(IY+d) }
```

```
            if (anct11p)
               if (instp.0) chgreg1T1p=1; endif
            endif
         endif
      endif
   endif

   if (chgreg1T1p) reg1T=DATAIN; endif
   if (chgreg1T2p) reg1T=regPC2p.8:15; endif

   if ( nmiset.0
      | intset.0
      | chgreg1T1p
      |chgreg1T2p
      )
   else
      if (RESET)
         reg1T=0;
      else
         reg1T=reg1T;
      endif
   endif

{ ------------------------------------------------------------------- }
{    進み PC                                                          }
{ ------------------------------------------------------------------- }
   regPC1p=regPC+1;

{ ------------------------------------------------------------------- }
{    戻り PC                                                          }
{ ------------------------------------------------------------------- }
   regPC2p=regPC+2;

{ ------------------------------------------------------------------- }
{    レジスタ R                                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
      regR=0;
   else
      if (!WAIT)
         switch(expinst)
            case 3:
               if (anct1p)
                  if (instp.19)   { LD R,A }
                     regR=regA;
                  else
                     regR=regR;
                  endif
               else
                  regR=regR;
               endif
            default:
               regR=regR;
         endswitch
      else
         regR=regR;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ I                                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
      regI=0;
   else
```

```
      if (!WAIT)
         switch(expinst)
            case 3:
               if (anct1p)
                  if (instp.17)   { LD A,I }
                     regI=regA;
                  else
                     regI=regI;
                  endif
               else
                  regI=regI;
               endif
            default:
            regI=regI;
         endswitch
      else
         regI=regI;
      endif
   endif

{ ------------------------------------------------------------------------ }
{    レジスタ IX                                          }
{ ------------------------------------------------------------------------ }
   if (!RESET)
      if (!WAIT)
         switch(expinst)
            case 1:
               if (instp.5&anct1p) chgregIX7p=1; endif   { LD IX,n'n }

               if (anct1p)
                  if ( instp.25   { LD IX,(n'n) }
                     | instp.36   { POP IX }
                     )
                     chgregIX1p=1;
                  endif
               endif

               if (anct2p)
                  if ( instp.25   { LD IX,(n'n) }
                     | instp.36   { POP IX }
                     )
                     chgregIX2p=1;
                  endif
               endif

               if (anct1p)
                  if (instp.42) chgregIX3p=1; endif   { EX (SP),IX }
               endif

               if (anct2p)
                  if (instp.42) chgregIX4p=1; endif   { EX (SP),IX }
               endif

               if (instp.114&anct1p) chgregIX5p=1; endif   { ADD IX,pp }

               if (anct1p)
                  if ( instp.117   { INC IX }
                     | instp.120   { DEC IX }
                     )
                     chgregIX6p=1;
                  endif
               endif
         endswitch
      endif
   endif
```

```
   if (chgregIX1p)
      regIX.8:15=DATAIN;
      regIX.0:7=regIX.0:7;
   endif

   if (chgregIX2p)
      regIX.0:7=DATAIN;
      regIX.8:15=regIX.8:15;
   endif

   if (chgregIX3p)
      regIX.0:7=reg0T;
      regIX.8:15=regIX.8:15;
   endif

   if (chgregIX4p)
      regIX.0:7=regIX.0:7;
      regIX.8:15=reg1T;
   endif

   if (chgregIX5p)
      regIX.0:15=alu16out.0:15;
   endif

   if (chgregIX6p)
      regIX.0:15=alu16out.0:15;
    endif

   if (chgregIX7p)
      regIX.0:7=code2d;
      regIX.8:15=code1d;
   endif

   if ( chgregIX1p
      | chgregIX2p
      | chgregIX3p
      | chgregIX4p
      | chgregIX5p
      | chgregIX6p
      | chgregIX7p
      )
   else
      if (RESET)
         regIX=0;
      else
         regIX=regIX;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ IY                                          }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         switch(expinst)
            case 2:
               if (instp.5&anct1p) chgregIY1p=1; endif   { LD IY,n'n }
               if (instp.27&anct1p) chgregIY2p=1; endif  { LD IY,(n'n) }

               if (anct2p)
                  if ( instp.27   { LD IY,(n'n) }
                     | instp.37   { POP IY }
                     )
                     chgregIY3p=1;
```

```
               endif
            endif

            if (instp.37&anct1p) chgregIY4p=1; endif   { POP IY }

            if (anct1p)
               if (instp.43) chgregIY5p=1; endif   { EX (SP),IY }
            endif

            if (anct2p)
               if (instp.43) chgregIY6p=1; endif   { EX (SP),IY }
            endif

            if (instp.115&anct1p) chgregIY7p=1; endif   { ADD IY,rr }

            if (anct1p)
               if ( instp.118   { INC IY }
                  | instp.121   { DEC IY }
                  )
                  chgregIY7p=1;
               endif
            endif
      endswitch
   endif
endif

if (chgregIY1p)
   regIY.0:7=code2d;
   regIY.8:15=code1d;
endif

if (chgregIY2p)
   regIY.0:7=regIY.0:7;
   regIY.8:15=DATAIN;
endif

if (chgregIY3p)
   regIY.0:7=DATAIN;
   regIY.8:15=regIY.8:15;
endif

if (chgregIY4p)
   regIY.8:15=DATAIN;
   regIY.0:7=regIY.0:7;
endif

if (chgregIY5p)
   regIY.0:7=reg0T;
   regIY.8:15=regIY.8:15;
endif

if (chgregIY6p)
   regIY.0:7=regIY.0:7;
   regIY.8:15=reg1T;
endif

if (chgregIY7p) regIY.0:15=alu16out.0:15; endif

if ( chgregIY1p
   | chgregIY2p
   | chgregIY3p
   | chgregIY4p
   | chgregIY5p
   | chgregIY6p
   | chgregIY7p
   )
```

```
   else
      if (RESET)
         regIY=0;
      else
         regIY=regIY;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助フラグ EI                                          }
{ ------------------------------------------------------------------- }
   if (RESET)
      flgREI=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)    { EX AF,AF' }
            flgREI=flgEI;
         else
            flgREI=flgREI;
         endif
      else
         flgREI=flgREI;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助フラグ N                                           }
{ ------------------------------------------------------------------- }
   if (RESET)
      flgRN=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)    { EX AF,AF' }
            flgRN=flgN;
         else
            flgRN=flgRN;
         endif
      else
         flgRN=flgRN;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助フラグ PV                                          }
{ ------------------------------------------------------------------- }
   if (RESET)
      flgRPV=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)    { EX AF,AF' }
            flgRPV=flgPV;
         else
            flgRPV=flgRPV;
         endif
      else
         flgRPV=flgRPV;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助フラグ H                                           }
{ ------------------------------------------------------------------- }
   if (RESET)
      flgRH=0;
   else
```

```
      if (!WAIT)
         if (instp.39&anct1p)   { EX AF,AF' }
            flgRH=flgH;
         else
            flgRH=flgRH;
         endif
      else
         flgRH=flgRH;
      endif
   endif

{ ------------------------------------------------------------------ }
{    補助フラグ Z                                          }
{ ------------------------------------------------------------------ }
   if (RESET)
      flgRZ=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)   { EX AF,AF' }
            flgRZ=flgZ;
         else
            flgRZ=flgRZ;
         endif
      else
         flgRZ=flgRZ;
      endif
   endif

{ ------------------------------------------------------------------ }
{    補助フラグ S                                          }
{ ------------------------------------------------------------------ }
   if (RESET)
      flgRS=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)   { EX AF,AF' }
            flgRS=flgS;
         else
            flgRS=flgRS;
         endif
      else
         flgRS=flgRS;
      endif
   endif

{ ------------------------------------------------------------------ }
{    補助フラグ C                                          }
{ ------------------------------------------------------------------ }
   if (RESET)
      flgRC=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)   { EX AF,AF' }
            flgRC=flgC;
         else
            flgRC=flgRC;
         endif
      else
         flgRC=flgRC;
      endif
   endif

{ ------------------------------------------------------------------ }
{    補助レジスタ A                                         }
{ ------------------------------------------------------------------ }
```

```
   if (RESET)
      regRA=0;
   else
      if (!WAIT)
         if (instp.39&anct1p)    { EX AF,AF' }
            regRA=regA;
         else
            regRA=regRA;
         endif
      else
         regRA=regRA;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助レジスタ B                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRB=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRB=regB;
         else
            regRB=regRB;
         endif
      else
         regRB=regRB;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助レジスタ C                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRC=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRC=regC;
         else
            regRC=regRC;
         endif
      else
         regRC=regRC;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助レジスタ D                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRD=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRD=regD;
         else
            regRD=regRD;
         endif
      else
         regRD=regRD;
      endif
   endif
```

```
{ ------------------------------------------------------------------- }
{    補助レジスタ E                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRE=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRE=regE;
         else
            regRE=regRE;
         endif
      else
         regRE=regRE;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助レジスタ H                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRH=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRH=regH;
         else
            regRH=regRH;
         endif
      else
         regRH=regRH;
      endif
   endif

{ ------------------------------------------------------------------- }
{    補助レジスタ L                                                   }
{ ------------------------------------------------------------------- }
   if (RESET)
      regRL=0;
   else
      if (!WAIT)
         if (instp.40&anct1p)    { EXX }
            regRL=regL;
         else
            regRL=regRL;
         endif
      else
         regRL=regRL;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ A                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (anct1p)
            if ( instp.52     { ADD A,r }
               | instp.53     { ADD A,n }
               | instp.54     { ADD A,(HL) }
               | instp.55     { ADD A,(IX+d) }
               | instp.56     { ADD A,(IY+d) }
               | instp.57     { ADC A,r }
               | instp.58     { ADC A,n }
```

```
      | instp.59     { ADC A,(HL) }
      | instp.60     { ADC A,(IX+d) }
      | instp.61     { ADC A,(IY+d) }
      | instp.62     { SUB r }
      | instp.63     { SUB n }
      | instp.64     { SUB (HL) }
      | instp.65     { SUB (IX+d) }
      | instp.66     { SUB (IY+d) }
      | instp.67     { SBC A,r }
      | instp.68     { SBC A,n }
      | instp.69     { SBC A,(HL) }
      | instp.70     { SBC A,(IX+d) }
      | instp.71     { SBC A,(IY+d) }
      | instp.72     { AND r }
      | instp.73     { AND n }
      | instp.74     { AND (HL) }
      | instp.75     { AND (IX[IY]+d) }
      | instp.77     { OR r }
      | instp.78     { OR n }
      | instp.79     { OR (HL) }
      | instp.80     { OR (IX+d) }
      | instp.81     { OR (IY+d) }
      | instp.82     { XOR r }
      | instp.83     { XOR n }
      | instp.84     { XOR (HL) }
      | instp.85     { XOR (IX+d) }
      | instp.86     { XOR (IY+d) }
      | instp.101    { CPL }
      | instp.102    { NEG }
      )
      chgregA1p=1;
   endif
endif

if (anct1p)
   if ( instp.92     { INC A }
      | instp.96     { DEC A }
      | instp.188    { IN r,(C) }
      )
      chgregA2p=1;
   endif
endif

if (instp.100&anct1p) chgregA3p=1; endif   { DAA }

if (anct1p)
   if ( instp.122    { RLCA }
      | instp.123    { RLA }
      | instp.124    { RRCA }
      | instp.125    { RRA }
      )
      chgregA4p=1;
   endif
endif

if (anct1p)
   if ( instp.126    { RLC A }
      | instp.130    { RL A }
      | instp.134    { RRC A }
      | instp.138    { RR A }
      | instp.142    { SLA A }
      | instp.146    { SRL A }
```

```
            | instp.150    { SRA A }
            | instp.160    { SET b,A }
            | instp.164    { RES b,A }
            )
            chgregA5p=1;
         endif
      endif

      if (instp.186&anct1p) chgregA15p=1; endif { IN A,(n) }
      if (instp.1&anct1p) chgregA6p=1; endif    { LD A,n }

      { LD A,B }
      { LD A,C }
      { LD A,D }
      { LD A,E }
      { LD A,H }
      { LD A,L }
      { LD A,A }
      if (instp.2&anct1p) chgregA7p=1; endif

      if (anct1p)
         if ( instp.3    { LD A,(HL) }
            | instp.7    { LD A,(IX[IY]+d) }
            | instp.188  { IN r,(C) }
            )
            chgregA9p=1;
         endif
      endif

      if (instp.16&anct1p) chgregA10p=1; endif  { LD A,I }
      if (instp.18&anct1p) chgregA11p=1; endif  { LD A,R }
      if (instp.39&anct1p) chgregA12p=1; endif  { EX AF,AF' }
      if (instp.154&anct1p) chgregA13p=1; endif { RLD }

      if (instp.155&anct1p) chgregA14p=1; endif { RRD }

      if (anct1p)
         if ( instp.10   { LD A,(BC) }
            | instp.12   { LD A,(DE) }
            | instp.14   { LD A,(n'n) }
            )
            chgregA15p=1;
         endif
      endif

      if (anct1p)
         if (instp.33)   { POP AF }
            switch(code0d.4:5)
               case 3: chgregA15p=1;
            endswitch
         endif
      endif

   endif
endif

if (chgregA1p) regA=alu8out.0:7; endif

if (chgregA2p)
   switch(code0d.3:5)
      case 7:  regA=alu8out.0:7;
      default: regA=regA;
   endswitch
endif
```

```
if (chgregA3p)
   regA.0:3=daalA.0:3;
   regA.4:7=daahA.0:3;
endif

if (chgregA4p) regA=alu8out.0:7; endif

if (chgregA5p)
   switch(code0d.0:2)
      case 7:  regA=alu8out.0:7;
      default: regA=regA;
   endswitch
endif

if (chgregA6p)
   if (code0d.3:5==0b111)
      regA=code1d;
   else
      regA=regA;
   endif
 endif

if (chgregA7p)
   if (code0d.3:5==0b111)
      switch(code0d.0:2)
         case 0: regA=regB;
         case 1: regA=regC;
         case 2: regA=regD;
         case 3: regA=regE;
         case 4: regA=regH;
         case 5: regA=regL;
         case 7: regA=regA;
      endswitch
   else
      regA=regA;
   endif
endif

if (chgregA9p)
   switch(code0d.3:5)
      case 7: regA=DATAIN;
      default: regA=regA;
   endswitch
endif

if (chgregA10p) regA=regI; endif
if (chgregA11p) regA=regR; endif
if (chgregA12p) regA=regRA; endif

if (chgregA13p)
   regA.0:3=reg0T.4:7;
   regA.4:7=regA.4:7;
endif

if (chgregA14p)
   regA.0:3=reg0T.0:3;
   regA.4:7=regA.4:7;
endif

if (chgregA15p) regA=DATAIN; endif

if ( chgregA1p
   | chgregA2p
   | chgregA3p
   | chgregA4p
```

```
      | chgregA5p
      | chgregA6p
      | chgregA7p
      | chgregA9p
      | chgregA10p
      | chgregA11p
      | chgregA12p
      | chgregA13p
      | chgregA14p
      | chgregA15p
      )
   else
      if (RESET)
         regA=0;
      else
         regA=regA;
      endif
   endif

{ ------------------------------------------------------------------- }
{    10 進補正                                                       }
{ ------------------------------------------------------------------- }
   daalin.0:3=regA.0:3;
   daahin.0:3=regA.4:7;
   ab10lin=daalin>9;
   ab10hin=daahin>9;

   if (flgC)
      daacarry=1;
   else
      if (daahA.4)
         daacarry=1;
      else
         if (ab10hin) daacarry=1; endif
      endif
   endif

   if (flgN)                          { 減算 }
      switch(flgH,ab10lin)
         case 0,0: daalA=daalin;
         case 0,1: daalA=daalin-10;
         case 1,0:
            switch(daalin)
               case 0:  daalA=0;
               case 1:  daalA=1;
               case 2:  daalA=2;
               case 3:  daalA=3;
               case 4:  daalA=4;
               case 5:  daalA=5;
               case 6:  daalA=0;
               case 7:  daalA=1;
               case 8:  daalA=2;
               case 9:  daalA=3;
            endswitch
         case 1,1: daalA=daalin-6;
      endswitch
   else                               { 加算 }
      switch(flgH,ab10lin)
         case 0,0: daalA=daalin;
         case 0,1: daalA=daalin+0b0110; { -10 は 5 ビット目に桁上げをするために 0b0110 にする }
         case 1,0: daalA=daalin+6;
         case 1,1:                    { 無効 }
      endswitch
   endif

   if (flgN)                             { 減算 }
```

```
      switch(flgC,ab10hin)
         case 0,0: daahA=daahin;
         case 0,1: daahA=daahin-10;
         case 1,0:
            switch(daahin)
               case 0:  daahA=0;
               case 1:  daahA=1;
               case 2:  daahA=2;
               case 3:  daahA=3;
               case 4:  daahA=4;
               case 5:  daahA=5;
               case 6:  daahA=0;
               case 7:  daahA=1;
               case 8:  daahA=2;
               case 9:  daahA=3;
            endswitch
         case 1,1: daahA=daahin-6;
      endswitch
   else                                  { 加算 }
      switch(flgC,ab10hin,ab10lin)
         case 0,0,0: daahA=daahin;
         case 0,0,1: daahA=daahin+1;
         case 0,1,0: daahA=daahin-10;
         case 0,1,1: daahA=daahin-9;
         case 1,0,0: daahA=daahin+6;
         case 1,0,1: daahA=daahin+7;
         case 1,1,0:                   { 無効 }
         case 1,1,1:                   { 無効 }
      endswitch
   endif

