```
**************************************************************************
                                TABASCO
**************************************************************************
Job nr :     5 734
tilmeldt fra datbodil

Printerudskrift

printjob  30.4.1975  17.30.18

områdenavn :  sim8080

**********************************

begin
  comment the program simulates an intel 8080 processing unit;
  integer A,B,C,D,E,H,L,carry,ic,time,M,
          SP,PC,carryextra,temp,F,stackbottom,
          ownadr,bufadr,adr,coreadr,bufout,bufin,inword,char,
          action,k,x,n,m,i,val,arg,ress,bufdescr,first,last,r,
          max_adr,ao1,ao2,aosign,aozero,aoparity;
  integer array stack(0:7), op(0:255), tab(65:88), t(1:10),
                mes,mo,mi(1:10),indtabel(0:127);
  boolean sign,zero,parity,normal,exc,io,nl,sp,inte;
  boolean array core(0:1023);
  array name,a(1:2);

  procedure alarm(kind); integer kind;
  begin
    write(out,nl,1,(case kind of
              (<:** impossible:>,<:** address:>,<:** illegal:>,
               <:** instruction counter:>,<:** stack pointer:>)));
    setposition(out,0,0);
    goto interrupt
  end;

  procedure exam(m); integer m;
  begin integer i,c;
    comment the procedure sets the sign,zero and parity flip-flops
            according to the parameter;
    if m<0 then m:=m+256;
    if m>255 then m:=m-256;
    c:=m;
    zero:=c=0;
    sign:=c>127;
    r:=0;
    for i:=1 step 1 until 8 do
    begin
      r:=r+c extract 1; c:=c shift (-1)
    end;
    parity:=r mod 2=0
  end;

  procedure addi(a,b,c); integer a,b,c;
  begin
  a:=a+b+c;
  if a>255 then carry:=1 else carry:=0;
    exam(a)
  end addi;

  procedure sub(a,b,c); integer a,b,c;
  begin
  a:=a-b-c;
  if a<0 then carry:=1 else carry:=0;
    exam(a)
  end sub;
```

```
procedure annd(a,b);  value b; integer a,b;
begin
  a:=integerand(a,b); carry:=0; exam(a)
end;

procedure exor(a,b);  value b; integer a,b;
begin
  a:=integerexor(a,b);  carry:=0; exam(a)
end;

procedure orr(a,b);  value b; integer a,b;
begin
  a:=integeror(a,b); carry:=0; exam(a)
end;

procedure examdouble(m); integer m;
  begin
     L:=m extract 8;
     H:=(m shift (-8)) extract 8;
     carry:= if (m shift (-16))>0 then 1 else 0;
  end;

procedure doublecount(high,low,count);
integer high,low,count;
  begin integer t;
     t := high shift 8 + low + count;
     low := t extract 8;
     high := (t shift (-8)) extract 8;
  end;


procedure display(val); integer val;
begin integer char,class;
comment the procedure writes out value of val,if a new
        value is typed it is assigned to val;
  write(out,val,sp,2);
  setposition(out,0,0); setposition(in,0,0);
  class:=readchar(in,char);
  if class=2 then
  begin
    repeatchar(in); read(in,val)
  end;
end;

comment initialization of tables;

for i:=65 step 1 until 88 do
tab(i):=(case i-64 of
        (2,2,2,2,2,9,16,2,14,1,
         10,2,12,13,15,4,11,5,7,8,
         1,1,3,6));

for i:=0 step 1 until 255 do
begin
arg:=(case i+1 of
       (0,
       2,0,0,0,0,1,0,0,0,0,
       0,0,0,1,0,0,2,0,0,0,
       0,1,0,0,0,0,0,0,0,1,
       0,0,2,2,0,0,0,1,0,0,
       0,2,0,0,0,1,0,0,2,2,
       0,0,0,1,0,0,0,2,0,0,
       0,1,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,
```

```
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,2,2,2,0,1,0,0,
        0,2,0,2,2,1,0,0,0,2,
        1,2,0,1,0,0,0,2,1,2,
        0,1,0,0,0,2,0,2,0,1,
        0,0,0,2,0,2,0,1,0,0,
        0,2,0,2,0,1,0,0,0,2,
        0,2,0,1,0));
time:=(case i+1 of
       (1,
        3,2,1,1,1,2,1,1,3,2,
        1,1,1,2,1,1,3,2,1,1,
        1,2,1,1,3,2,1,1,1,2,
        1,1,3,5,1,1,1,2,1,1,
        3,5,1,1,1,2,1,1,3,4,
        1,3,3,3,1,1,3,4,1,1,
        1,2,1,1,1,1,1,1,1,2,
        1,1,1,1,1,1,1,2,1,1,
        1,1,1,1,1,2,1,1,1,1,
        1,1,1,2,1,1,1,1,1,1,
        1,2,1,1,1,1,1,1,1,2,
        1,2,2,2,2,2,2,1,2,1,
        1,1,1,1,1,2,1,1,1,1,
        1,1,1,2,1,1,1,1,1,1,
        1,2,1,1,1,1,1,1,1,2,
        1,1,1,1,1,1,1,2,1,1,
        1,1,1,1,1,2,1,1,1,1,
        1,1,1,2,1,1,1,1,1,1,
        1,2,1,1,1,1,1,1,1,2,
        1,3,3,3,3,5,3,2,3,3,
        3,3,1,5,5,2,3,3,3,3,
        3,5,3,2,3,3,1,3,3,5,
        1,2,3,3,3,3,0,5,3,2,
        3,3,1,3,1,5,1,2,3,3,
        3,3,1,5,3,2,3,3,1,3,
        1,5,1,2,3));
val:= case i+1 of
       (244,
        197,219,235,072,078,065,156,243,215,221,
        236,073,079,066,157,243,198,220,237,074,
        080,067,158,243,216,222,238,075,081,068,
        159,243,199,229,239,076,082,069,225,243,
        217,230,240,077,083,070,224,243,200,209,
        241,233,234,071,223,243,218,210,242,227,
        228,064,226,009,010,011,012,013,014,051,
        008,016,017,018,019,020,021,052,015,023,
        024,025,026,027,028,053,022,030,031,032,
        033,034,035,054,029,037,038,039,040,041,
        042,055,036,044,045,046,047,048,049,056,
        043,058,059,060,061,062,063,243,057,002,
        003,004,005,006,007,050,001,085,086,087,
        088,089,090,091,084,094,095,096,097,098,
        099,100,093,103,104,105,106,107,108,109,
        102,112,113,114,115,116,117,118,111,121,
        122,123,124,125,126,127,120,130,131,132,
        133,134,135,136,129,139,140,141,142,143,
        144,145,138,148,149,150,151,152,153,154,
```

```
        147,170,205,189,188,161,201,092,178,171,
        169,190,243,162,160,101,179,172,206,191,
        187,163,202,110,180,173,243,192,186,164,
        243,119,181,174,207,193,212,165,203,128,
        182,175,214,194,211,166,243,137,183,176,
        208,195,232,167,204,146,184,177,213,196,
        231,168,243,155,185);

  if arg=2 then time := time-2;
  op(i):=(time*4+arg) shift 8 add val
  end;

  comment initialization;

  coreadr:=first_addr(core);
  bufin:=bufout:=-1;
  nl:=false add 10;
  sp:=false add 32;
  exc:=normal:=false;
  max_edr:=1023;
  stackbottom:=50;
  for i:=0 step 1 until 127 do indtabel(i):=6 shift 12+i mod 16;
  intable(0);
interrupt:
      begin comment indlæsning;
         real array name(1:10);
         zone z(128,1,stderror);
         boolean field pp;
         real array firstsegment(1:128);
         integer i,j,words,bitno;

         write(out,<:<10>code:   :>);setposition(out,0,0);
         readstring(in,name,1); read(in,words);

         bitno:=-36; i:=1; pp:=1;
         open(z,4,string name(increase(i)),0);
         inrec(z,128);

         for i:=1 step 1 until 128 do firstsegment(i):=z(i);
         for i:=2 step 1 until firstsegment(1)+1 do
         for j:=firstsegment(i) shift (-24) extract 24 step 1
               until firstsegment(i) extract 24 do
         begin
              if bitno=-36 then inrec(z,1);
              core(j):=false add(z(1) shift bitno extract 8);
              bitno := if bitno = 0 then -36 else bitno+12;
              if j>words then alarm(3);
         end;
         exc:=false;
         close(z,true);
      end overførsel;


execute:;

  comment an instruction is executed either in normal or debugging
        mode;

  comment debugging mode;

  if exc then goto instr;
  setposition(in,0,0);
  write(out,<:<10> *:>); setposition(out,0,0);
  readchar(in,char);
  action:=if char>64 and char<89 then tab(char) else 1;
  case action of
```

```
  begin
    begin comment illegal;
      write(out,nl,1,<:**  illegal:>); goto execute
    end;

    begin comment display register;
      if char=65 then display(A) else if char=66 then display(B) else
      if char=67 then display(C) else if char=68 then display(D) else
      if char=69 then display(E) else if char=72 then display(H) else
      display(L)
    end;

    begin comment display word;
      read(in,n);
      if n<0 or n>maxadr then alarm(2);
      val:=core(n) extract 12;
      display(val);
      core(n):=false add (val extract 12)
    end;

    begin comment print words from n to m;
      read(in,n,m);
      if n<0 or m<0 or n>max_adr  or m>max_adr then alarm(2);
      if n>m then alarm(1);
      write(out,<: addr  :>,<<ddd>,n);