{ ------------------------------------------------------------------- }
{    レジスタ B                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.1&anct1p) chgregB6p=1; endif   { LD B,n }
         if (instp.2&anct1p) chgregB7p=1; endif   { LD B,r }

         if (anct1p)
            if ( instp.3   { LD B,(HL) }
               | instp.7   { LD B,(IX[IY]+d) }
               | instp.188 { IN B,(C) }
               )
               chgregB9p=1;
            endif
         endif

         if (instp.20&anct1p) chgregB10p=1; endif { LD BC,n'n }

         if (anct1p)
            if (instp.23)    { LD BC,(n'n) }
               chgregB12p=1;
            endif
         endif

         if (anct1p)
            if (instp.33)    { POP BC }
               switch(code0d.4:5)
                  case 0: chgregB12p=1;
               endswitch
            endif
         endif

         if (instp.40&anct1p) chgregB13p=1; endif { EXX }
```

```
        if (anct1p)
           if ( instp.44  { LDI }
              | instp.45  { LDD }
              | instp.46  { LDIR }
              | instp.47  { LDDR }
              | instp.48  { CPI }
              | instp.49  { CPD }
              | instp.50  { CPIR }
              | instp.51  { CPDR }
              )
              chgregB1p=1;
           endif
        endif

        if (anct1p)
           if ( instp.92  { INC B }
              | instp.96  { DEC B }
              )
              chgregB2p=1;
           endif
        endif

        if (anct1p)
           if ( instp.116 { INC BC }
              | instp.119 { DEC BC }
              )
              chgregB3p=1;
           endif
        endif

        if (anct1p)
           if ( instp.126 { RLC B }
              | instp.130 { RL B }
              | instp.134 { RRC B }
              | instp.138 { RR B }
              | instp.142 { SLA B }
              | instp.146 { SRL B }
              | instp.150 { SRA B }
              | instp.160 { SET b,B }
              | instp.164 { RES b,B }
              )
              chgregB4p=1;
           endif
        endif

        if (anct1p)
           if ( instp.178 { DJNZ e }
              | instp.190 { INI }
              | instp.191 { IND }
              | instp.192 { INIR }
              | instp.193 { INDR }
              | instp.194 { OUTI }
              | instp.195 { OUTD }
              | instp.196 { OTIR }
              | instp.197 { OTDR }
              )
              chgregB5p=1;
           endif
        endif
     endif
  endif
```

```
if (chgregB1p)
   switch(regC)
      case 0:  regB=regB-1;
      default: regB=regB;
   endswitch
endif

if (chgregB2p)
   switch(code0d.3:5)
      case 0:  regB=alu8out.0:7;
      default: regB=regB;
   endswitch
endif

if (chgregB3p)
   switch(code0d.4:5)
      case 0:  regB=alu16out.8:15;
      default: regB=regB;
   endswitch
endif

if (chgregB4p)
   switch(code0d.0:2)
      case 0:  regB=alu8out.0:7;
      default: regB=regB;
   endswitch
endif

if (chgregB5p) regB=regB1m; endif

if (chgregB6p)
   if (code0d.3:5==0b000)
      regB=code1d;
   else
      regB=regB;
   endif
endif

if (chgregB7p)
   if (code0d.3:5==0b000)
      switch(code0d.0:2)
         case 0: regB=regB;
         case 1: regB=regC;
         case 2: regB=regD;
         case 3: regB=regE;
         case 4: regB=regH;
         case 5: regB=regL;
         case 7: regB=regA;
      endswitch
   else
      regB=regB;
   endif
endif

if (chgregB9p)
   switch(code0d.3:5)
      case 0: regB=DATAIN;
      default:regB=regB;
   endswitch
endif

if (chgregB10p)
   switch(code0d.4:5)
      case 0: regB=code1d;
      default:regB=regB;
   endswitch
```

```
   endif

   if (chgregB12p)
      switch(code0d.4:5)
         case 0:
            regB=DATAIN;
         default:regB=regB;
      endswitch
   endif

   if (chgregB13p) regB=regRB; endif

   if ( chgregB1p
      | chgregB2p
      | chgregB3p
      | chgregB4p
      | chgregB5p
      | chgregB6p
      | chgregB7p
      | chgregB9p
      | chgregB10p
      | chgregB12p
      | chgregB13p
      )
   else
      if (RESET)
         regB=0;
      else
         regB=regB;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ C                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.1&anct1p) chgregC5p=1; endif   { LD C,n }
         if (instp.2&anct1p) chgregC6p=1; endif   { LD C,r }

         if (anct1p)
            if ( instp.3   { LD C,(HL) }
               | instp.7   { LD C,(IX[IY]+d) }
               | instp.188 { IN C,(C) }
               )
               chgregC8p=1;
            endif
         endif

         if (instp.20&anct1p) chgregC9p=1; endif  { LD BC,n'n }
         if (instp.23&anct2p) chgregC10p=1; endif { LD BC,(n'n) }

         if (instp.33&anct2p) { POP BC }
            switch(code0d.4:5)
               case 0: chgregC11p=1;
            endswitch
         endif

         if (instp.40&anct1p) chgregC12p=1; endif { EXX }

         if (anct1p)
            if ( instp.44     { LDI }
               | instp.45     { LDD }
               | instp.46     { LDIR }
               | instp.47     { LDDR }
```

```
            | instp.48     { CPI }
            | instp.49     { CPD }
            | instp.50     { CPIR }
            | instp.51     { CPDR }
            )
            chgregC1p=1;
         endif
      endif

      if (anct1p)
         if ( instp.92     { INC C }
            | instp.96     { DEC C }
            )
            chgregC2p=1;
         endif
      endif

      if (anct1p)
         if ( instp.116    { INC BC }
            | instp.119    { DEC BC }
            )
            chgregC3p=1;
         endif
      endif

      if (anct1p)
         if ( instp.126    { RLC C }
            | instp.130    { RL C }
            | instp.134    { RRC C }
            | instp.138    { RR C }
            | instp.142    { SLA C }
            | instp.146    { SRL C }
            | instp.150    { SRA C }
            | instp.160    { SET b,C }
            | instp.164    { RES b,C }
            )
            chgregC4p=1;
         endif
      endif
   endif
endif

if (chgregC1p) regC=regC-1; endif

if (chgregC2p)
   switch(code0d.3:5)
      case 1:  regC=alu8out.0:7;
      default: regC=regC;
   endswitch
endif

if (chgregC3p)
   switch(code0d.4:5)
      case 0:  regC=alu16out.0:7;
      default: regC=regC;
   endswitch
endif

if (chgregC4p)
   switch(code0d.0:2)
      case 1:  regC=alu8out.0:7;
      default: regC=regC;
   endswitch
endif
```

```
if (chgregC5p)
   if (code0d.3:5==0b001)
      regC=code1d;
   else
      regC=regC;
   endif
endif

if (chgregC6p)
   if (code0d.3:5==0b001)
      switch(code0d.0:2)
         case 0: regC=regB;
         case 1: regC=regC;
         case 2: regC=regD;
         case 3: regC=regE;
         case 4: regC=regH;
         case 5: regC=regL;
         case 7: regC=regA;
      endswitch
   else
      regC=regC;
   endif
endif

if (chgregC8p)
   switch(code0d.3:5)
      case 1: regC=DATAIN;
      default:regC=regC;
   endswitch
endif

if (chgregC9p)
   switch(code0d.4:5)
      case 0: regC=code2d;
      default:regC=regC;
   endswitch
endif

if (chgregC10p)
   switch(code0d.4:5)
      case 0: regC=DATAIN;
      default:regC=regC;
   endswitch
endif

if (chgregC11p)
   switch(code0d.4:5)
      case 0: regC=DATAIN;
      default:regC=regC;
   endswitch
endif

if (chgregC12p) regC=regRC; endif

if ( chgregC1p
   | chgregC2p
   | chgregC3p
   | chgregC4p
   | chgregC5p
   | chgregC6p
   | chgregC8p
   | chgregC9p
   | chgregC10p
   | chgregC11p
   | chgregC12p
```

```
      )
   else
      if (RESET)
         regC=0;
      else
         regC=regC;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ D                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.1&anct1p) chgregD6p=1; endif   { LD D,n }
         if (instp.2&anct1p) chgregD7p=1; endif   { LD D,r }

         if (anct1p)
            if ( instp.3    { LD D,(HL) }
               | instp.7    { LD D,(IX[IY]+d) }
               | instp.188  { IN D,(C) }
               )
               chgregD9p=1;
            endif
         endif

         if (instp.20&anct1p) chgregD10p=1; endif { LD DE,n'n }

         if (anct1p)
            if (instp.23)   { LD DE,(n'n) }
               chgregD11p=1;
            endif
         endif

         if (anct1p)
            if (instp.33)   { POP DE }
               switch(code0d.4:5)
                  case 1: chgregD11p=1;
               endswitch
            endif
         endif

         if (instp.38&anct1p) chgregD12p=1; endif { EX DE,HL }
         if (instp.40&anct1p) chgregD13p=1; endif { EXX }

         if (anct1p)
            if ( instp.45       { LDD }
               | instp.47       { LDDR }
               )
               chgregD1p=1;
            endif
         endif

         if (anct1p)
            if ( instp.44       { LDI }
               | instp.46       { LDIR }
               )
               chgregD2p=1;
            endif
         endif

         if (anct1p)
            if ( instp.92       { INC D }
               | instp.96       { DEC D }
               )
```

```
            chgregD3p=1;
         endif
      endif

      if (anct1p)
         if ( instp.116      { INC DE }
            | instp.119      { DEC DE }
            )
            chgregD4p=1;
         endif
      endif

      if (anct1p)
         if ( instp.126      { RLC E }
            | instp.130      { RL E }
            | instp.134      { RRC E }
            | instp.138      { RR E }
            | instp.142      { SLA E }
            | instp.146      { SRL E }
            | instp.150      { SRA E }
            | instp.160      { SET b,E }
            | instp.164      { RES b,E }
            )
            chgregD5p=1;
         endif
      endif
   endif
endif

if (chgregD1p)
   switch(regE)
      case 0x00: regD=regD-1; { LDDR }
      default: regD=regD;
   endswitch
endif

if (chgregD2p)
   switch(regE)
      case 0xff: regD=regD+1; { LDIR }
      default: regD=regD;
   endswitch
endif

if (chgregD3p)
   switch(code0d.3:5)
      case 2:  regD=alu8out.0:7;
      default: regD=regD;
   endswitch
endif

if (chgregD4p)
   switch(code0d.4:5)
      case 1:  regD=alu16out.8:15;
      default: regD=regD;
   endswitch
endif

if (chgregD5p)
   switch(code0d.0:2)
      case 2:  regD=alu8out.0:7;
      default: regD=regD;
   endswitch
endif

if (chgregD6p)
```

```
   if (code0d.3:5==0b010)
      regD=code1d;
   else
      regD=regD;
   endif
endif

if (chgregD7p)
   if (code0d.3:5==0b010)
      switch(code0d.0:2)
         case 0: regD=regB;
         case 1: regD=regC;
         case 2: regD=regD;
         case 3: regD=regE;
         case 4: regD=regH;
         case 5: regD=regL;
         case 7: regD=regA;
      endswitch
   else
      regD=regD;
   endif
endif

if (chgregD9p)
   switch(code0d.3:5)
      case 2: regD=DATAIN;
      default:regD=regD;
   endswitch
endif

if (chgregD10p)
   switch(code0d.4:5)
      case 1: regD=code1d;
      default:regD=regD;
   endswitch
endif

if (chgregD11p)
   switch(code0d.4:5)
      case 1: regD=DATAIN;
      default:regD=regD;
   endswitch
endif

if (chgregD12p) regD=regH; endif
if (chgregD13p) regD=regRD; endif

if ( chgregD1p
   | chgregD2p
   | chgregD3p
   | chgregD4p
   | chgregD5p
   | chgregD6p
   | chgregD7p
   | chgregD9p
   | chgregD10p
   | chgregD11p
   | chgregD12p
   | chgregD13p
   )
else
   if (RESET)
      regD=0;
   else
      regD=regD;
   endif
```

```
   endif

{ ------------------------------------------------------------------- }
{    レジスタ E                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.1&anct1p) chgregE6p=1; endif   { LD E,n }
         if (instp.2&anct1p) chgregE7p=1; endif   { LD E,r }

         if (anct1p)
            if ( instp.3   { LD E,(HL) }
               | instp.7   { LD E,(IX[IY]+d) }
               | instp.188 { IN r,(C) }
               )
               chgregE9p=1;
            endif
         endif

         if (instp.20&anct1p) chgregE10p=1; endif { LD DE,n'n }

         if (anct2p)
            if (instp.23)  { LD DE,(n'n) }
               chgregE11p=1;
            endif
         endif

         if (anct2p)
            if (instp.33)  { POP DE }
               switch(code0d.4:5)
                  case 1: chgregE11p=1;
               endswitch
            endif
         endif

         if (instp.38&anct1p) chgregE12p=1; endif { EX DE,HL }
         if (instp.40&anct1p) chgregE13p=1; endif { EXX }

         if (anct1p)
            if ( instp.45     { LDD }
               | instp.47     { LDDR }
               )
               chgregE1p=1;
            endif
         endif

         if (anct1p)
            if ( instp.44     { LDI }
               | instp.46     { LDIR }
               )
               chgregE2p=1;
            endif
         endif

         if (anct1p)
            if ( instp.92     { INC E }
               | instp.96     { INC E }
               )
               chgregE3p=1;
            endif
         endif

         if (anct1p)
            if ( instp.116    { INC DE }
```

```
            | instp.119    { DEC DE }
            )
            chgregE4p=1;
         endif
      endif

      if (anct1p)
         if ( instp.126    { RLC E }
            | instp.130    { RL E }
            | instp.134    { RRC E }
            | instp.138    { RR E }
            | instp.142    { SLA E }
            | instp.146    { SRL E }
            | instp.150    { SRA E }
            | instp.160    { SET b,E }
            | instp.164    { RES b,E }
            )
            chgregE5p=1;
         endif
      endif
   endif
endif

if (chgregE1p) regE=regE-1; endif
if (chgregE2p) regE=regE+1; endif

if (chgregE3p)
   switch(code0d.3:5)
      case 3:  regE=alu8out.0:7;
      default: regE=regE;
   endswitch
endif

if (chgregE4p)
   switch(code0d.4:5)
      case 1:  regE=alu16out.0:7;
      default: regE=regE;
   endswitch
endif

if (chgregE5p)
   switch(code0d.0:2)
      case 3:  regE=alu8out.0:7;
      default: regE=regE;
   endswitch
endif

if (chgregE6p)
   if (code0d.3:5==0b011)
      regE=code1d;
   else
      regE=regE;
   endif
endif

if (chgregE7p)
   if (code0d.3:5==0b011)
      switch(code0d.0:2)
         case 0: regE=regB;
         case 1: regE=regC;
         case 2: regE=regD;
         case 3: regE=regE;
         case 4: regE=regH;
         case 5: regE=regL;
         case 7: regE=regA;
```

```
            endswitch
         else
            regE=regE;
         endif
      endif

      if (chgregE9p)
         switch(code0d.3:5)
            case 3: regE=DATAIN;
            default:regE=regE;
         endswitch
      endif

      if (chgregE10p)
         switch(code0d.4:5)
            case 1: regE=code2d;
            default:regE=regE;
         endswitch
      endif

      if (chgregE11p)
         switch(code0d.4:5)
            case 1: regE=DATAIN;
            default:regE=regE;
         endswitch
      endif

      if (chgregE12p) regE=regL; endif
      if (chgregE13p) regE=regRE; endif

      if ( chgregE1p
         | chgregE2p
         | chgregE3p
         | chgregE4p
         | chgregE5p
         | chgregE6p
         | chgregE7p
         | chgregE9p
         | chgregE10p
         | chgregE11p
         | chgregE12p
         | chgregE13p
         )
      else
         if (RESET)
            regE=0;
         else
            regE=regE;
         endif
      endif

   { ------------------------------------------------------------------- }
   {    レジスタ H                                                      }
   { ------------------------------------------------------------------- }
      if (!RESET)
         if (!WAIT)
            if (instp.1&anct1p) chgregH7p=1; endif   { LD H,n }
            if (instp.2&anct1p) chgregH8p=1; endif   { LD H,r }

            if (anct1p)
               if ( instp.3    { LD H,(HL) }
                  | instp.7    { LD H,(IX[IY]+d) }
                  | instp.188  { IN r,(C) }
                  )
                  chgregH10p=1;
```

```
   endif
endif

if (instp.20&anct1p) chgregH11p=1; endif { LD HL,n'n }

if (anct1p)
   if (instp.21)   { LD HL,(n'n) }
      chgregH12p=1;
   endif
endif

if (anct1p)
   if (instp.33)   { POP HL }
      switch(code0d.4:5))
          case 2: chgregH12p=1;
      endswitch
   endif
endif

if (instp.23&anct1p) chgregH13p=1; endif { LD HL,(n'n) }
if (instp.38&anct1p) chgregH14p=1; endif { EX DE,HL }
if (instp.40&anct1p) chgregH15p=1; endif { EXX }
if (instp.41&anct2p) chgregH16p=1; endif { EX (SP),HL }

if (anct1p)
   if ( instp.45       { LDD }
      | instp.47       { LDDR }
      | instp.49       { CPD }
      | instp.51       { CPDR }
      )
      chgregH1p=1;
   endif
endif

if (anct1p)
   if ( instp.44       { LDI }
      | instp.46       { LDIR }
      | instp.48       { CPI }
      | instp.50       { CPIR }
      )
      chgregH2p=1;
   endif
endif

if (anct1p)
   if ( instp.92       { INC H }
      | instp.96       { DEC H }
      )
      chgregH3p=1;
   endif
endif

if (anct1p)
   if ( instp.111      { ADD HL,ss }
      | instp.112      { ADC HL,ss }
      | instp.113      { SBC HL,ss }
      | instp.190      { INI }
      | instp.191      { IND }
      | instp.192      { INIR }
      | instp.193      { INDR }
      | instp.194      { OUTI }
      | instp.195      { OUTD }
      | instp.196      { OTIR }
```

```
            | instp.197      { OTDR }
            )
            chgregH4p=1;
         endif
      endif

      if (anct1p)
         if ( instp.116      { INC HL }
            | instp.119      { DEC HL }
            )
            chgregH5p=1;
         endif
      endif

      if (anct1p)
         if ( instp.126      { RLC H }
            | instp.130      { RL H }
            | instp.134      { RRC H }
            | instp.138      { RR H }
            | instp.142      { SLA H }
            | instp.146      { SRL H }
            | instp.150      { SRA H }
            | instp.160      { SET b,H }
            | instp.164      { RES b,H }
            )
            chgregH6p=1;
         endif
      endif
   endif
endif

if (chgregH1p)
   switch(regL)
      case 0x00: regH=regH-1; { CPDR }
      default: regH=regH;
   endswitch
endif

if (chgregH2p)
   switch(regL)
      case 0xff: regH=regH+1; { CPIR }
      default: regH=regH;
   endswitch
endif

if (chgregH3p)
   switch(code0d.3:5)
      case 4:  regH=alu8out.0:7;
      default: regH=regH;
   endswitch
endif

if (chgregH4p) regH=alu16out.8:15; endif

if (chgregH5p)
   switch(code0d.4:5)
      case 2:  regH=alu16out.8:15;
      default: regH=regH;
   endswitch
endif

if (chgregH6p)
   switch(code0d.0:2)
      case 4:  regH=alu8out.0:7;
      default: regH=regH;
```