      for i:=n step 1 until m do
      begin
        if i mod 10=0 and i<>n then write(out,nl,1,<: addr :>,<<ddd>,i);
        write(out,<<ddd>,sp,2,core(i) extract 12)
      end
    end;

    begin comment reset registers;
      A:=B:=C:=D:=E:=H:=L:=0;
      SP:=1000;
      PC:=0;
      carry:=carryextra:=0;
      sign:=zero:=parity:=false;
      time:=0
    end;

    begin comment execute instructions until address x;
      read(in,x); exc:=true;
      if x>max_adr or x<0 then alarm(1);
instr:
      if PC>max_adr or PC<0 then alarm(4);
      if x=PC then
      begin
        if sign then aosign:=1 else aosign:=0;
        if zero then aozero:=1 else aozero:=0;
        if parity then aoparity:=1 else aoparity:=0;
        write(out,<: addr :>,<<ddddd>,PC,<:    reg:>,A,B,C,D,E,H,L,carry
>,
        carryextra,aosign,aozero,aoparity);
        exc:=false;
        goto execute
      end else
      begin comment an instruction is executed;
        ao1:=op(core(PC) extract 12);
        val:=ao1 shift(-8) extract 4;
        time:=time+val//4;
        arg:=val mod 4;
        if arg=2 then time:=time+2;
        val:=ao1 extract 8;
        case arg+1 of begin
          ;
```

```
   adr:=core(PC+1) extract 8;
   adr:=core(PC+2) extract 6 shift 8 add
                    (core(PC+1) extract 8);
end case arg+1;
M:=(0 add H shift 8 add L);
M:=(M+max_adr) mod max_adr;
PC:=PC+arg+1;
case val of
begin
comment register transfers;
   A:=A;
   A:=B;
   A:=C;
   A:=D;
   A:=E;
   A:=H;
   A:=L;
   B:=A;
   B:=B;
   B:=C;
   B:=D;
   B:=E;
   B:=H;
   B:=L;
   C:=A;
   C:=B;
   C:=C;
   C:=D;
   C:=E;
   C:=H;
   C:=L;
   D:=A;
   D:=B;
   D:=C;
   D:=D;
   D:=E;
   D:=H;
   D:=L;
   E:=A;
   E:=B;
   E:=C;
   E:=D;
   E:=E;
   E:=H;
   E:=L;
   H:=A;
   H:=B;
   H:=C;
   H:=D;
   H:=E;
   H:=H;
   H:=L;
   L:=A;
   L:=B;
   L:=C;
   L:=D;
   L:=E;
   L:=H;
   L:=L;
   A:=core(M) extract 8;
   B:=core(M) extract 8;
   C:=core(M) extract 8;
   D:=core(M) extract 8;
   E:=core(M) extract 8;
   H:=core(M) extract 8;
   L:=core(M) extract 8;
```

```
    core(M):=false add A;
    core(M):=false add B;
    core(M):=false add C;
    core(M):=false add D;
    core(M):=false add E;
    core(M):=false add H;
    core(M):=false add L;
    A:=adr;
    B:=adr;
    C:=adr;
    D:=adr;
    E:=adr;
    H:=adr;
    L:=adr;
    core(M):=false add adr;
  comment increase register;
    begin B:=(B+1) ; exam(B) end;
    begin C:=(C+1) ; exam(C) end;
    begin D:=(D+1) ; exam(D) end;
    begin E:=(E+1) ; exam(E) end;
    begin H:=(H+1) ; exam(H) end;
    begin L:=(L+1) ; exam(L) end;
  comment decrease register;
    begin B:=(B-1) ; exam(B) end;
    begin C:=(C-1) ; exam(C) end;
    begin D:=(D-1) ; exam(D) end;
    begin E:=(E-1) ; exam(E) end;
    begin H:=(H-1) ; exam(H) end;
    begin L:=(L-1) ; exam(L) end;
  comment add register to A;
    addi(A,A,0);
    addi(A,B,0);
    addi(A,C,0);
    addi(A,D,0);
    addi(A,E,0);
    addi(A,H,0);
    addi(A,L,0);
    addi(A,core(M) extract 8,0);
    addi(A,adr,0);
  comment add register with carry to A;
    addi(A,A,carry);
    addi(A,B,carry);
    addi(A,C,carry);
    addi(A,D,carry);
    addi(A,E,carry);
    addi(A,H,carry);
    addi(A,L,carry);
    addi(A,core(M) extract 8,carry);
    addi(A,adr,carry);
  comment subtract register from A;
    sub(A,A,0);
    sub(A,B,0);
    sub(A,C,0);
    sub(A,D,0);
    sub(A,E,0);
    sub(A,H,0);
    sub(A,L,0);
    sub(A,core(M) extract 8,0);
    sub(A,adr,0);
  comment subtract register with carry from A;
    sub(A,A,carry);
    sub(A,B,carry);
    sub(A,C,carry);
    sub(A,D,carry);
    sub(A,E,carry);
    sub(A,H,carry);
```

```
          sub(A,L,carry);
          sub(A,core(M) extract 8,carry);
          sub(A,adr,carry);
        comment logical instructions;
            annd(A,A);
            annd(A,B);
            annd(A,C);
            annd(A,D);
            annd(A,E);
            annd(A,H);
            annd(A,L);