```
   endswitch
endif

if (chgregH7p)
   if (code0d.3:5==0b100)
      regH=code1d;
   else
      regH=regH;
   endif
endif

if (chgregH8p)
   if (code0d.3:5==0b100)
      switch(code0d.0:2)
         case 0: regH=regB;
         case 1: regH=regC;
         case 2: regH=regD;
         case 3: regH=regE;
         case 4: regH=regH;
         case 5: regH=regL;
         case 7: regH=regA;
      endswitch
   else
      regH=regH;
   endif
endif

if (chgregH10p)
   switch(code0d.3:5)
      case 4: regH=DATAIN;
      default:regH=regH;
   endswitch
endif

if (chgregH11p)
   switch(code0d.4:5)
      case 2: regH=code1d;
      default:regH=regH;
   endswitch
endif

if (chgregH12p) regH=DATAIN; endif

if (chgregH13p)
   switch(code0d.4:5)
      case 2: regH=DATAIN;
      default:regH=regH;
   endswitch
endif

if (chgregH14p) regH=regD; endif
if (chgregH15p) regH=regRH; endif
if (chgregH16p) regH=reg1T; endif

if ( chgregH1p
   | chgregH2p
   | chgregH3p
   | chgregH4p
   | chgregH5p
   | chgregH6p
   | chgregH7p
   | chgregH8p
   | chgregH10p
   | chgregH11p
   | chgregH12p
   | chgregH13p
```

```
      | chgregH14p
      | chgregH15p
      | chgregH16p
      )
   else
      if (RESET)
         regH=0;
      else
         regH=regH;
      endif
   endif

{ ------------------------------------------------------------------- }
{    レジスタ L                                                       }
{ ------------------------------------------------------------------- }
   if (!RESET)
      if (!WAIT)
         if (instp.1&anct1p) chgregL7p=1; endif  { LD H,n }
         if (instp.2&anct1p) chgregL8p=1; endif  { LD H,r }

         if (anct1p)
            if ( instp.3    { LD L,(HL) }
               | instp.7    { LD L,(IX[IY]+d) }
               | instp.188  { IN r,(C) }
               )
               chgregL10p=1;
            endif
         endif

         if (instp.20&anct1p) chgregL11p=1; endif { LD HL,n'n }

         if (anct2p)
            if (instp.21)   { LD HL,(n'n) }
               chgregL12p=1;
            endif
         endif

         if (anct2p)
            if (instp.33)   { POP HL }
               switch(code0d.4:5))
                  case 2: chgregL12p=1;
               endswitch
            endif
         endif

         if (instp.23&anct2p) chgregL13p=1; endif { LD HL,(n'n) }
         if (instp.38&anct1p) chgregL14p=1; endif { EX DE,HL }
         if (instp.40&anct1p) chgregL15p=1; endif { EXX }
         if (instp.41&anct1p) chgregL16p=1; endif { EX (SP),HL }

         if (anct1p)
            if ( instp.45     { LDD }
               | instp.47     { LDDR }
               | instp.49     { CPD }
               | instp.51     { CPDR }
               )
               chgregL1p=1;
            endif
         endif

         if (anct1p)
            if ( instp.44     { LDI }
               | instp.46     { LDIR }
               | instp.48     { CPI }
```

```
            | instp.50     { CPIR }
            )
            chgregL2p=1;
         endif
      endif

      if (anct1p)
         if ( instp.92     { INC L }
            | instp.96     { DEC L }
            )
            chgregL3p=1;
         endif
      endif

      if (anct1p)
         if ( instp.111    { ADD HL,ss }
            | instp.112    { ADC HL,ss }
            | instp.113    { SBC HL,ss }
            | instp.190    { INI }
            | instp.191    { IND }
            | instp.192    { INIR }
            | instp.193    { INDR }
            | instp.194    { OUTI }
            | instp.195    { OUTD }
            | instp.196    { OTIR }
            | instp.197    { OTDR }
            )
            chgregL4p=1;
         endif
      endif

      if (anct1p)
         if ( instp.116    { INC HL }
            | instp.119    { DEC HL }
            )
            chgregL5p=1;
         endif
      endif

      if (anct1p)
         if ( instp.126    { RLC L }
            | instp.130    { RL L }
            | instp.134    { RRC L }
            | instp.138    { RR L }
            | instp.142    { SLA L }
            | instp.146    { SRL L }
            | instp.150    { SRA L }
            | instp.160    { SET b,L }
            | instp.164    { RES b,L }
            )
            chgregL6p=1;
         endif
      endif
   endif
endif

if (chgregL1p) regL=regL-1; endif
if (chgregL2p) regL=regL+1; endif

if (chgregL3p)
   switch(code0d.3:5)
      case 5:  regL=alu8out.0:7;
      default: regL=regL;
   endswitch
```

```
endif

if (chgregL4p) regL=alu16out.0:7; endif

if (chgregL5p)
   switch(code0d.4:5)
      case 2:  regL=alu16out.0:7;
      default: regL=regL;
   endswitch
endif

if (chgregL6p)
   switch(code0d.0:2)
      case 5:  regL=alu8out.0:7;
      default: regL=regL;
   endswitch
endif

if (chgregL7p)
   if (code0d.3:5==0b101)
      regL=code1d;
   else
      regL=regL;
   endif
endif

if (chgregL8p)
   if (code0d.3:5==0b101)
      switch(code0d.0:2)
         case 0: regL=regB;
         case 1: regL=regC;
         case 2: regL=regD;
         case 3: regL=regE;
         case 4: regL=regH;
         case 5: regL=regL;
         case 7: regL=regA;
      endswitch
   else
      regL=regL;
   endif
endif

if (chgregL10p)
   switch(code0d.3:5)
      case 5: regL=DATAIN;
      default:regL=regL;
   endswitch
endif

if (chgregL11p)
   switch(code0d.4:5)
      case 2: regL=code2d;
      default:regL=regL;
   endswitch
endif

if (chgregL12p) regL=DATAIN; endif

if (chgregL13p)
   switch(code0d.4:5)
      case 2: regL=DATAIN;
      default:regL=regL;
   endswitch
endif

if (chgregL14p) regL=regE; endif
```

```
   if (chgregL15p) regL=regRL; endif
   if (chgregL16p) regL=reg0T; endif

   if ( chgregL1p
      | chgregL2p
      | chgregL3p
      | chgregL4p
      | chgregL5p
      | chgregL6p
      | chgregL7p
      | chgregL8p
      | chgregL10p
      | chgregL11p
      | chgregL12p
      | chgregL13p
      | chgregL14p
      | chgregL15p
      | chgregL16p
      )
   else
      if (RESET)
         regL=0;
      else
         regL=regL;
      endif
   endif

{ ------------------------------------------------------------------- }
{    8 ビットキャリー                                                 }
{ ------------------------------------------------------------------- }
   carry.0=flgC;

{ ------------------------------------------------------------------- }
{    16 ビットキャリー                                                }
{ ------------------------------------------------------------------- }
   wcarry.0=flgC;

{ ------------------------------------------------------------------- }
{    4 ビット加算                                                     }
{ ------------------------------------------------------------------- }
   alu4p=alu4ina+alu4inb;

{ ------------------------------------------------------------------- }
{    4 ビット加算桁上げ                                               }
{ ------------------------------------------------------------------- }
   alu4pc=alu4ina+alu4inb;

{ ------------------------------------------------------------------- }
{    8 ビット加算                                                     }
{ ------------------------------------------------------------------- }
   alu8p=alu8ina+alu8inb;

{ ------------------------------------------------------------------- }
{    8 ビット加算桁上げ                                               }
{ ------------------------------------------------------------------- }
   alu8pc=alu8ina+alu8inb+carry;

{ ------------------------------------------------------------------- }
{    8 ビット左桁移動                                                 }
{ ------------------------------------------------------------------- }
   alu8ls.8=alu8ina.7;
   alu8ls.7=alu8ina.6;
   alu8ls.6=alu8ina.5;
   alu8ls.5=alu8ina.4;
```

```
    alu8ls.4=alu8ina.3;
    alu8ls.3=alu8ina.2;
    alu8ls.2=alu8ina.1;
    alu8ls.1=alu8ina.0;
    alu8ls.0=alu8ina.7;

{ ------------------------------------------------------------------- }
{    8 ビット左桁移動キャリー                                         }
{ ------------------------------------------------------------------- }
    alu8lsc.8=alu8ina.7;
    alu8lsc.7=alu8ina.6;
    alu8lsc.6=alu8ina.5;
    alu8lsc.5=alu8ina.4;
    alu8lsc.4=alu8ina.3;
    alu8lsc.3=alu8ina.2;
    alu8lsc.2=alu8ina.1;
    alu8lsc.1=alu8ina.0;
    alu8lsc.0=flgC;

{ ------------------------------------------------------------------- }
{    8 ビット左桁移動 0 入力                                          }
{ ------------------------------------------------------------------- }
    alu8lsz.8=alu8ina.7;
    alu8lsz.7=alu8ina.6;
    alu8lsz.6=alu8ina.5;
    alu8lsz.5=alu8ina.4;
    alu8lsz.4=alu8ina.3;
    alu8lsz.3=alu8ina.2;
    alu8lsz.2=alu8ina.1;
    alu8lsz.1=alu8ina.0;
    alu8lsz.0=0;

{ ------------------------------------------------------------------- }
{    8 ビット右桁移動                                                 }
{ ------------------------------------------------------------------- }
    alu8rs.8=alu8ina.0;
    alu8rs.7=alu8ina.0;
    alu8rs.6=alu8ina.7;
    alu8rs.5=alu8ina.6;
    alu8rs.4=alu8ina.5;
    alu8rs.3=alu8ina.4;
    alu8rs.2=alu8ina.3;
    alu8rs.1=alu8ina.2;
    alu8rs.0=alu8ina.1;

{ ------------------------------------------------------------------- }
{    8 ビット右桁移動キャリー                                         }
{ ------------------------------------------------------------------- }
    alu8rsc.8=alu8ina.0;
    alu8rsc.7=flgC;
    alu8rsc.6=alu8ina.7;
    alu8rsc.5=alu8ina.6;
    alu8rsc.4=alu8ina.5;
    alu8rsc.3=alu8ina.4;
    alu8rsc.2=alu8ina.3;
    alu8rsc.1=alu8ina.2;
    alu8rsc.0=alu8ina.1;

{ ------------------------------------------------------------------- }
{    8 ビット右桁移動 0 入力                                          }
{ ------------------------------------------------------------------- }
    alu8rsz.8=alu8ina.0;
    alu8rsz.7=0;
    alu8rsz.6=alu8ina.7;
    alu8rsz.5=alu8ina.6;
```

```
   alu8rsz.4=alu8ina.5;
   alu8rsz.3=alu8ina.4;
   alu8rsz.2=alu8ina.3;
   alu8rsz.1=alu8ina.2;
   alu8rsz.0=alu8ina.1;

{ ------------------------------------------------------------------- }
{    8 ビット右桁移動符号固定                                         }
{ ------------------------------------------------------------------- }
   alu8rss.8=alu8ina.0;
   alu8rss.7=alu8ina.7;
   alu8rss.6=alu8ina.7;
   alu8rss.5=alu8ina.6;
   alu8rss.4=alu8ina.5;
   alu8rss.3=alu8ina.4;
   alu8rss.2=alu8ina.3;
   alu8rss.1=alu8ina.2;
   alu8rss.0=alu8ina.1;

{ ------------------------------------------------------------------- }
{    4 ビット減算                                                     }
{ ------------------------------------------------------------------- }
   alu4m=alu4ina+(!alu4inb+1);

{ ------------------------------------------------------------------- }
{    4 ビット減算桁借り                                               }
{ ------------------------------------------------------------------- }
   alu4mc=(alu4ina+(!alu4inb+1))+(!carry.0:4+1);

{ ------------------------------------------------------------------- }
{    8 ビット減算                                                     }
{ ------------------------------------------------------------------- }
   alu8m=alu8ina+(!alu8inb+1);

{ ------------------------------------------------------------------- }
{    8 ビット減算桁借り                                               }
{ ------------------------------------------------------------------- }
   alu8mc=(alu8ina+(!alu8inb+1))+(!carry+1);

{ ------------------------------------------------------------------- }
{    8 ビット論理積                                                   }
{ ------------------------------------------------------------------- }
   alu8and=alu8ina&alu8inb;

{ ------------------------------------------------------------------- }
{    8 ビット論理和                                                   }
{ ------------------------------------------------------------------- }
   alu8or=alu8ina|alu8inb;

{ ------------------------------------------------------------------- }
{    8 ビット排他的論理和                                             }
{ ------------------------------------------------------------------- }
   alu8xor=alu8ina^alu8inb;

{ ------------------------------------------------------------------- }
{    16 ビット加算                                                    }
{ ------------------------------------------------------------------- }
   alu16p=alu16ina+alu16inb;

{ ------------------------------------------------------------------- }
{    16 ビット加算桁上げ                                              }
{ ------------------------------------------------------------------- }
   alu16pc=alu16ina+alu16inb+wcarry;
```

```
{ ------------------------------------------------------------------------ }
{    16 ビット減算                                                        }
{ ------------------------------------------------------------------------ }
   alu16m=alu16ina+(!alu16inb+1);

{ ------------------------------------------------------------------------ }
{    16 ビット減算桁借り                                                  }
{ ------------------------------------------------------------------------ }
   alu16mc=(alu16ina+(!alu16inb+1))+(!wcarry+1);

{ ------------------------------------------------------------------------ }
{    4 ビット演算結果                                                     }
{ ------------------------------------------------------------------------ }
   if ( instp.52   { ADD A,r }
      | instp.53   { ADD A,n }
      | instp.54   { ADD A,(HL) }
      | instp.55   { ADD A,(IX+d) }
      | instp.56   { ADD A,(IY+d) }
      | instp.92   { INC r }
      | instp.93   { INC (HL) }
      | instp.94   { INC (IX+d) }
      | instp.95   { INC (IY+d) }
      | instp.96   { DEC r }
      | instp.97   { DEC (HL) }
      | instp.98   { DEC (IX+d) }
      | instp.99   { DEC (IY+d) }
      | instp.102  { NEG }
      )
      chgalu4out1p=1;
   endif

   if ( instp.62   { SUB r }
      | instp.63   { SUB n }
      | instp.64   { SUB (HL) }
      | instp.65   { SUB (IX+d) }
      | instp.66   { SUB (IY+d) }
      | instp.87   { CP r }
      | instp.88   { CP n }
      | instp.89   { CP (HL) }
      | instp.90   { CP (IX+d) }
      | instp.91   { CP (IY+d) }
      )
      chgalu4out2p=1;
   endif

   if ( instp.57   { ADC A,r }
      | instp.58   { ADC A,n }
      | instp.59   { ADC A,(HL) }
      | instp.60   { ADC A,(IX+d) }
      | instp.61   { ADC A,(IY+d) }
      )
      chgalu4out3p=1;
   endif

   if ( instp.67   { SBC A,r }
      | instp.68   { SBC A,n }
      | instp.69   { SBC A,(HL) }
      | instp.70   { SBC A,(IX+d) }
      | instp.71   { SBC A,(IY+d) }
      )
      chgalu4out4p=1;
   endif
```

```
   if (chgalu4out1p) alu4out=alu4p; endif
   if (chgalu4out2p) alu4out=alu4m; endif
   if (chgalu4out3p) alu4out=alu4pc; endif
   if (chgalu4out4p) alu4out=alu4mc; endif

{ ------------------------------------------------------------------- }
{    4 ビット演算器入力 a                                            }
{ ------------------------------------------------------------------- }
   if ( instp.52   { ADD A,r }
      | instp.53   { ADD A,n }
      | instp.54   { ADD A,(HL) }
      | instp.55   { ADD A,(IX+d) }
      | instp.56   { ADD A,(IY+d) }
      | instp.57   { ADC A,r }
      | instp.58   { ADC A,n }
      | instp.59   { ADC A,(HL) }
      | instp.60   { ADC A,(IX+d) }
      | instp.61   { ADC A,(IY+d) }
      | instp.62   { SUB r }
      | instp.63   { SUB n }
      | instp.64   { SUB (HL) }
      | instp.65   { SUB (IX+d) }
      | instp.66   { SUB (IY+d) }
      | instp.67   { SBC A,r }
      | instp.68   { SBC A,n }
      | instp.69   { SBC A,(HL) }
      | instp.70   { SBC A,(IX+d) }
      | instp.71   { SBC A,(IY+d) }
      | instp.87   { CP r }
      | instp.88   { CP n }
      | instp.89   { CP (HL) }
      | instp.90   { CP (IX+d) }
      | instp.91   { CP (IY+d) }
      )
      chgalu4ina1p=1;
   endif

   if ( instp.92   { INC r }
      | instp.93   { INC (HL) }
      | instp.94   { INC (IX+d) }
      | instp.95   { INC (IY+d) }
      )
      chgalu4ina2p=1;
   endif

   if ( instp.96   { DEC r }
      | instp.97   { DEC (HL) }
      | instp.98   { DEC (IX+d) }
      | instp.99   { DEC (IY+d) }
      )
      chgalu4ina3p=1;
   endif

   if (instp.102)  { NEG }
      chgalu4ina4p=1;
   endif


   if (chgalu4ina1p) alu4ina.0:3=regA.0:3; endif
   if (chgalu4ina2p) alu4ina.0:3=1; endif
   if (chgalu4ina3p) alu4ina.0:3=0xf; endif
   if (chgalu4ina4p) alu4ina.0:3=!regA.0:3; endif
```

```
{ ------------------------------------------------------------------------ }
{    4 ビット演算器入力 b                                                  }
{ ------------------------------------------------------------------------ }
   if ( instp.52           { ADD A,r }
      | instp.57           { ADC A,r }
      | instp.62           { SUB r }
      | instp.67           { SBC A,r }
      | instp.87           { CP r }
      )
      chgalu4in1p=1;
   endif

   if ( instp.53           { ADD A,n }
      | instp.54           { ADD A,(HL) }
      | instp.55           { ADD A,(IX+d) }
      | instp.56           { ADD A,(IY+d) }
      | instp.58           { ADC A,n }
      | instp.59           { ADC A,(HL) }
      | instp.60           { ADC A,(IX+d) }
      | instp.61           { ADC A,(IY+d) }
      | instp.63           { SUB n }
      | instp.64           { SUB (HL) }
      | instp.65           { SUB (IX+d) }
      | instp.66           { SUB (IY+d) }
      | instp.68           { SBC A,n }
      | instp.69           { SBC A,(HL) }
      | instp.70           { SBC A,(IX+d) }
      | instp.71           { SBC A,(IY+d) }
      | instp.88           { CP n }
      | instp.89           { CP (HL) }
      | instp.90           { CP (IX+d) }
      | instp.91           { CP (IY+d) }
      | instp.93           { INC (HL) }
      | instp.94           { INC (IX+d) }
      | instp.95           { INC (IY+d) }
      | instp.97           { DEC (HL) }
      | instp.98           { DEC (IX+d) }
      | instp.99           { DEC (IY+d) }
      )
      chgalu4in2p=1;
   endif

   if (instp.102) chgalu4in3p=1; endif { NEG }

   if ( instp.92           { INC r }
      | instp.96           { DEC r }
      )
      chgalu4in4p=1;
   endif

   if (chgalu4in1p)
      switch(code0d.0:2)
         case 0: alu4inb.0:3=regB.0:3;
         case 1: alu4inb.0:3=regC.0:3;
         case 2: alu4inb.0:3=regD.0:3;
         case 3: alu4inb.0:3=regE.0:3;
         case 4: alu4inb.0:3=regH.0:3;
         case 5: alu4inb.0:3=regL.0:3;
         case 7: alu4inb.0:3=regA.0:3;
      endswitch
   endif
```

```
   if (chgalu4in2p)
      alu4inb.0:3=DATAIN.0:3;
   endif

   if (chgalu4in3p)
      alu4inb.0:3=1;
   endif

   if (chgalu4in4p)
      switch(code0d.3:5)
         case 0: alu4inb.0:3=regB.0:3;
         case 1: alu4inb.0:3=regC.0:3;
         case 2: alu4inb.0:3=regD.0:3;
         case 3: alu4inb.0:3=regE.0:3;
         case 4: alu4inb.0:3=regH.0:3;
         case 5: alu4inb.0:3=regL.0:3;
         case 7: alu4inb.0:3=regA.0:3;
      endswitch
   endif

{ ------------------------------------------------------------------- }
{     16 ビット演算器                                                 }
{ ------------------------------------------------------------------- }
   if (instp.112) alu16out=alu16pc; endif { ADC HL,ss }
   if (instp.113) alu16out=alu16mc; endif { SBC HL,ss }

   if ( instp.111   { ADD HL,ss }
      | instp.192   { INIR }
      | instp.194   { OUTI }
      | instp.196   { OTIR }
      | instp.114   { ADD IX,pp }
      | instp.115   { ADD IY,rr }
      | instp.116   { INC ss }
      | instp.117   { INC IX }
      | instp.118   { INC IY }
      | instp.190   { INI }
      )
      chgalu16out1p=1;
   endif

   if ( instp.119   { DEC ss }
      | instp.120   { DEC IX }
      | instp.121   { DEC IY }
      | instp.191   { IND }
      | instp.193   { INDR }
      | instp.195   { OUTD }
      | instp.197   { OTDR }
      )
      chgalu16out2p=1;
   endif

   if (chgalu16out1p) alu16out=alu16p; endif
   if (chgalu16out2p) alu16out=alu16m; endif

{ ------------------------------------------------------------------- }
{     16 ビット演算器入力 a                                           }
{ ------------------------------------------------------------------- }
   if ( instp.111  { ADD HL,ss }
      | instp.112  { ADC HL,ss }
      | instp.113  { SBC HL,ss }
      | instp.190  { INI }
      | instp.191  { IND }
      | instp.192  { INIR }
```

```
      | instp.193  { INDR }
      | instp.194  { OUTI }
      | instp.195  { OUTD }
      | instp.196  { OTIR }
      | instp.197  { OTDR }
      )
      alu16ina1p=1;
   endif

   if ( instp.114  { ADD IX,pp }
      | instp.120  { DEC IX }
      )
      alu16ina2p=1;
   endif

   if ( instp.115  { ADD IY,rr }
      | instp.121  { DEC IY }
      )
      alu16ina3p=1;
   endif

   if ( instp.116  { INC ss }
      | instp.117  { INC IX }
      | instp.118  { INC IY }
      )
      alu16ina4p=1;
   endif

   if (instp.119) alu16ina5p=1; endif { DEC ss }

   if (alu16ina1p)
      alu16ina.0:7=regL;
      alu16ina.8:15=regH;
   endif

   if (alu16ina2p) alu16ina.0:15=regIX; endif
   if (alu16ina3p) alu16ina.0:15=regIY; endif
   if (alu16ina4p) alu16ina=1; endif

   if (alu16ina5p)
      switch(code0d.4:5)
         case 0: alu16ina.0:7=regC;
                 alu16ina.8:15=regB;
         case 1: alu16ina.0:7=regE;
                 alu16ina.8:15=regD;
         case 2: alu16ina.0:7=regL;
                 alu16ina.8:15=regH;
         case 3: alu16ina.0:7=regSP.0:7;
                 alu16ina.8:15=regSP.8:15;
      endswitch
   endif

{ ------------------------------------------------------------------- }
{    16 ビット演算器入力 b                                          }
{ ------------------------------------------------------------------- }
   if ( instp.111            { ADD HL,ss }
      | instp.112            { ADC HL,ss }
      | instp.113            { SBC HL,ss }
      | instp.116            { INC ss }
      )
      alu16inb1p=1;
   endif

   if (instp.114) alu16inb2p=1; endif { ADD IX,pp }
```

```
   if (instp.115) alu16inb3p=1; endif { ADD IY,rr }
   if (instp.117) alu16inb4p=1; endif { INC IX }
   if (instp.118) alu16inb5p=1; endif { INC IY }

   if ( instp.119            { DEC ss }
      | instp.120            { DEC IX }
      | instp.121            { DEC IY }
      | instp.190            { INI }
      | instp.191            { IND }
      | instp.192            { INIR }
      | instp.193            { INDR }
      | instp.194            { OUTI }
      | instp.195            { OUTD }
      | instp.196            { OTIR }
      | instp.197            { OTDR }
      )
      alu16inb6p=1;
   endif

   if (alu16inb1p)
      switch(code0d.4:5)
         case 0: alu16inb.0:7=regC;
                 alu16inb.8:15=regB;
         case 1: alu16inb.0:7=regE;
                 alu16inb.8:15=regD;
         case 2: alu16inb.0:7=regL;
                 alu16inb.8:15=regH;
         case 3: alu16inb.0:7=regSP.0:7;
                 alu16inb.8:15=regSP.8:15;
      endswitch
   endif

   if (alu16inb2p)
      switch(code0d.4:5)
         case 0: alu16inb.0:7=regC;
                 alu16inb.8:15=regB;
         case 1: alu16inb.0:7=regE;
                 alu16inb.8:15=regD;
         case 2: alu16inb.0:7=regIX.0:7;
                 alu16inb.8:15=regIX.8:15;
         case 3: alu16inb.0:7=regSP.0:7;
                 alu16inb.8:15=regSP.8:15;
      endswitch
   endif

   if (alu16inb3p)
      switch(code0d.4:5)
         case 0: alu16inb.0:7=regC;
                 alu16inb.8:15=regB;
         case 1: alu16inb.0:7=regE;
                 alu16inb.8:15=regD;
         case 2: alu16inb.0:7=regIY.0:7;
                 alu16inb.8:15=regIY.8:15;
         case 3: alu16inb.0:7=regSP.0:7;
                 alu16inb.8:15=regSP.8:15;
      endswitch
   endif

   if (alu16inb4p) alu16inb.0:15=regIX; endif
   if (alu16inb5p) alu16inb.0:15=regIY; endif
   if (alu16inb6p) alu16inb=1; endif

{ ------------------------------------------------------------------- }
{    8ビット演算器                                                    }
{ ------------------------------------------------------------------- }
```

```
if (instp.100)                { DAA }
   alu8out.0:3=daalA.0:3;
   alu8out.4:7=daahA.0:3;
endif

if (instp.101) alu8out=!regA; endif { CPL }

if ( instp.52  { ADD A,r }
   | instp.53  { ADD A,n }
   | instp.54  { ADD A,(HL) }
   | instp.55  { ADD A,(IX+d) }
   | instp.56  { ADD A,(IY+d) }
   | instp.92  { INC r }
   | instp.93  { INC (HL) }
   | instp.94  { INC (IX+d) }
   | instp.95  { INC (IY+d) }
   | instp.96  { DEC r }
   | instp.97  { DEC (HL) }
   | instp.98  { DEC (IX+d) }
   | instp.99  { DEC (IY+d) }
   | instp.102 { NEG }
   )
   chgalu8out1p=1;
endif

if ( instp.62  { SUB r }
   | instp.63  { SUB n }
   | instp.64  { SUB (HL) }
   | instp.65  { SUB (IX+d) }
   | instp.66  { SUB (IY+d) }
   | instp.87  { CP r }
   | instp.88  { CP n }
   | instp.89  { CP (HL) }
   | instp.90  { CP (IX+d) }
   | instp.91  { CP (IY+d) }
   | instp.190 { INI }
   | instp.191 { IND }
   | instp.192 { INIR }
   | instp.193 { INDR }
   | instp.194 { OUTI }
   | instp.195 { OUTD }
   | instp.196 { OTIR }
   | instp.197 { OTDR }
   )
   chgalu8out2p=1;
endif

if ( instp.57 { ADC A,r }
   | instp.58 { ADC A,n }
   | instp.59 { ADC A,(HL) }
   | instp.60 { ADC A,(IX+d) }
   | instp.61 { ADC A,(IY+d) }
   )
   chgalu8out3p=1;
endif

if ( instp.67 { SBC A,r }
   | instp.68 { SBC A,n }
   | instp.69 { SBC A,(HL) }
   | instp.70 { SBC A,(IX+d) }
   | instp.71 { SBC A,(IY+d) }
   )
```

```
   chgalu8out4p=1;
endif

if ( instp.72 { AND r }
   | instp.73 { AND n }
   | instp.74 { AND (HL) }
   | instp.75 { AND (IX[IY]+d) }
   | instp.156 { BIT b,r }
   | instp.157 { BIT b,(HL) }
   | instp.158 { BIT b,(IX+d) }
   | instp.159 { BIT b,(IY+d) }
   | instp.164 { RES b,r }
   | instp.165 { RES b,(HL) }
   | instp.166 { RES b,(IX+d) }
   | instp.167 { RES b,(IY+d) }
   )
   chgalu8out5p=1;
endif

if ( instp.77 { OR r }
   | instp.78 { OR n }
   | instp.79 { OR (HL) }
   | instp.80 { OR (IX+d) }
   | instp.81 { OR (IY+d) }
   | instp.160  { SET b,r }
   | instp.161  { SET b,(HL) }
   | instp.162  { SET b,(IX+d) }
   | instp.163  { SET b,(IY+d) }
   | instp.188  { IN r,(C) }
   )
   chgalu8out6p=1;
endif

if ( instp.82 { XOR r }
   | instp.83 { XOR n }
   | instp.84 { XOR (HL) }
   | instp.85 { XOR (IX+d) }
   | instp.86 { XOR (IY+d) }
   )
   chgalu8out7p=1;
endif

if ( instp.122 { RLCA }
   | instp.126 { RLC r }
   | instp.127 { RLC (HL) }
   | instp.128 { RLC (IX+d) }
   | instp.129 { RLC (IY+d) }
   )
   chgalu8out8p=1;
endif

if ( instp.123 { RLA }
   | instp.130 { RL r }
   | instp.131 { RL (HL) }
   | instp.132 { RL (IX+d) }
   | instp.133 { RL (IY+d) }
   )
   chgalu8out9p=1;
endif

if ( instp.134 { RRC r }
   | instp.135 { RRC (HL) }
```

```
   | instp.136 { RRC (IX+d) }
   | instp.137 { RRC (IY+d) }
   | instp.124 { RRCA }
   )
   chgalu8out10p=1;
endif

if ( instp.138 { RR r }
   | instp.139 { RR (HL) }
   | instp.140 { RR (IX+d) }
   | instp.141 { RR (IY+d) }
   | instp.125 { RRA }
   )
   chgalu8out11p=1;
endif

if ( instp.142 { SLA r }
   | instp.143 { SLA (HL) }
   | instp.144 { SLA (IX+d) }
   | instp.145 { SLA (IY+d) }
   )
   chgalu8out12p=1;
endif

if ( instp.146 { SRL r }
   | instp.147 { SRL (HL) }
   | instp.148 { SRL (IX+d) }
   | instp.149 { SRL (IY+d) }
   )
   chgalu8out13p=1;
endif

if ( instp.150 { SRA r }
   | instp.151 { SRA (HL) }
   | instp.152 { SRA (IX+d) }
   | instp.153 { SRA (IY+d) }
   )
   chgalu8out14p=1;
endif

if (instp.154) chgalu8out15p=1; endif { RLD }
if (instp.155) chgalu8out16p=1; endif { RRD }

if (chgalu8out1p) alu8out=alu8p; endif
if (chgalu8out2p) alu8out=alu8m; endif
if (chgalu8out3p) alu8out=alu8pc; endif
if (chgalu8out4p) alu8out=alu8mc; endif
if (chgalu8out5p) alu8out=alu8and; endif
if (chgalu8out6p) alu8out=alu8or; endif
if (chgalu8out7p) alu8out=alu8xor; endif
if (chgalu8out8p) alu8out=alu8ls; endif
if (chgalu8out9p) alu8out=alu8lsc; endif
if (chgalu8out10p) alu8out=alu8rs; endif
if (chgalu8out11p) alu8out=alu8rsc; endif
if (chgalu8out12p) alu8out=alu8lsz; endif
if (chgalu8out13p) alu8out=alu8rsz; endif
if (chgalu8out14p) alu8out=alu8rss; endif

if (chgalu8out15p)
   alu8out.0:3=reg0T.4:7;
   alu8out.4:7=regA.4:7;
endif

if (chgalu8out16p)
```

```
      alu8out.0:3=reg0T.0:3;
      alu8out.4:7=regA.4:7;
   endif

{ ------------------------------------------------------------------------ }
{    8 ビット演算器入力 a                                                  }
{ ------------------------------------------------------------------------ }
   if ( instp.52   { ADD A,r }
      | instp.53   { ADD A,n }
      | instp.54   { ADD A,(HL) }
      | instp.55   { ADD A,(IX+d) }
      | instp.56   { ADD A,(IY+d) }
      | instp.57   { ADC A,r }
      | instp.58   { ADC A,n }
      | instp.59   { ADC A,(HL) }
      | instp.60   { ADC A,(IX+d) }
      | instp.61   { ADC A,(IY+d) }
      | instp.62   { SUB r }
      | instp.63   { SUB n }
      | instp.64   { SUB (HL) }
      | instp.65   { SUB (IX+d) }
      | instp.66   { SUB (IY+d) }
      | instp.67   { SBC A,r }
      | instp.68   { SBC A,n }
      | instp.69   { SBC A,(HL) }
      | instp.70   { SBC A,(IX+d) }
      | instp.71   { SBC A,(IY+d) }
      | instp.72   { AND r }
      | instp.73   { AND n }
      | instp.74   { AND (HL) }
      | instp.75   { AND (IX[IY]+d) }
      | instp.77   { OR r }
      | instp.78   { OR n }
      | instp.79   { OR (HL) }
      | instp.80   { OR (IX+d) }
      | instp.81   { OR (IY+d) }
      | instp.82   { XOR r }
      | instp.83   { XOR n }
      | instp.84   { XOR (HL) }
      | instp.85   { XOR (IX+d) }
      | instp.86   { XOR (IY+d) }
      | instp.87   { CP r }
      | instp.88   { CP n }
      | instp.89   { CP (HL) }
      | instp.90   { CP (IX+d) }
      | instp.91   { CP (IY+d) }
      | instp.101  { CPL }
      | instp.122  { RLCA }
      | instp.123  { RLA }
      | instp.124  { RRCA }
      | instp.125  { RRA }
      )
      chgalu8ina1p=1;
   endif

   if ( instp.92   { INC r }
      | instp.93   { INC (HL) }
      | instp.94   { INC (IX+d) }
      | instp.95   { INC (IY+d) }
      )
      chgalu8ina2p=1;
```

```
endif

if ( instp.96   { DEC r }
   | instp.97   { DEC (HL) }
   | instp.98   { DEC (IX+d) }
   | instp.99   { DEC (IY+d) }
   )
   chgalu8ina3p=1;
endif

if ( instp.126  { RLC r }
   | instp.130  { RL r }
   | instp.134  { RRC r }
   | instp.138  { RR r }
   | instp.142  { SLA r }
   | instp.146  { SRL r }
   | instp.150  { SRA r }
   | instp.156  { BIT b,r }
   | instp.160  { SET b,r }
   | instp.164  { RES b,r }
   )
   chgalu8ina4p=1;
endif

if ( instp.127  { RLC (HL) }
   | instp.128  { RLC (IX+d) }
   | instp.129  { RLC (IY+d) }
   | instp.131  { RL (HL) }
   | instp.132  { RL (IX+d) }
   | instp.133  { RL (IY+d) }
   | instp.135  { RRC (HL) }
   | instp.136  { RRC (IX+d) }
   | instp.137  { RRC (IY+d) }
   | instp.139  { RR (HL) }
   | instp.140  { RR (IX+d) }
   | instp.141  { RR (IY+d) }
   | instp.143  { SLA (HL) }
   | instp.144  { SLA (IX+d) }
   | instp.145  { SLA (IY+d) }
   | instp.147  { SRL (HL) }
   | instp.148  { SRL (IX+d) }
   | instp.149  { SRL (IY+d) }
   | instp.151  { SRA (HL) }
   | instp.152  { SRA (IX+d) }
   | instp.153  { SRA (IY+d) }
   | instp.161  { SET b,(HL) }
   | instp.162  { SET b,(IX+d) }
   | instp.163  { SET b,(IY+d) }
   | instp.165  { RES b,(HL) }
   | instp.166  { RES b,(IX+d) }
   | instp.167  { RES b,(IY+d) }
   )
   chgalu8ina5p=1;
endif

if ( instp.157  { BIT b,(HL) }
   | instp.158  { BIT b,(IX+d) }
   | instp.159  { BIT b,(IY+d) }
   )
   chgalu8ina6p=1;
endif
```