            annd(A,core(M) extract 8);
            annd(A,adr);
            exor(A,A);
            exor(A,B);
            exor(A,C);
            exor(A,D);
            exor(A,E);
            exor(A,H);
            exor(A,L);
            exor(A,core(M) extract 8);
            exor(A,adr);
            orr(A,A);
            orr(A,B);
            orr(A,C);
            orr(A,D);
            orr(A,E);
            orr(A,H);
            orr(A,L);
            orr(A,core(M) extract 8);
            orr(A,adr);
        comment register with A;
          begin m:=A; sub(m,A,0) end;
          begin m:=A; sub(m,B,0) end;
          begin m:=A; sub(m,C,0) end;
          begin m:=A; sub(m,D,0) end;
          begin m:=A; sub(m,E,0) end;
          begin m:=A; sub(m,H,0) end;
          begin m:=A; sub(m,L,0) end;
          begin m:=A; sub(m,core(M) extract 8 ,0) end;
          begin m:=A; sub(m,adr,0) end;
        comment rotate (rlc,rrc,ral,rar);
            begin
              carry:=A shift(-7) extract 1;
              A:=(A shift 1) extract 8 add carry
            end;
            begin
              carry:=A extract 1;
              A:=carry shift 7 add (A extract 8 shift (-1))
            end;
            begin m:=carry; carry:=A shift(-7) extract 1;
                  A:=(A shift 1) extract 8 add (m extract 1)
            end;
            begin m:=carry; carry:=A extract 1;
                  A:=m shift 7 add (A extract 8 shift(-1))
            end;
call:     begin
            if SP < 2 then alarm(5);
            core(SP-1) := false add ( PC shift (-8));
            core(SP-2) := false add ( PC extract 8);
            SP := SP - 2;
            PC := adr;
          end;

            if -, zero then goto call;
            if zero then goto call;
```

```
            if carry = 0 then goto call;
            if carry = 1 then goto call;
            if -, parity then goto call;
            if parity then goto call;
            if -, sign then goto call;
            if sign then goto call;

ret:        begin
               PC := (core(SP+1) extract 8) shift 8  add
                     (core(SP) extract 8);
               SP := SP + 2;
            end;

            if -, zero then goto ret;
            if zero then goto ret;
            if carry = 0 then goto ret;
            if carry = 1 then goto ret;
            if -, parity then goto ret;
            if parity then goto ret;
            if -, sign then goto ret;
            if sign then goto ret;

            begin adr := 0; goto call end;
            begin adr := 8; goto call end;
            begin adr := 16; goto call end;
            begin adr := 24; goto call end;
            begin adr := 32; goto call end;
            begin adr := 40; goto call end;
            begin adr := 48; goto call end;
            begin adr := 56; goto call end;

            begin  comment only one port - input;
                  setposition(in,0,0);
                  readchar(in,A);
            end;

            begin  comment only one port - output;
                  write(out,false add A,1);
                  setposition(out,0,0);
            end;

            comment jump;
               PC := adr;
               if -, zero then PC := adr;
               if zero then PC := adr;
               if carry = 0 then PC := adr;
               if carry = 1 then PC := adr;
               if -, parity then PC := adr;
               if parity then PC := adr;
               if -, sign then PC := adr;
               if sign then PC := adr;

            begin comment (B,C) := adr;
                  C := core(PC-2) extract 8;
                  B := core(PC-1) extract 8;
            end;

            begin comment (D,E) := adr;
                  E := core(PC-2) extract 8;
                  D := core(PC-1) extract 8;
            end;

            begin comment (H,L) := adr;
                  L := core(PC-2) extract 8;
                  H := core(PC-1) extract 8;