```
   if (instp.188) chgalu8ina7p=1; endif { IN r,(C) }

   if ( instp.190  { INI }
      | instp.191  { IND }
      | instp.192  { INIR }
      | instp.193  { INDR }
      | instp.194  { OUTI }
      | instp.195  { OUTD }
      | instp.196  { OTIR }
      | instp.197  { OTDR }
      )
      chgalu8ina8p=1;
   endif

   if (instp.102)  { NEG }
      chgalu8ina9p=1;
   endif

   if (chgalu8ina1p) alu8ina.0:7=regA; endif
   if (chgalu8ina2p) alu8ina.0:7=1; endif
   if (chgalu8ina3p) alu8ina.0:7=0xff; endif

   if (chgalu8ina4p)
      switch(code0d.0:2)
         case 0: alu8ina.0:7=regB;
         case 1: alu8ina.0:7=regC;
         case 2: alu8ina.0:7=regD;
         case 3: alu8ina.0:7=regE;
         case 4: alu8ina.0:7=regH;
         case 5: alu8ina.0:7=regL;
         case 7: alu8ina.0:7=regA;
      endswitch
   endif

   if (chgalu8ina5p) alu8ina.0:7=reg0T; endif
   if (chgalu8ina6p) alu8ina.0:7=DATAIN; endif
   if (chgalu8ina7p) alu8ina.0:7=0; endif
   if (chgalu8ina8p) alu8ina.0:7=regB; endif
   if (chgalu8ina9p) alu8ina.0:7=!regA; endif

{ ------------------------------------------------------------------- }
{    8 ビット演算器入力 b                                            }
{ ------------------------------------------------------------------- }
   if ( instp.52           { ADD A,r }
      | instp.57           { ADC A,r }
      | instp.62           { SUB r }
      | instp.67           { SBC A,r }
      | instp.72           { AND r }
      | instp.77           { OR r }
      | instp.82           { XOR r }
      | instp.87           { CP r }
      )
      chgalu8inb1p=1;
   endif

   if ( instp.53           { ADD A,n }
      | instp.54           { ADD A,(HL) }
      | instp.55           { ADD A,(IX+d) }
      | instp.56           { ADD A,(IY+d) }
      | instp.58           { ADC A,n }
      | instp.59           { ADC A,(HL) }
      | instp.60           { ADC A,(IX+d) }
      | instp.61           { ADC A,(IY+d) }
```

```
    | instp.63          { SUB n }
    | instp.64          { SUB (HL) }
    | instp.65          { SUB (IX+d) }
    | instp.66          { SUB (IY+d) }
    | instp.68          { SBC A,n }
    | instp.69          { SBC A,(HL) }
    | instp.70          { SBC A,(IX+d) }
    | instp.71          { SBC A,(IY+d) }
    | instp.73          { AND n }
    | instp.74          { AND (HL) }
    | instp.75          { AND (IX[IY]+d) }
    | instp.78          { OR n }
    | instp.79          { OR (HL) }
    | instp.80          { OR (IX+d) }
    | instp.81          { OR (IY+d) }
    | instp.83          { XOR n }
    | instp.84          { XOR (HL) }
    | instp.85          { XOR (IX+d) }
    | instp.86          { XOR (IY+d) }
    | instp.88          { CP n }
    | instp.89          { CP (HL) }
    | instp.90          { CP (IX+d) }
    | instp.91          { CP (IY+d) }
    | instp.188         { IN r,(C) }
    )
    chgalu8inb2p=1;
endif

if ( instp.92           { INC r }
    | instp.96          { DEC r }
    )
    chgalu8inb3p=1;
endif

if ( instp.93           { INC (HL) }
    | instp.94          { INC (IX+d) }
    | instp.95          { INC (IY+d) }
    | instp.97          { DEC (HL) }
    | instp.98          { DEC (IX+d) }
    | instp.99          { DEC (IY+d) }
    )
    chgalu8inb4p=1;
endif

if ( instp.102          { NEG }
    | instp.190         { INI }
    | instp.191         { IND }
    | instp.192         { INIR }
    | instp.193         { INDR }
    | instp.194         { OUTI }
    | instp.195         { OUTD }
    | instp.196         { OTIR }
    | instp.197         { OTDR }
    )
    chgalu8inb5p=1;
endif

if ( instp.156          { BIT b,r }
    | instp.157         { BIT b,(HL) }
    | instp.158         { BIT b,(IX+d) }
    | instp.159         { BIT b,(IY+d) }
    | instp.160         { SET b,r }
```

```
   | instp.161          { SET b,(HL) }
   | instp.162          { SET b,(IX+d) }
   | instp.163          { SET b,(IY+d) }
   )
   chgalu8inb6p=1;
endif

if ( instp.164          { RES b,r }
   | instp.165          { RES b,(HL) }
   | instp.166          { RES b,(IX+d) }
   | instp.167          { RES b,(IY+d) }
   )
   chgalu8inb7p=1;
endif

if (chgalu8inb1p)
   switch(code0d.0:2)
      case 0: alu8inb.0:7=regB;
      case 1: alu8inb.0:7=regC;
      case 2: alu8inb.0:7=regD;
      case 3: alu8inb.0:7=regE;
      case 4: alu8inb.0:7=regH;
      case 5: alu8inb.0:7=regL;
      case 7: alu8inb.0:7=regA;
   endswitch
endif

if (chgalu8inb2p)
   alu8inb.0:7=DATAIN;
endif

if (chgalu8inb3p)
   switch(code0d.3:5)
      case 0: alu8inb.0:7=regB;
      case 1: alu8inb.0:7=regC;
      case 2: alu8inb.0:7=regD;
      case 3: alu8inb.0:7=regE;
      case 4: alu8inb.0:7=regH;
      case 5: alu8inb.0:7=regL;
      case 7: alu8inb.0:7=regA;
   endswitch
endif

if (chgalu8inb4p)
   alu8inb.0:7=reg0T;
endif

if (chgalu8inb5p)
   alu8inb.0:7=1;
endif

if (chgalu8inb6p)
   switch(code0d.3:5)
      case 0: alu8inb.0:7=1;
      case 1: alu8inb.0:7=2;
      case 2: alu8inb.0:7=4;
      case 3: alu8inb.0:7=8;
      case 4: alu8inb.0:7=16;
      case 5: alu8inb.0:7=32;
      case 6: alu8inb.0:7=64;
      case 7: alu8inb.0:7=128;
   endswitch
endif

if (chgalu8inb7p)
```

```
      switch(code0d.3:5)
         case 0: alu8inb.0:7=0b11111110;
         case 1: alu8inb.0:7=0b11111101;
         case 2: alu8inb.0:7=0b11111011;
         case 3: alu8inb.0:7=0b11110111;
         case 4: alu8inb.0:7=0b11101111;
         case 5: alu8inb.0:7=0b11011111;
         case 6: alu8inb.0:7=0b10111111;
         case 7: alu8inb.0:7=0b01111111;
      endswitch
   endif

{ ------------------------------------------------------------------- }
{    命令検出                                                        }
{ ------------------------------------------------------------------- }
   switch(DATAIN.0:2)
      case 6:
      default:
         switch(DATAIN.3:7)
            case 0x00: ancode125p=1;          { RLC r }
            case 0x01: ancode133p=1;          { RRC r }
            case 0x02: ancode129p=1;          { RL r }
            case 0x03: ancode137p=1;          { RR r }
            case 0x04: ancode141p=1;          { SLA r }
            case 0x05: ancode149p=1;          { SRA r }
            case 0x07: ancode145p=1;          { SRL r }
            case 0x10: ancode51p=1;           { ADD A,r }
            case 0x11: ancode56p=1;           { ADC A,r }
            case 0x12: ancode61p=1;           { SUB r }
            case 0x13: ancode66p=1;           { SBC A,r }
            case 0x14: ancode71p=1;           { AND r }
            case 0x16: ancode76p=1;           { OR r }
            case 0x15: ancode81p=1;           { XOR r }
            case 0x17: ancode86p=1;           { CP r }
         endswitch
   endswitch

   if (DATAIN.3:7==0xe) ancode5p=1; endif     { LD (IX[IY]+d),r }

   switch(DATAIN.6:7)
      case 0:
         switch(DATAIN.3:5)
            case 6:
            default:
            switch(DATAIN.0:2)
               case 4: ancode91p=1;           { INC r }
               case 5: ancode95p=1;           { DEC r }
               case 6: ancode0p=1;            { LD r,n }
            endswitch
         endswitch

         switch(DATAIN.0:3)
            case 1: ancode19p=1;            { LD dd,n'n }
            case 3: ancode115p=1;           { INC ss }
            case 9: ancode110p=1;           { ADD HL,ss }
                    ancode113p=1;           { ADD IX,pp }
                    ancode114p=1;           { ADD IY,rr }
            case 11: ancode118p=1;          { DEC ss }
         endswitch
      case 1:
         switch(DATAIN.0:2)
            case 0:
               switch(DATAIN.3:5)
                  case 6:
```

```
            default:
               ancode187p=1;              { IN r,(C) }

         endswitch
      case 1:
         switch(DATAIN.3:5)
            case 6:
            default:
               ancode188p=1;              { OUT (C),r }
         endswitch
      case 6:
         ancode156p=1;                    { BIT b,(HL) }
         ancode157p=1;                    { BIT b,(IX+d) }
         ancode158p=1;                    { BIT b,(IY+d) }
   endswitch

   switch(DATAIN.0:2)
      case 6:
      default:
         ancode155p=1;                    { BIT b,r }
   endswitch

   switch(DATAIN.0:2)
      case 6:
         switch(DATAIN.3:5)
            case 6:
            default:
              ancode2p=1;                 { LD r,(HL) }
              ancode6p=1;                 { LD r,(IX[IY]+d) }
         endswitch
   endswitch

   switch(DATAIN.0:2)
      case 6:
      default:
         if (DATAIN.3:5==6) ancode3p=1; endif             { LD (HL),r }
   endswitch

   if ((DATAIN.3:5!=6)&(DATAIN.0:2!=6)) ancode1p=1; endif { LD ra,rb }

   switch(DATAIN.0:3)
      case 2: ancode112p=1;                               { SBC HL,ss }
      case 3: ancode23p=1;                                { LD (n'n),dd }
      case 10: ancode111p=1;                              { ADC HL,ss }
      case 11: ancode22p=1;                               { LD dd,(n'n) }
   endswitch
case 2:
   switch(DATAIN.0:2)
      case 6:
         ancode164p=1;                                    { RES b,(HL) }
         ancode165p=1;                                    { RES b,(IX+d) }
         ancode166p=1;                                    { RES b,(IY+d) }
      default: ancode163p=1;                              { RES b,r }
   endswitch

case 3:
   switch(DATAIN.0:3) case 5: ancode31p=1; endswitch      { PUSH qq }
   switch(DATAIN.0:3) case 1: ancode32p=1; endswitch      { POP  qq }
   switch(DATAIN.0:2)
      case 0: ancode181p=1;                               { RET cc }
      case 2: ancode168p=1;                               { JP cc,n'n }
      case 4: ancode179p=1;                               { CALL cc,n'n }
      case 7: ancode184p=1;                               { RST p }
   endswitch
```

```
      switch(DATAIN.0:2)
         case 6:
            ancode160p=1;                                { SET b,(HL) }
            ancode161p=1;                                { SET b,(IX+d) }
            ancode162p=1;                                { SET b,(IY+d) }
         default:
            ancode159p=1;                                { SET b,r }
      endswitch
endswitch

switch(DATAIN)
   case 0xdd: z80code0p=1;
   case 0xfd: z80code1p=1;
   case 0xed: z80code2p=1;
   case 0xcb: z80code3p=1;
   case 0x21: ancode4p=1;                          { LD IX[IY],n'n }
   case 0x36: ancode7p=1;                          { LD (HL),n }
   case 0x36: ancode8p=1;                          { LD (IX[IY]+d),n }
   case 0x0a: ancode9p=1;                          { LD A,(BC) }
   case 0x02: ancode10p=1;                         { LD (BC),A }
   case 0x1a: ancode11p=1;                         { LD A,(DE) }
   case 0x12: ancode12p=1;                         { LD (DE),A }
   case 0x3a: ancode13p=1;                         { LD A,(n'n) }
   case 0x32: ancode14p=1;                         { LD (n'n),A }
   case 0x57: ancode15p=1;                         { LD A,I }
   case 0x47: ancode16p=1;                         { LD I,A }
   case 0x5f: ancode17p=1;                         { LD A,R }
   case 0x4f: ancode18p=1;                         { LD R,A }
   case 0x2a: ancode20p=1;                         { LD HL,(n'n) }
   case 0x22: ancode21p=1;                         { LD (n'n),HL }
   case 0x2a: ancode24p=1;                         { LD IX,(n'n) }
   case 0x22: ancode25p=1;                         { LD (n'n),IX }
   case 0x2a: ancode26p=1;                         { LD IY,(n'n) }
   case 0x22: ancode27p=1;                         { LD (n'n),IY }
   case 0xf9: ancode28p=1;                         { LD SP,HL }
   case 0xf9: ancode29p=1;                         { LD SP,IX }
   case 0xf9: ancode30p=1;                         { LD SP,IY }
   case 0xe5: ancode33p=1;                         { PUSH IX }
   case 0xe5: ancode34p=1;                         { PUSH IY }
   case 0xe1: ancode35p=1;                         { POP IX }
   case 0xe1: ancode36p=1;                         { POP IY }
   case 0xeb: ancode37p=1;                         { EX DE,HL }
   case 0x08: ancode38p=1;                         { EX AF,AF' }
   case 0xd9: ancode39p=1;                         { EXX }
   case 0xe3: ancode40p=1;                         { EX (SP),HL }
   case 0xe3: ancode41p=1;                         { EX (SP),IX }
   case 0xe3: ancode42p=1;                         { EX (SP),IY }
   case 0xa0: ancode43p=1;                         { LDI }
   case 0xa8: ancode44p=1;                         { LDD }
   case 0xB0: ancode45p=1;                         { LDIR }
   case 0xB8: ancode46p=1;                         { LDDR }
   case 0xa1: ancode47p=1;                         { CPI }
   case 0xa9: ancode48p=1;                         { CPD }
   case 0xb1: ancode49p=1;                         { CPIR }
   case 0xb9: ancode50p=1;                         { CPDR }
   case 0xc6: ancode52p=1;                         { ADD A,n }
   case 0x86: ancode53p=1;                         { ADD A,(HL) }
   case 0x86: ancode54p=1;                         { ADD A,(IX+d) }
   case 0x86: ancode55p=1;                         { ADD A,(IY+d) }
   case 0xce: ancode57p=1;                         { ADC A,n }
   case 0x8e: ancode58p=1;                         { ADC A,(HL) }
```

```
case 0x8e: ancode59p=1;                 { ADC A,(IX+d) }
case 0x8e: ancode60p=1;                 { ADC A,(IY+d) }
case 0xd6: ancode62p=1;                 { SUB n }
case 0x96: ancode63p=1;                 { SUB (HL) }
case 0x96: ancode64p=1;                 { SUB (IX+d) }
case 0x96: ancode65p=1;                 { SUB (IY+d) }
case 0xde: ancode67p=1;                 { SBC A,n }
case 0x9e: ancode68p=1;                 { SBC A,(HL) }
case 0x9e: ancode69p=1;                 { SBC A,(IX+d) }
case 0x9e: ancode70p=1;                 { SBC A,(IY+d) }
case 0xe6: ancode72p=1;                 { AND n }
case 0xa6: ancode73p=1;                 { AND (HL) }
case 0xa6: ancode74p=1;                 { AND (IX+d) }
case 0xa6: ancode75p=1;                 { AND (IY+d) }
case 0xf6: ancode77p=1;                 { OR n }
case 0xb6: ancode78p=1;                 { OR (HL) }
case 0xb6: ancode79p=1;                 { OR (IX+d) }
case 0xb6: ancode80p=1;                 { OR (IY+d) }
case 0xee: ancode82p=1;                 { XOR n }
case 0xae: ancode83p=1;                 { XOR (HL) }
case 0xae: ancode84p=1;                 { XOR (IX+d) }
case 0xae: ancode85p=1;                 { XOR (IY+d) }
case 0xfe: ancode87p=1;                 { CP n }
case 0xbe: ancode88p=1;                 { CP (HL) }
case 0xbe: ancode89p=1;                 { CP (IX+d) }
case 0xbe: ancode90p=1;                 { CP (IY+d) }
case 0x34: ancode92p=1;                 { INC (HL) }
case 0x34: ancode93p=1;                 { INC (IX+d) }
case 0x34: ancode94p=1;                 { INC (IY+d) }
case 0x35: ancode96p=1;                 { DEC (HL) }
case 0x35: ancode97p=1;                 { DEC (IX+d) }
case 0x35: ancode98p=1;                 { DEC (IY+d) }
case 0x27: ancode99p=1;                 { DAA }
case 0x2f: ancode100p=1;                { CPL }
case 0x44: ancode101p=1;                { NEG }
case 0x3f: ancode102p=1;                { CCF }
case 0x37: ancode103p=1;                { SCF }
case 0x76: ancode104p=1;                { HALT }
case 0xfb: ancode105p=1;                { EI }
case 0xf3: ancode106p=1;                { DI }
case 0x46: ancode107p=1;                { IM 0 }
case 0x56: ancode108p=1;                { IM 1 }
case 0x5e: ancode109p=1;                { IM 2 }
case 0x23: ancode116p=1;                { INC IX }
case 0x23: ancode117p=1;                { INC IY }
case 0x2b: ancode119p=1;                { DEC IX }
case 0x2b: ancode120p=1;                { DEC IY }
case 0x07: ancode121p=1;                { RLCA }
case 0x17: ancode122p=1;                { RLA }
case 0x0f: ancode123p=1;                { RRCA }
case 0x1f: ancode124p=1;                { RRA }
case 0x06: ancode126p=1;                { RLC (HL) }
case 0x06: ancode127p=1;                { RLC (IX+d) }
case 0x06: ancode128p=1;                { RLC (IY+d) }
case 0x16: ancode130p=1;                { RL (HL) }
case 0x16: ancode131p=1;                { RL (IX+d) }
case 0x16: ancode132p=1;                { RL (IY+d) }
case 0x0e: ancode134p=1;                { RRC (HL) }
case 0x0e: ancode135p=1;                { RRC (IX+d) }
case 0x0e: ancode136p=1;                { RRC (IY+d) }
```

```
      case 0x1e: ancode138p=1;                    { RR (HL) }
      case 0x1e: ancode139p=1;                    { RR (IX+d) }
      case 0x1e: ancode140p=1;                    { RR (IY+d) }
      case 0x26: ancode142p=1;                    { SLA (HL) }
      case 0x26: ancode143p=1;                    { SLA (IX+d) }
      case 0x26: ancode144p=1;                    { SLA (IY+d) }
      case 0x3e: ancode146p=1;                    { SRL (HL) }
      case 0x3e: ancode147p=1;                    { SRL (IX+d) }
      case 0x3e: ancode148p=1;                    { SRL (IY+d) }
      case 0x2e: ancode150p=1;                    { SRA (HL) }
      case 0x2e: ancode151p=1;                    { SRA (IX+d) }
      case 0x2e: ancode152p=1;                    { SRA (IY+d) }
      case 0x6f: ancode153p=1;                    { RLD }
      case 0x67: ancode154p=1;                    { RRD }
      case 0xc3: ancode167p=1;                    { JP n'n }
      case 0x18: ancode169p=1;                    { JR e }
      case 0x38: ancode170p=1;                    { JR C,e }
      case 0x30: ancode171p=1;                    { JR NC,e }
      case 0x28: ancode172p=1;                    { JR Z,e }
      case 0x20: ancode173p=1;                    { JR NZ,e }
      case 0xe9: ancode174p=1;                    { JP (HL) }
      case 0xe9: ancode175p=1;                    { JP (IX) }
      case 0xe9: ancode176p=1;                    { JP (IY) }
      case 0x10: ancode177p=1;                    { DJNZ e }
      case 0xcd: ancode178p=1;                    { CALL n'n }
      case 0xc9: ancode180p=1;                    { RET }
      case 0x4d: ancode182p=1;                    { RETI }
      case 0x45: ancode183p=1;                    { RETN }
      case 0xdb: ancode185p=1;                    { IN A,(n) }
      case 0xd3: ancode186p=1;                    { OUT (n),A }
      case 0xa2: ancode189p=1;                    { INI }
      case 0xaa: ancode190p=1;                    { IND }
      case 0xb2: ancode191p=1;                    { INIR }
      case 0xba: ancode192p=1;                    { INDR }
      case 0xa3: ancode193p=1;                    { OUTI }
      case 0xab: ancode194p=1;                    { OUTD }
      case 0xb3: ancode195p=1;                    { OTIR }
      case 0xbb: ancode196p=1;                    { OTDR }
   endswitch

{ ------------------------------------------------------------------- }
{    拡張命令指標                                                     }
{ ------------------------------------------------------------------- }
   if (RESET)
      expinst=0;
   else
      if (!WAIT)
         if (anct0p)
            if (z80code0p) expinst=1; endif
            if (z80code1p) expinst=2; endif
            if (z80code2p) expinst=3; endif
            if (z80code3p) expinst=4; endif
            if ( !z80code0p
               & !z80code1p
               & !z80code2p
               & !z80code3p
               & !ancode175p
               & !ancode176p
               )
               expinst=expinst;
            endif
         else
```

```
            if (anct1p)
               expinst=0;
            else
               if (anct10p)
                  if (z80code3p)
                     switch(expinst)
                        case 1:  expinst=5;
                        case 2:  expinst=6;
                        default: expinst=expinst;
                     endswitch
                  else
                     expinst=expinst;
                  endif
               else
                  expinst=expinst;
               endif
            endif
         endif
      else
         expinst=expinst;
      endif
   endif

{ ------------------------------------------------------------------- }
{    命令解析 動作長                                    }
{ ------------------------------------------------------------------- }
   if (RESET)
      anct=0;
   else
      if (!WAIT)
         switch(anct)
            case 0:
               switch(nmiset.0,intset.0)
                  case 0,1:                              { INT }
                     switch(im)
                        case 0: anct=5;
                        case 1: anct=2;
                        case 2: anct=5;
                     endswitch
                  case 1,0: anct=2;                      { NMI }
                  case 1,1: anct=2;                      { NMI }
                  default:
                  if (ancode174p) anct=0; endif          { JP (HL) }

                  if (ancode1p                           { LD ra,rb }
                     |ancode2p                           { LD r,(HL) }
                     |ancode3p                           { LD (HL),r }
                     |ancode9p                           { LD A,(BC) }
                     |ancode10p                          { LD (BC),A }
                     |ancode11p                          { LD A,(DE) }
                     |ancode12p                          { LD (DE),A }
                     |ancode28p                          { LD SP,HL }
                     |ancode37p                          { EX DE,HL }
                     |ancode38p                          { EX AF,AF' }
                     |ancode39p                          { EXX }
                     |ancode51p                          { ADD A,r }
                     |ancode53p                          { ADD A,(HL) }
                     |ancode56p                          { ADC A,r }
                     |ancode58p                          { ADC A,(HL) }
                     |ancode61p                          { SUB r }
                     |ancode66p                          { SBC A,r }
                     |ancode71p                          { AND r }
                     |ancode76p                          { OR r }
                     |ancode81p                          { XOR r }
```

```
    |ancode86p                      { CP r }
    |ancode91p                      { INC r }
    |ancode95p                      { DEC r }
    |ancode99p                      { DAA }
    |ancode100p                     { CPL }
    |ancode102p                     { CCF }
    |ancode103p                     { SCF }
    |ancode104p                     { HALT }
    |ancode105p                     { EI }
    |ancode106p                     { DI }
    |ancode110p                     { ADD HL,ss }
    |ancode115p                     { INC ss }
    |ancode118p                     { DEC ss }
    |ancode121p                     { RLCA }
    |ancode122p                     { RLA }
    |ancode123p                     { RRCA }
    |ancode124p                     { RRA }
    |ancode169p                     { JR e }
    |ancode170p                     { JR C,e }
    |ancode171p                     { JR NC,e }
    |ancode172p                     { JR Z,e }
    |ancode173p                     { JR NZ,e }
    |ancode177p) anct=1; endif      { DJNZ e }

if (ancode0p                        { LD r,n }
    |ancode7p                       { LD (HL),n }
    |ancode31p                      { PUSH qq }
    |ancode32p                      { POP qq }
    |ancode52p                      { ADD A,n }
    |ancode57p                      { ADC A,n }
    |ancode62p                      { SUB n }
    |ancode63p                      { SUB (HL) }
    |ancode67p                      { SBC A,n }
    |ancode68p                      { SBC A,(HL) }
    |ancode72p                      { AND n }
    |ancode73p                      { AND (HL) }
    |ancode77p                      { OR n }
    |ancode78p                      { OR (HL) }
    |ancode82p                      { XOR n }
    |ancode83p                      { XOR (HL) }
    |ancode87p                      { CP n }
    |ancode88p                      { CP (HL) }
    |ancode92p                      { INC (HL) }
    |ancode96p                      { DEC (HL) }
    |ancode167p                     { JP n'n }
    |ancode168p                     { JP cc,n'n }
    |ancode184p                     { RST p }
    |ancode185p                     { IN A,(n) }
    |ancode186p                     { OUT (n),A }
    |ancode180p) anct=2; endif      { RET }

if (ancode13p                       { LD A,(n'n) }
    |ancode14p                      { LD (n'n),A }
    |ancode19p) anct=3; endif       { LD dd,n'n }

if (ancode20p                       { LD HL,(n'n) }
    |ancode21p                      { LD (n'n),HL }
    |ancode40p                      { EX (SP),HL }
    |ancode178p) anct=4; endif      { CALL n'n }

if (ancode179p) if (ccen) anct=4; else anct=0; endif endif  { CALL cc,n'n }
```

```
      if (ancode181p) if (ccen) anct=2; else anct=0; endif endif  { RET cc }
      if (z80code0p|z80code1p|z80code2p|z80code3p) anct=10; endif
   endswitch
case 10:
   switch(expinst)
      case 1:
         if (ancode175p) anct=0; endif { JP (IX) }

         if (ancode29p                 { LD SP,IX }
            |ancode113p                { ADD IX,pp }
            |ancode117p                { INC IX }
            |ancode119p) anct=1; endif { DEC IX }

         if (ancode5p                  { LD (IX+d),r }
            |ancode6p                  { LD r,(IX+d) }
            |ancode33p                 { PUSH IX }
            |ancode35p                 { POP IX }
            |ancode54p                 { ADD A,(IX+d) }
            |ancode59p                 { ADC A,(IX+d) }
            |ancode64p                 { SUB (IX+d) }
            |ancode69p                 { SBC (IX+d) }
            |ancode74p                 { AND (IX+d) }
            |ancode79p                 { OR (IX+d) }
            |ancode84p                 { XOR (IX+d) }
            |ancode89p) anct=2; endif  { CP (IX+d) }

         if (ancode4p                  { LD IX,n'n   }
            |ancode8p                  { LD (IX+d),n }
            |ancode93p                 { INC (IX+d) }
            |ancode97p) anct=3; endif  { DEC (IX+d) }

         if (ancode24p                 { LD IX,(n'n) }
            |ancode25p                 { LD (n'n),IX }
            |ancode41p) anct=4; endif  { EX (SP),IX }

         if (z80code3p) anct=11; endif { ビット操作命令 }
      case 2:
         if (ancode176p) anct=0; endif { JP (IY) }

         if (ancode30p                 { LD SP,IY }
            |ancode114p                { ADD IY,rr }
            |ancode117p                { INC IY }
            |ancode120p) anct=1; endif { DEC IY }

         if (ancode5p                  { LD (IY+d),r }
            |ancode6p                  { LD r,(IY+d) }
            |ancode34p                 { PUSH IY }
            |ancode36p                 { POP IY }
            |ancode55p                 { ADD A,(IY+d) }
            |ancode60p                 { ADC A,(IY+d) }
            |ancode65p                 { SUB (IY+d) }
            |ancode70p                 { SBC (IY+d) }
            |ancode75p                 { AND (IY+d) }
            |ancode80p                 { OR (IY+d) }
            |ancode85p                 { XOR (IY+d) }
            |ancode90p) anct=2; endif  { CP (IY+d) }

         if (ancode4p                  { LD IY,n'n   }
            |ancode8p                  { LD (IY+d),n }
            |ancode94p                 { INC (IY+d) }
            |ancode98p) anct=3; endif  { DEC (IY+d) }
```

```
   if (ancode26p                 { LD IY,(n'n) }
      |ancode27p                 { LD (n'n),IY }
      |ancode42p) anct=4; endif  { EX (SP),IY }

   if (z80code3p) anct=11; endif { ビット操作命令 }
case 3:
   if (ancode15p                 { LD A,I }
      |ancode16p                 { LD I,A }
      |ancode17p                 { LD A,R }
      |ancode18p                 { LD R,A }
      |ancode47p                 { CPI }
      |ancode48p                 { CPD }
      |ancode49p                 { CPIR }
      |ancode50p                 { CPDR }
      |ancode101p                { NEG }
      |ancode107p                { IM 0 }
      |ancode108p                { IM 1 }
      |ancode109p                { IM 2 }
      |ancode111p                { ADC HL,ss }
      |ancode112p                { SBC HL,ss }
      |ancode187p                { IN r,(C) }
      |ancode188p) anct=1; endif { OUT (C),r }

   if (ancode43p                 { LDI }
      |ancode44p                 { LDD }
      |ancode45p                 { LDIR }
      |ancode46p                 { LDDR }
      |ancode153p                { RLD }
      |ancode154p                { RRD }
      |ancode182p                { RETI }
      |ancode183p                { RETN }
      |ancode189p                { INI }
      |ancode190p                { IND }
      |ancode191p                { INIR }
      |ancode192p                { INDR }
      |ancode193p                { OUTI }
      |ancode194p                { OUTD }
      |ancode195p                { OTIR }
      |ancode196p) anct=2; endif { OTDR }

   if (ancode22p                 { LD dd,(n'n) }
      |ancode23p) anct=4; endif  { LD (n'n),dd }

case 4:
   if (ancode125p                { RLC r }
      |ancode129p                { RL r }
      |ancode133p                { RRC r }
      |ancode137p                { RR r }
      |ancode141p                { SLA r }
      |ancode145p                { SRL r }
      |ancode149p                { SRA r }
      |ancode155p                { BIT b,r }
      |ancode156p                { BIT b,(HL) }
      |ancode159p                { SET b,r }
      |ancode163p) anct=1; endif { RES b,r }

   if (ancode126p                { RLC (HL) }
      |ancode130p                { RL (HL) }
      |ancode134p                { RRC (HL) }
      |ancode138p                { RR (HL) }
      |ancode142p                { SLA (HL) }
```

```
                     |ancode146p                { SRL (HL) }
                     |ancode150p                { SRA (HL) }
                     |ancode160p                { SET b,(HL) }
                     |ancode164p) anct=2; endif { RES b,(HL) }

               case 5:
                  if (ancode157p) anct=1; endif { BIT b,(IX+d) }

                  if (ancode127p                { RLC (IX+d) }
                     |ancode131p                { RL (IX+d) }
                     |ancode135p                { RRC (IX+d) }
                     |ancode139p                { RR (IX+d) }
                     |ancode143p                { SLA (IX+d) }
                     |ancode147p                { SRL (IX+d) }
                     |ancode151p                { SRA (IX+d) }
                     |ancode161p                { SET b,(IX+d) }
                     |ancode165p) anct=2; endif { RES b,(IX+d) }

               case 6:
                  if (ancode158p) anct=1; endif { BIT b,(IY+d) }

                  if (ancode128p                { RLC (IY+d) }
                     |ancode132p                { RL (IY+d) }
                     |ancode136p                { RRC (IY+d) }
                     |ancode140p                { RR (IY+d) }
                     |ancode144p                { SLA (IY+d) }
                     |ancode148p                { SRL (IY+d) }
                     |ancode152p                { SRA (IY+d) }
                     |ancode162p                { SET b,(IY+d) }
                     |ancode166p) anct=2; endif { RES b,(IY+d) }
            endswitch
         default:
            switch(inst,anct)
               case 199,5:                      { INT }
                  switch(im)
                     case 0: if (ancode178p) anct=anct-1; else anct=2; endif
                     case 2: anct=anct-1;
                  endswitch
               case 192,1:chganct3p=1;            { INIR }
               case 193,1:chganct3p=1;            { INDR }
               case 196,1:chganct3p=1;            { OTIR }
               case 197,1:chganct3p=1;            { OTDR }
               case 46,1: chganct1p=1;            { LDIR }
               case 47,1: chganct1p=1;            { LDDR }
               case 50,1: chganct2p=1;            { CPIR }
               case 51,1: chganct2p=1;            { CPDR }
               default:
                  if (anct==0) anct=anct; else anct=anct-1; endif
            endswitch
      endswitch
   else
      anct=anct;
   endif
endif

if (chganct1p)
   switch(regB)
      case 0:
         switch(regC)
            case 1:  anct=0;
            default: anct=2;
         endswitch
      default:
```

```
            anct=2;
      endswitch
   endif

   if (chganct2p)
      switch(regB)
         case 0:
            switch(regC)
               case 1:  anct=0;
               default:
                  if (regA==DATAIN)
                     anct=0;
                  else
                     anct=1;
                  endif
            endswitch
         default: anct=1;
      endswitch
   endif

   if (chganct3p)
      switch(regB)
         case 1:  anct=0;
         default: anct=2;
      endswitch
   endif

{ ------------------------------------------------------------------- }
{    命令長指標                                                       }
{ ------------------------------------------------------------------- }
   switch(anct)
      case 0:  anct0p=1;
      case 1:  anct1p=1;
      case 2:  anct2p=1;
      case 3:  anct3p=1;
      case 4:  anct4p=1;
      case 5:  anct5p=1;
      case 10: anct10p=1;
      case 11: anct11p=1;
   endswitch

{ ------------------------------------------------------------------- }
{    命令解析 命令長                                                  }
{ ------------------------------------------------------------------- }
   if (RESET)
      codesize=0;
   else
      if (!WAIT)
         if ((anct0p&!nmiset.0&!intset.0)|anct10p)
            switch(codesize)
               case 0:
                  if (ancode174p                    { JP (HL) }
                     |ancode181p) codesize=0; endif { RET cc }

                  if (ancode1p                      { LD ra,rb }
                     |ancode2p                      { LD r,(HL) }
                     |ancode3p                      { LD (HL),r }
                     |ancode9p                      { LD A,(BC) }
                     |ancode10p                     { LD (BC),A }
                     |ancode11p                     { LD A,(DE) }
                     |ancode12p                     { LD (DE),A }
                     |ancode28p                     { LD SP,HL }
                     |ancode31p                     { PUSH qq }
                     |ancode32p                     { POP qq }
```

```
    |ancode37p                          { EX DE,HL }
    |ancode38p                          { EX AF,AF' }
    |ancode39p                          { EXX }
    |ancode40p                          { EX (SP),HL }
    |ancode51p                          { ADD A,r }
    |ancode53p                          { ADD A,(HL) }
    |ancode56p                          { ADC A,r }
    |ancode58p                          { ADC A,(HL) }
    |ancode59p                          { ADC A,(HL) }
    |ancode61p                          { SUB r }
    |ancode63p                          { SUB (HL) }
    |ancode66p                          { SBC A,r }
    |ancode68p                          { SBC A,(HL) }
    |ancode71p                          { AND r }
    |ancode73p                          { AND (HL) }
    |ancode76p                          { OR r }
    |ancode78p                          { OR (HL) }
    |ancode81p                          { XOR r }
    |ancode83p                          { XOR (HL) }
    |ancode86p                          { CP r }
    |ancode88p                          { CP (HL) }
    |ancode91p                          { INC r }
    |ancode92p                          { INC (HL) }
    |ancode95p                          { DEC r }
    |ancode96p                          { DEC (HL) }
    |ancode99p                          { DAA }
    |ancode100p                         { CPL }
    |ancode102p                         { CCF }
    |ancode103p                         { SCF }
    |ancode104p                         { HALT }
    |ancode105p                         { EI }
    |ancode106p                         { DI }
    |ancode110p                         { ADD HL,ss }
    |ancode115p                         { INC ss }
    |ancode118p                         { DEC ss }
    |ancode121p                         { RLCA }
    |ancode122p                         { RLA }
    |ancode123p                         { RRCA }
    |ancode124p                         { RRA }
    |ancode169p                         { JR e }
    |ancode170p                         { JR C,e }
    |ancode171p                         { JR NC,e }
    |ancode172p                         { JR Z,e }
    |ancode173p                         { JR NZ,e }
    |ancode177p                         { DJNZ e }
    |ancode180p                         { RET }
    |ancode184p) codesize=1; endif      { RST p }

if (ancode0p                            { LD r,n }
    |ancode7p                           { LD (HL),n }
    |ancode52p                          { ADD A,n }
    |ancode57p                          { ADC A,n }
    |ancode62p                          { SUB n }
    |ancode67p                          { SBC A,n }
    |ancode72p                          { AND n }
    |ancode77p                          { OR n }
    |ancode82p                          { XOR n }
    |ancode87p                          { CP n }
    |ancode167p                         { JP n'n }
    |ancode168p                         { JP cc,n'n }
```

```
      |ancode185p                      { IN A,(n) }
      |ancode186p) codesize=2; endif   { OUT (n),A }

   if (ancode13p                       { LD A,(n'n) }
      |ancode14p                       { LD (n'n),A }
      |ancode19p                       { LD dd,n'n }
      |ancode20p                       { LD HL,(n'n) }
      |ancode21p                       { LD (n'n),HL }
      |ancode178p) codesize=3; endif   { CALL n'n }

   if (ancode179p) if (ccen) codesize=3; else codesize=0; endif endif { CALL cc,n'n }
   if (z80code0p|z80code1p|z80code2p|z80code3p) codesize=10; endif

case 10:
   switch(expinst)
      case 1:
         if (ancode175p) codesize=0; endif { JP (IX) }

         if (ancode29p                     { LD SP,IX }
            |ancode33p                     { PUSH IX }
            |ancode35p                     { POP IX }
            |ancode41p                     { EX (SP),IX }
            |ancode113p                    { ADD IX,pp }
            |ancode116p                    { INC IX }
            |ancode119p) codesize=1; endif { DEC IX }

         if (ancode5p                     { LD (IX+d),r }
            |ancode6p                     { LD r,(IX+d) }
            |ancode54p                    { ADD A,(IX+d) }
            |ancode59p                    { ADC A,(IX+d) }
            |ancode64p                    { SUB (IX+d) }
            |ancode69p                    { SBC (IX+d) }
            |ancode74p                    { AND (IX+d) }
            |ancode79p                    { OR (IX+d) }
            |ancode84p                    { XOR (IX+d) }
            |ancode89p                    { CP (IX+d) }
            |ancode93p                    { INC (IX+d) }
            |ancode97p) codesize=2; endif { DEC (IX+d) }

         if (ancode4p                     { LD IX,n'n   }
            |ancode8p                     { LD (IX+d),n }
            |ancode24p                    { LD IX,(n'n) }
            |ancode25p) codesize=3; endif { LD (n'n),IX }

         if (z80code3p) codesize=11; endif { ビット操作命令 }

      case 2:
         if (ancode176p) codesize=0; endif { JP (IY) }

         if (ancode30p                     { LD SP,IY }
            |ancode34p                     { PUSH IY }
            |ancode36p                     { POP IY }
            |ancode42p                     { EX (SP),IY }
            |ancode114p                    { ADD IY,rr }
            |ancode117p                    { INC IY }
            |ancode120p) codesize=1; endif { DEC IY }

         if (ancode5p                     { LD (IY+d),r }
            |ancode6p                     { LD r,(IY+d) }
            |ancode55p                    { ADD A,(IY+d) }
            |ancode60p                    { ADC A,(IY+d) }
            |ancode65p                    { SUB (IY+d) }
```

```
      |ancode70p                    { SBC (IY+d) }
      |ancode75p                    { AND (IY+d) }
      |ancode80p                    { OR (IY+d) }
      |ancode85p                    { XOR (IY+d) }
      |ancode90p                    { CP (IY+d) }
      |ancode94p                    { INC (IY+d) }
      |ancode98p) codesize=2; endif { DEC (IY+d) }

   if (ancode4p                     { LD IY,n'n   }
      |ancode8p                     { LD (IY+d),n }
      |ancode26p                    { LD IY,(n'n) }
      |ancode27p) codesize=3; endif { LD (n'n),IY }

   if (z80code3p) codesize=11; endif { ビット操作命令 }

case 3:
   if (ancode15p                     { LD A,I }
      |ancode16p                     { LD I,A }
      |ancode17p                     { LD A,R }
      |ancode18p                     { LD R,A }
      |ancode43p                     { LDI }
      |ancode44p                     { LDD }
      |ancode45p                     { LDIR }
      |ancode46p                     { LDDR }
      |ancode47p                     { CPI }
      |ancode48p                     { CPD }
      |ancode49p                     { CPIR }
      |ancode50p                     { CPDR }
      |ancode101p                    { NEG }
      |ancode107p                    { IM 0 }
      |ancode108p                    { IM 1 }
      |ancode109p                    { IM 2 }
      |ancode111p                    { ADC HL,ss }
      |ancode112p                    { SBC HL,ss }
      |ancode153p                    { RLD }
      |ancode154p                    { RRD }
      |ancode182p                    { RETI }
      |ancode183p                    { RETN }
      |ancode187p                    { IN r,(C) }
      |ancode188p                    { OUT (C),r }
      |ancode189p                    { INI }
      |ancode190p                    { IND }
      |ancode191p                    { INIR }
      |ancode192p                    { INDR }
      |ancode193p                    { OUTI }
      |ancode194p                    { OUTD }
      |ancode195p                    { OTIR }
      |ancode196p) codesize=1; endif { OTDR }

   if (ancode22p                     { LD dd,(n'n) }
      |ancode23p) codesize=3; endif  { LD (n'n),dd }

case 4:
   if (ancode125p                    { RLC r }
      |ancode126p                    { RLC (HL) }
      |ancode129p                    { RL r }
      |ancode130p                    { RL (HL) }
      |ancode133p                    { RRC r }
      |ancode134p                    { RRC (HL) }
      |ancode137p                    { RR r }
      |ancode138p                    { RR (HL) }
```

```
                        |ancode141p                    { SLA r }
                        |ancode142p                    { SLA (HL) }
                        |ancode145p                    { SRL r }
                        |ancode146p                    { SRL (HL) }
                        |ancode149p                    { SRA r }
                        |ancode150p                    { SRA (HL) }
                        |ancode155p                    { BIT b,r }
                        |ancode156p                    { BIT b,(HL) }
                        |ancode159p                    { SET b,r }
                        |ancode160p                    { SET b,(HL) }
                        |ancode163p                    { RES b,r }
                        |ancode164p) codesize=1; endif { RES b,(HL) }

                  case 5:
                     if (ancode127p                    { RLC (IX+d) }
                        |ancode131p                    { RL (IX+d) }
                        |ancode135p                    { RRC (IX+d) }
                        |ancode139p                    { RR (IX+d) }
                        |ancode143p                    { SLA (IX+d) }
                        |ancode147p                    { SRL (IX+d) }
                        |ancode151p                    { SRA (IX+d) }
                        |ancode157p                    { BIT b,(IX+d) }
                        |ancode161p                    { SET b,(IX+d) }
                        |ancode165p) codesize=1; endif { RES b,(IX+d) }

                  case 6:
                     if (ancode128p                    { RLC (IY+d) }
                        |ancode132p                    { RL (IY+d) }
                        |ancode136p                    { RRC (IY+d) }
                        |ancode140p                    { RR (IY+d) }
                        |ancode144p                    { SLA (IY+d) }
                        |ancode148p                    { SRL (IY+d) }
                        |ancode152p                    { SRA (IY+d) }
                        |ancode158p                    { BIT b,(IY+d) }
                        |ancode162p                    { SET b,(IY+d) }
                        |ancode166p) codesize=1; endif { RES b,(IY+d) }
               endswitch
            endswitch
         else
            if (codesize==0)
               codesize=codesize;
            else
               codesize=codesize-1;
            endif
         endif
      else
         codesize=codesize;
      endif
   endif

{ ------------------------------------------------------------------- }
{    命令解析 命令番号                                  }
{ ------------------------------------------------------------------- }
   if (!WAIT)
      if ((anct0p&!nmiset.0&!intset.0)|anct10p)
         switch(expinst)
            case 0:
               if (ancode0p)  inst=1; endif    { LD r,n }
               if (ancode1p)  inst=2; endif    { LD ra,rb }
               if (ancode2p)  inst=3; endif    { LD r,(HL) }
               if (ancode3p)  inst=4; endif    { LD (HL),r }
               if (ancode7p)  inst=6; endif    { LD (HL),n }
```

```
if (ancode9p)  inst=10; endif   { LD A,(BC) }
if (ancode10p) inst=11; endif   { LD (BC),A }
if (ancode11p) inst=12; endif   { LD A,(DE) }
if (ancode12p) inst=13; endif   { LD (DE),A }
if (ancode13p) inst=14; endif   { LD A,(n'n) }
if (ancode14p) inst=15; endif   { LD (n'n),A }
if (ancode19p) inst=20; endif   { LD dd,n'n }
if (ancode20p) inst=21; endif   { LD HL,(n'n) }
if (ancode21p) inst=22; endif   { LD (n'n),HL }
if (ancode28p) inst=29; endif   { LD SP,HL }
if (ancode31p) inst=32; endif   { PUSH qq }
if (ancode32p) inst=33; endif   { POP qq }
if (ancode37p) inst=38; endif   { EX DE,HL }
if (ancode38p) inst=39; endif   { EX AF,AF' }
if (ancode39p) inst=40; endif   { EXX }
if (ancode40p) inst=41; endif   { EX (SP),HL }
if (ancode51p) inst=52; endif   { ADD A,r }
if (ancode52p) inst=53; endif   { ADD A,n }
if (ancode53p) inst=54; endif   { ADD A,(HL) }
if (ancode56p) inst=57; endif   { ADC A,r }
if (ancode57p) inst=58; endif   { ADC A,n }
if (ancode58p) inst=59; endif   { ADC A,(HL) }
if (ancode61p) inst=62; endif   { SUB r }
if (ancode62p) inst=63; endif   { SUB n }
if (ancode63p) inst=64; endif   { SUB (HL) }
if (ancode66p) inst=67; endif   { SBC A,r }
if (ancode67p) inst=68; endif   { SBC A,n }
if (ancode68p) inst=69; endif   { SBC A,(HL) }
if (ancode71p) inst=72; endif   { AND r }
if (ancode72p) inst=73; endif   { AND n }
if (ancode73p) inst=74; endif   { AND (HL) }
if (ancode76p) inst=77; endif   { OR r }
if (ancode77p) inst=78; endif   { OR n }
if (ancode78p) inst=79; endif   { OR (HL) }
if (ancode81p) inst=82; endif   { XOR r }
if (ancode82p) inst=83; endif   { XOR n }
if (ancode83p) inst=84; endif   { XOR (HL) }
if (ancode86p) inst=87; endif   { CP r }
if (ancode87p) inst=88; endif   { CP n }
if (ancode88p) inst=89; endif   { CP (HL) }
if (ancode91p) inst=92; endif   { INC r }
if (ancode92p) inst=93; endif   { INC (HL) }
if (ancode95p) inst=96; endif   { DEC r }
if (ancode96p) inst=97; endif   { DEC (HL) }
if (ancode99p) inst=100; endif  { DAA }
if (ancode100p) inst=101; endif { CPL }
if (ancode102p) inst=103; endif { CCF }
if (ancode103p) inst=104; endif { SCF }
if (ancode104p) inst=105; endif { HALT }
if (ancode105p) inst=106; endif { EI }
if (ancode106p) inst=107; endif { DI }
if (ancode110p) inst=111; endif { ADD HL,ss }
if (ancode115p) inst=116; endif { INC ss }
if (ancode118p) inst=119; endif { DEC ss }
if (ancode121p) inst=122; endif { RLCA }
if (ancode122p) inst=123; endif { RLA }
if (ancode123p) inst=124; endif { RRCA }
if (ancode124p) inst=125; endif { RRA }
if (ancode167p) inst=168; endif { JP n'n }
if (ancode168p) inst=169; endif { JP cc,n'n }
```

```
   if (ancode169p) inst=170; endif { JR e }
   if (ancode170p) inst=171; endif { JR C,e }
   if (ancode171p) inst=172; endif { JR NC,e }
   if (ancode172p) inst=173; endif { JR Z,e }
   if (ancode173p) inst=174; endif { JR NZ,e }
   if (ancode174p) inst=175; endif { JP (HL) }
   if (ancode177p) inst=178; endif { DJNZ e }
   if (ancode178p) inst=179; endif { CALL n'n }
   if (ancode179p) inst=180; endif { CALL cc,n'n }
   if (ancode180p) inst=181; endif { RET }
   if (ancode181p) inst=182; endif { RET cc }
   if (ancode184p) inst=185; endif { RST p }
   if (ancode185p) inst=186; endif { IN A,(n) }
   if (ancode186p) inst=187; endif { OUT (n),A }
case 1:
   if (ancode4p)  inst=5; endif   { LD IX,n'n   }
   if (ancode6p)  inst=7; endif   { LD r,(IX+d) }
   if (ancode5p)  inst=8; endif   { LD (IX+d),r }
   if (ancode8p)  inst=9; endif   { LD (IX+d),n }
   if (ancode24p) inst=25; endif  { LD IX,(n'n) }
   if (ancode25p) inst=26; endif  { LD (n'n),IX }
   if (ancode29p) inst=30; endif  { LD SP,IX }
   if (ancode33p) inst=34; endif  { PUSH IX }
   if (ancode35p) inst=36; endif  { POP IX }
   if (ancode41p) inst=42; endif  { EX (SP),IX }
   if (ancode54p) inst=55; endif  { ADD A,(IX+d) }
   if (ancode59p) inst=60; endif  { ADC A,(IX+d) }
   if (ancode64p) inst=65; endif  { SUB (IX+d) }
   if (ancode69p) inst=70; endif  { SBC A,(IX+d) }
   if (ancode74p) inst=75; endif  { AND (IX+d) }
   if (ancode79p) inst=80; endif  { OR (IX+d) }
   if (ancode84p) inst=85; endif  { XOR (IX+d) }
   if (ancode89p) inst=90; endif  { CP (IX+d) }
   if (ancode93p) inst=94; endif  { INC (IX+d) }
   if (ancode97p) inst=98; endif  { DEC (IX+d) }
   if (ancode113p) inst=114; endif { ADD IX,pp }
   if (ancode116p) inst=117; endif { INC IX }
   if (ancode119p) inst=120; endif { DEC IX }
   if (ancode175p) inst=176; endif { JP (IX) }
case 2:
   if (ancode4p)  inst=5; endif   { LD IY,n'n   }
   if (ancode6p)  inst=7; endif   { LD r,(IY+d) }
   if (ancode5p)  inst=8; endif   { LD (IY+d),r }
   if (ancode8p)  inst=9; endif   { LD (IY+d),n }
   if (ancode26p) inst=27; endif  { LD IY,(n'n) }
   if (ancode27p) inst=28; endif  { LD (n'n),IY }
   if (ancode30p) inst=31; endif  { LD SP,IY }
   if (ancode34p) inst=35; endif  { PUSH IY }
   if (ancode36p) inst=37; endif  { POP IY }
   if (ancode42p) inst=43; endif  { EX (SP),IY }
   if (ancode55p) inst=56; endif  { ADD A,(IY+d) }
   if (ancode60p) inst=61; endif  { ADC A,(IY+d) }
   if (ancode65p) inst=66; endif  { SUB (IY+d) }
   if (ancode70p) inst=71; endif  { SBC A,(IY+d) }
   if (ancode75p) inst=75; endif  { AND (IY+d) }
   if (ancode80p) inst=81; endif  { OR (IY+d) }
   if (ancode85p) inst=86; endif  { XOR (IY+d) }
   if (ancode90p) inst=91; endif  { CP (IY+d) }
   if (ancode94p) inst=95; endif  { INC (IY+d) }
   if (ancode98p) inst=99; endif  { DEC (IY+d) }
   if (ancode114p) inst=115; endif  { ADD IY,rr }
```

```
   if (ancode117p) inst=118; endif  { INC IY }
   if (ancode120p) inst=121; endif  { DEC IY }
   if (ancode176p) inst=177; endif  { JP (IY) }
case 3:
   if (ancode15p) inst=16; endif  { LD A,I }
   if (ancode16p) inst=17; endif  { LD I,A }
   if (ancode17p) inst=18; endif  { LD A,R }
   if (ancode18p) inst=19; endif  { LD R,A }
   if (ancode22p) inst=23; endif  { LD dd,(n'n) }
   if (ancode23p) inst=24; endif  { LD (n'n),dd }
   if (ancode43p) inst=44; endif  { LDI }
   if (ancode44p) inst=45; endif  { LDD }
   if (ancode45p) inst=46; endif  { LDIR }
   if (ancode46p) inst=47; endif  { LDDR }
   if (ancode47p) inst=48; endif  { CPI }
   if (ancode48p) inst=49; endif  { CPD }
   if (ancode49p) inst=50; endif  { CPIR }
   if (ancode50p) inst=51; endif  { CPDR }
   if (ancode101p) inst=102; endif { NEG }
   if (ancode107p) inst=108; endif { IM 0 }
   if (ancode108p) inst=109; endif { IM 1 }
   if (ancode109p) inst=110; endif { IM 2 }
   if (ancode111p) inst=112; endif { ADC HL,ss }
   if (ancode112p) inst=113; endif { SBC HL,ss }
   if (ancode153p) inst=154; endif { RLD }
   if (ancode154p) inst=155; endif { RRD }
   if (ancode182p) inst=183; endif { RETI }
   if (ancode183p) inst=184; endif { RETN }
   if (ancode187p) inst=188; endif { IN r,(C) }
   if (ancode188p) inst=189; endif { OUT (C),r }
   if (ancode189p) inst=190; endif { INI }
   if (ancode190p) inst=191; endif { IND }
   if (ancode191p) inst=192; endif { INIR }
   if (ancode192p) inst=193; endif { INDR }
   if (ancode193p) inst=194; endif { OUTI }
   if (ancode194p) inst=195; endif { OUTD }
   if (ancode195p) inst=196; endif { OTIR }
   if (ancode196p) inst=197; endif { OTDR }
case 4:
   if (ancode125p) inst=126; endif { RLC r }
   if (ancode126p) inst=127; endif { RLC (HL) }
   if (ancode129p) inst=130; endif { RL r }
   if (ancode130p) inst=131; endif { RL (HL) }
   if (ancode133p) inst=134; endif { RRC r }
   if (ancode134p) inst=135; endif { RRC (HL) }
   if (ancode137p) inst=138; endif { RR r }
   if (ancode138p) inst=139; endif { RR (HL) }
   if (ancode141p) inst=142; endif { SLA r }
   if (ancode142p) inst=143; endif { SLA (HL) }
   if (ancode145p) inst=146; endif { SRL r }
   if (ancode146p) inst=147; endif { SRL (HL) }
   if (ancode149p) inst=150; endif { SRA r }
   if (ancode150p) inst=151; endif { SRA (HL) }
   if (ancode155p) inst=156; endif { BIT b,r }
   if (ancode156p) inst=157; endif { BIT b,(HL) }
   if (ancode159p) inst=160; endif { SET b,r }
   if (ancode160p) inst=161; endif { SET b,(HL) }
   if (ancode163p) inst=164; endif { RES b,r }
   if (ancode164p) inst=165; endif { RES b,(HL) }
case 5:
   if (ancode127p) inst=128; endif  { RLC (IX+d) }
```

```
                if (ancode131p) inst=132; endif  { RL (IX+d) }
                if (ancode135p) inst=136; endif  { RRC (IX+d) }
                if (ancode139p) inst=140; endif  { RR (IX+d) }
                if (ancode143p) inst=144; endif  { SLA (IX+d) }
                if (ancode147p) inst=148; endif  { SRL (IX+d) }
                if (ancode151p) inst=152; endif  { SRA (IX+d) }
                if (ancode157p) inst=158; endif  { BIT b,(IX+d) }
                if (ancode161p) inst=162; endif  { SET b,(IX+d) }
                if (ancode165p) inst=166; endif  { RES b,(IX+d) }
             case 6:
                if (ancode128p) inst=129; endif  { RLC (IY+d) }
                if (ancode132p) inst=133; endif  { RL (IY+d) }
                if (ancode136p) inst=137; endif  { RRC (IY+d) }
                if (ancode140p) inst=141; endif  { RR (IY+d) }
                if (ancode144p) inst=145; endif  { SLA (IY+d) }
                if (ancode148p) inst=149; endif  { SRL (IY+d) }
                if (ancode152p) inst=153; endif  { SRA (IY+d) }
                if (ancode158p) inst=159; endif  { BIT b,(IY+d) }
                if (ancode162p) inst=163; endif  { SET b,(IY+d) }
                if (ancode166p) inst=167; endif  { RES b,(IY+d) }
          endswitch
       else
          switch(anct,nmiset.0,intset.0)
             case 0,1,0: inst=198;                { NMI }
             case 0,1,1: inst=198;                { NMI }
             case 0,0,1: inst=199;                { INT }
             default:  inst=inst;
          endswitch
       endif
    else
       inst=inst;
    endif

{ ------------------------------------------------------------------------ }
{    命令指標                                                  }
{ ------------------------------------------------------------------------ }
    switch(inst)
       case 1:   instp.1=1;    { LD r,n }
       case 2:   instp.2=1;    { LD ra,rb }
       case 3:   instp.3=1;    { LD r,(HL) }
       case 4:   instp.4=1;    { LD (HL),r }
       case 5:   instp.5=1;    { LD IX[IY],n'n   }
       case 6:   instp.6=1;    { LD (HL),n }
       case 7:   instp.7=1;    { LD r,(IX[IY]+d) }
       case 8:   instp.8=1;    { LD (IX[IY]+d),r }
       case 9:   instp.9=1;    { LD (IX[IY]+d),n }
       case 10:  instp.10=1;   { LD A,(BC) }
       case 11:  instp.11=1;   { LD (BC),A }
       case 12:  instp.12=1;   { LD A,(DE) }
       case 13:  instp.13=1;   { LD (DE),A }
       case 14:  instp.14=1;   { LD A,(n'n) }
       case 15:  instp.15=1;   { LD (n'n),A }
       case 16:  instp.16=1;   { LD A,I }
       case 17:  instp.17=1;   { LD I,A }
       case 18:  instp.18=1;   { LD A,R }
       case 19:  instp.19=1;   { LD R,A }
       case 20:  instp.20=1;   { LD dd,n'n }
       case 21:  instp.21=1;   { LD HL,(n'n) }
       case 22:  instp.22=1;   { LD (n'n),HL }
       case 23:  instp.23=1;   { LD dd,(n'n) }
       case 24:  instp.24=1;   { LD (n'n),dd }
       case 25:  instp.25=1;   { LD IX,(n'n) }
```

```
case 26:  instp.26=1;   { LD (n'n),IX }
case 27:  instp.27=1;   { LD IY,(n'n) }
case 28:  instp.28=1;   { LD (n'n),IY }
case 29:  instp.29=1;   { LD SP,HL }
case 30:  instp.30=1;   { LD SP,IX }
case 31:  instp.31=1;   { LD SP,IY }
case 32:  instp.32=1;   { PUSH qq }
case 33:  instp.33=1;   { POP qq }
case 34:  instp.34=1;   { PUSH IX }
case 35:  instp.35=1;   { PUSH IY }
case 36:  instp.36=1;   { POP IX }
case 37:  instp.37=1;   { POP IY }
case 38:  instp.38=1;   { EX DE,HL }
case 39:  instp.39=1;   { EX AF,AF' }
case 40:  instp.40=1;   { EXX }
case 41:  instp.41=1;   { EX (SP),HL }
case 42:  instp.42=1;   { EX (SP),IX }
case 43:  instp.43=1;   { EX (SP),IY }
case 44:  instp.44=1;   { LDI }
case 45:  instp.45=1;   { LDD }
case 46:  instp.46=1;   { LDIR }
case 47:  instp.47=1;   { LDDR }
case 48:  instp.48=1;   { CPI }
case 49:  instp.49=1;   { CPD }
case 50:  instp.50=1;   { CPIR }
case 51:  instp.51=1;   { CPDR }
case 52:  instp.52=1;   { ADD A,r }
case 53:  instp.53=1;   { ADD A,n }
case 54:  instp.54=1;   { ADD A,(HL) }
case 55:  instp.55=1;   { ADD A,(IX+d) }
case 56:  instp.56=1;   { ADD A,(IY+d) }
case 57:  instp.57=1;   { ADC A,r }
case 58:  instp.58=1;   { ADC A,n }
case 59:  instp.59=1;   { ADC A,(HL) }
case 60:  instp.60=1;   { ADC A,(IX+d) }
case 61:  instp.61=1;   { ADC A,(IY+d) }
case 62:  instp.62=1;   { SUB r }
case 63:  instp.63=1;   { SUB n }
case 64:  instp.64=1;   { SUB (HL) }
case 65:  instp.65=1;   { SUB (IX+d) }
case 66:  instp.66=1;   { SUB (IY+d) }
case 67:  instp.67=1;   { SBC A,r }
case 68:  instp.68=1;   { SBC A,n }
case 69:  instp.69=1;   { SBC A,(HL) }
case 70:  instp.70=1;   { SBC A,(IX+d) }
case 71:  instp.71=1;   { SBC A,(IY+d) }
case 72:  instp.72=1;   { AND r }
case 73:  instp.73=1;   { AND n }
case 74:  instp.74=1;   { AND (HL) }
case 75:  instp.75=1;   { AND (IX[IY]+d) }
case 76:  instp.76=1;
case 77:  instp.77=1;   { OR r }
case 78:  instp.78=1;   { OR n }
case 79:  instp.79=1;   { OR (HL) }
case 80:  instp.80=1;   { OR (IX+d) }
case 81:  instp.81=1;   { OR (IY+d) }
case 82:  instp.82=1;   { XOR r }
case 83:  instp.83=1;   { XOR n }
case 84:  instp.84=1;   { XOR (HL) }
case 85:  instp.85=1;   { XOR (IX+d) }
```

```
case 86:  instp.86=1;   { XOR (IY+d) }
case 87:  instp.87=1;   { CP r }
case 88:  instp.88=1;   { CP n }
case 89:  instp.89=1;   { CP (HL) }
case 90:  instp.90=1;   { CP (IX+d) }
case 91:  instp.91=1;   { CP (IY+d) }
case 92:  instp.92=1;   { INC r }
case 93:  instp.93=1;   { INC (HL) }
case 94:  instp.94=1;   { INC (IX+d) }
case 95:  instp.95=1;   { INC (IY+d) }
case 96:  instp.96=1;   { DEC r }
case 97:  instp.97=1;   { DEC (HL) }
case 98:  instp.98=1;   { DEC (IX+d) }
case 99:  instp.99=1;   { DEC (IY+d) }
case 100: instp.100=1;  { DAA }
case 101: instp.101=1;  { CPL }
case 102: instp.102=1;  { NEG }
case 103: instp.103=1;  { CCF }
case 104: instp.104=1;  { SCF }
case 105: instp.105=1;  { HALT }
case 106: instp.106=1;  { EI }
case 107: instp.107=1;  { DI }
case 108: instp.108=1;  { IM 0 }
case 109: instp.109=1;  { IM 1 }
case 110: instp.110=1;  { IM 2 }
case 111: instp.111=1;  { ADD HL,ss }
case 112: instp.112=1;  { ADC HL,ss }
case 113: instp.113=1;  { SBC HL,ss }
case 114: instp.114=1;  { ADD IX,pp }
case 115: instp.115=1;  { ADD IY,rr }
case 116: instp.116=1;  { INC ss }
case 117: instp.117=1;  { INC IX }
case 118: instp.118=1;  { INC IY }
case 119: instp.119=1;  { DEC ss }
case 120: instp.120=1;  { DEC IX }
case 121: instp.121=1;  { DEC IY }
case 122: instp.122=1;  { RLCA }
case 123: instp.123=1;  { RLA }
case 124: instp.124=1;  { RRCA }
case 125: instp.125=1;  { RRA }
case 126: instp.126=1;  { RLC r }
case 127: instp.127=1;  { RLC (HL) }
case 128: instp.128=1;  { RLC (IX+d) }
case 129: instp.129=1;  { RLC (IY+d) }
case 130: instp.130=1;  { RL r }
case 131: instp.131=1;  { RL (HL) }
case 132: instp.132=1;  { RL (IX+d) }
case 133: instp.133=1;  { RL (IY+d) }
case 134: instp.134=1;  { RRC r }
case 135: instp.135=1;  { RRC (HL) }
case 136: instp.136=1;  { RRC (IX+d) }
case 137: instp.137=1;  { RRC (IY+d) }
case 138: instp.138=1;  { RR r }
case 139: instp.139=1;  { RR (HL) }
case 140: instp.140=1;  { RR (IX+d) }
case 141: instp.141=1;  { RR (IY+d) }
case 142: instp.142=1;  { SLA r }
case 143: instp.143=1;  { SLA (HL) }
case 144: instp.144=1;  { SLA (IX+d) }
case 145: instp.145=1;  { SLA (IY+d) }
```

```
      case 146: instp.146=1;  { SRL r }
      case 147: instp.147=1;  { SRL (HL) }
      case 148: instp.148=1;  { SRL (IX+d) }
      case 149: instp.149=1;  { SRL (IY+d) }
      case 150: instp.150=1;  { SRA r }
      case 151: instp.151=1;  { SRA (HL) }
      case 152: instp.152=1;  { SRA (IX+d) }
      case 153: instp.153=1;  { SRA (IY+d) }
      case 154: instp.154=1;  { RLD }
      case 155: instp.155=1;  { RRD }
      case 156: instp.156=1;  { BIT b,r }
      case 157: instp.157=1;  { BIT b,(HL) }
      case 158: instp.158=1;  { BIT b,(IX+d) }
      case 159: instp.159=1;  { BIT b,(IY+d) }
      case 160: instp.160=1;  { SET b,r }
      case 161: instp.161=1;  { SET b,(HL) }
      case 162: instp.162=1;  { SET b,(IX+d) }
      case 163: instp.163=1;  { SET b,(IY+d) }
      case 164: instp.164=1;  { RES b,r }
      case 165: instp.165=1;  { RES b,(HL) }
      case 166: instp.166=1;  { RES b,(IX+d) }
      case 167: instp.167=1;  { RES b,(IY+d) }
      case 168: instp.168=1;  { JP n'n }
      case 169: instp.169=1;  { JP cc,n'n }
      case 170: instp.170=1;  { JR e }
      case 171: instp.171=1;  { JR C,e }
      case 172: instp.172=1;  { JR NC,e }
      case 173: instp.173=1;  { JR Z,e }
      case 174: instp.174=1;  { JR NZ,e }
      case 175: instp.175=1;  { JP (HL) }
      case 176: instp.176=1;  { JP (IX) }
      case 177: instp.177=1;  { JP (IY) }
      case 178: instp.178=1;  { DJNZ e }
      case 179: instp.179=1;  { CALL n'n }
      case 180: instp.180=1;  { CALL cc,n'n }
      case 181: instp.181=1;  { RET }
      case 182: instp.182=1;  { RET cc }
      case 183: instp.183=1;  { RETI }
      case 184: instp.184=1;  { RETN }
      case 185: instp.185=1;  { RST p }
      case 186: instp.186=1;  { IN A,(n) }
      case 187: instp.187=1;  { OUT (n),A }
      case 188: instp.188=1;  { IN r,(C) }
      case 189: instp.189=1;  { OUT (C),r }
      case 190: instp.190=1;  { INI }
      case 191: instp.191=1;  { IND }
      case 192: instp.192=1;  { INIR }
      case 193: instp.193=1;  { INDR }
      case 194: instp.194=1;  { OUTI }
      case 195: instp.195=1;  { OUTD }
      case 196: instp.196=1;  { OTIR }
      case 197: instp.197=1;  { OTDR }
      case 198: instp.198=1;  { NMI }
      case 199: instp.199=1;  { INT }
   endswitch

{ ------------------------------------------------------------------- }
{    命令記憶 0                                                      }
{ ------------------------------------------------------------------- }
   if (RESET)
      code0d=0;
```

```
   else
      if (!WAIT)
         if (anct0p|anct10p)
            if ((codesize==0)|(codesize==10))
               code0d=DATAIN;
            else
               code0d=code0d;
            endif
         else
            code0d=code0d;
         endif
      else
         code0d=code0d;
      endif
   endif

{ ------------------------------------------------------------------- }
{    命令記憶 1                                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
      code1d=0;
   else
      if (!WAIT)
         if (codesize==2)
            code1d=DATAIN;
         else
            code1d=code1d;
         endif
      else
         code1d=code1d;
      endif
   endif

{ ------------------------------------------------------------------- }
{    命令記憶 2                                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
      code2d=0;
   else
      if (!WAIT)
         if (codesize==3)
            code2d=DATAIN;
         else
            if (anct11p)
               code2d=DATAIN;
            else
               code2d=code2d;
            endif
         endif
      else
         code2d=code2d;
      endif
   endif

{ ------------------------------------------------------------------- }
{    命令記憶 3                                                       }
{ ------------------------------------------------------------------- }
   if (RESET)
      code3d=0;
   else
      if (!WAIT)
         if (codesize==4)
            code3d=DATAIN;
         else
            code3d=code3d;
```

```
         endif
      else
         code3d=code3d;
      endif
   endif

ende
```

```
 ========================================================================
    機能実行譜
 ========================================================================
entity sim
output RESET;
output DATAIN[8];
output ADDRESS[16];
output DATAOUT[8];
output DIR;
output MIO;
output NMI;
output INT;
output INTA;
output WAIT;
output MEM0P[8];
output MEM1P[8];
output MEM2P[8];
output MEM3P[8];
output MEM4P[8];
output MEM5P[8];
output MEM6P[8];
output MEM7P[8];
output IO0P[8];
output IO1P[8];
output IO2P[8];
output IO3P[8];

bitn   node_Q[27];

bitr   mem0[8];
bitr   mem1[8];
bitr   mem2[8];
bitr   mem3[8];
bitr   mem4[8];
bitr   mem5[8];
bitr   mem6[8];
bitr   mem7[8];
bitr   io0[8];
bitr   io1[8];
bitr   io2[8];
bitr   io3[8];
bitr   tc[10];

   MEM0P=mem0;
   MEM1P=mem1;
   MEM2P=mem2;
   MEM3P=mem3;
   MEM4P=mem4;
   MEM5P=mem5;
   MEM6P=mem6;
   MEM7P=mem7;

   IO0P=io0;
   IO1P=io1;
   IO2P=io2;
   IO3P=io3;

   ADDRESS=node_Q.0:15;
```

```
DATAOUT=node_Q.16:23;
DIR=node_Q.24;
MIO=node_Q.25;
INTA=node_Q.26;

part main(RESET,DATAIN,NMI,INT,WAIT,node_Q)

tc=tc+1;

if (tc<5) RESET=1; endif

if (!node_Q.26)
   if (!node_Q.25)
      preparation codegen(node_Q.0:15,DATAIN,test)
   endif
endif

if (!node_Q.24&!node_Q.25&!node_Q.26)
   switch(node_Q.0:15)
      case 0x1234: DATAIN=mem0;
      case 0x1235: DATAIN=mem1;
      case 0x1236: DATAIN=mem2;
      case 0x1237: DATAIN=mem3;
      case 0x5678: DATAIN=mem4;
      case 0x5679: DATAIN=mem5;
      case 0x567a: DATAIN=mem6;
      case 0x567b: DATAIN=mem7;
   endswitch
endif

if (!node_Q.24&node_Q.25&!node_Q.26)
   switch(node_Q.0:7)
      case 0x12: DATAIN=io0;
      case 0x13: DATAIN=io1;
      case 0x14: DATAIN=io2;
      case 0x15: DATAIN=io3;
   endswitch
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x1234: mem0=node_Q.16:23;
      default: mem0=mem0;
   endswitch
else
   mem0=mem0;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x1235: mem1=node_Q.16:23;
      default: mem1=mem1;
   endswitch
else
   mem1=mem1;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x1236: mem2=node_Q.16:23;
      default: mem2=mem2;
   endswitch
else
   mem2=mem2;
endif
```

```
if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x1237: mem3=node_Q.16:23;
      default: mem3=mem3;
   endswitch
else
   mem3=mem3;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x5678: mem4=node_Q.16:23;
      default: mem4=mem4;
   endswitch
else
   mem4=mem4;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x5679: mem5=node_Q.16:23;
      default: mem5=mem5;
   endswitch
else
   mem5=mem5;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x567a: mem6=node_Q.16:23;
      default: mem6=mem6;
   endswitch
else
   mem6=mem6;
endif

if (node_Q.24&!node_Q.25)
   switch(node_Q.0:15)
      case 0x567b: mem7=node_Q.16:23;
      default: mem7=mem7;
   endswitch
else
   mem7=mem7;
endif

if (node_Q.24&node_Q.25)
   switch(node_Q.0:7)
      case 0x12: io0=node_Q.16:23;
      default: io0=io0;
   endswitch
else
   io0=io0;
endif

if (node_Q.24&node_Q.25)
   switch(node_Q.0:7)
      case 0x13: io1=node_Q.16:23;
      default: io1=io1;
   endswitch
else
   io1=io1;
endif

if (node_Q.24&node_Q.25)
   switch(node_Q.0:7)
      case 0x14: io2=node_Q.16:23;
```

```
            default: io2=io2;
         endswitch
      else
         io2=io2;
      endif

      if (node_Q.24&node_Q.25)
         switch(node_Q.0:7)
            case 0x15: io3=node_Q.16:23;
            default: io3=io3;
         endswitch
      else
         io3=io3;
      endif


      switch(tc)
        case 30: INT=1;
        case 32: DATAIN=0xcd;
        case 33: DATAIN=0x38;
        case 34: DATAIN=0x00;
      endswitch

   ende

   endlogic
```