            end;
```

```
SP := adr;

begin comment push B;
      if SP < 2 then alarm(5);
      core(SP-1) := false add B;
      core(SP-2) := false add C;
      SP := SP - 2;
end;

begin comment push D;
      if SP < 2 then alarm(5);
      core(SP-1) := false add D;
      core(SP-2) := false add E;
      SP := SP - 2;
end;

begin comment push H;
      if SP < 2 then alarm(5);
      core(SP-1) := false add H;
      core(SP-2) := false add L;
      SP := SP - 2;
end;

begin comment push register A and flip-flops;
      if SP < 2 then alarm(5);
      F := (parity extract 1) +
           ((sign extract 1) shift 1) +
           ((zero extract 1) shift 2) +
           (carryextra shift 3) +
           (carry shift 4);
      core(SP-1) := false add A;
      core(SP-2) := false add F;
      SP := SP - 2;
end;

begin comment pop B;
      C := core(SP) extract 8;
      B := core(SP+1) extract 8;
      SP := SP + 2;
end;

begin comment pop D;
      E := core(SP) extract 8;
      D := core(SP+1) extract 8;
      SP := SP + 2;
end;

begin comment pop H;
   L := core(SP) extract 8;
   H := core(SP+1) extract 8;
   SP := SP + 2;
end;

begin comment pop register A and flip-flops;
      F := core(SP) extract 8;
      parity := false add F;
      sign := false add (F shift (-1));
      zero := false add (F shift (-2));
      carryextra := (F extract 4) extract (-3);
      carry := (F extract 5) shift (-4);
      A := core(SP+1) extract 8;
      SP := SP + 2;
end;

comment sta; core(adr) := false add A;
```

```
comment lda; A := core(adr) extract 8;

begin comment xchg;
      temp := H; H := D; D := temp;
      temp := L; L := E; E := temp;
end;

begin comment xthl;
      temp := H;
      H := core(SP+1) extract 8;
      core(SP+1) := false add temp;
      temp := L;
      L := core(SP) extract 8;
      core(SP) := false add temp;
      comment because the representation is bad;;
      time;=time+5;
end;

SP := M;

PC := M;

begin comment dad B;
      temp := B shift 8 + C + M;
      examdouble(temp);
end;

begin comment dad D;
      temp := D shift 8 + E + M;
      examdouble(temp);
end;

begin comment dad H;
      temp := M + M;
      examdouble(temp);
end;

begin comment dad SP;
      temp := M + SP;
      examdouble(temp);
end;

 comment stax and ldax;
   core(B shift 8 + C) := false add A;
   core(D shift 8 + E) := false add A;
   A := core(B shift 8 + C) extract 8;
   A := core(D shift 8 + E) extract 8;

carry := 1;

A := (-1 - A) extract 8;

comment decimal adjust accumulator - not implemented;;

carry := if carry = 1 then 0 else 1;

begin A := A + 1; exam(A) end;
begin A := A - 1; exam(A) end;

begin core(adr):=false add L; core(adr+1):=false add H end;

begin
      L := core(adr) extract 8;
      H := core(adr+1) extract 8;
end;
```

```
      inte := true;
      inte := false;

      begin
            temp := core(adr) extract 8;
            temp := temp + 1;
            exam(temp);
            core(adr) := false add temp
      end;

      begin
            temp := core(adr) extract 8;
            temp := temp - 1;
            exam(temp);
            core(adr) := false add temp
      end;

      doublecount(B,C,  1 );
      doublecount(B,C, -1 );
      doublecount(D,E,  1 );
      doublecount(D,E, -1 );
      doublecount(H,L,  1 );
      doublecount(H,L, -1 );

      SP := SP + 1;

      begin
            SP := SP - 1;
            if SP < 0 then alarm(5);
      end;

      begin comment halt and non-existing operationcodes;
            exc := false;
            goto interrupt
      end;

      comment nop;;

      end case val;

      goto instr;

   end


end execute;

begin comment write 8 words of the address stack;
   for i:= SP step 1 until SP+7 do
   write(out,nl,1,sp,5,<<ddd>,i,sp,5,(core(i) extract 12));
end;

comment display time; display(time);

begin comment finish debugging;
  goto interrupt
end;

comment display carry; display(carry);

begin comment display sign;
  val:=sign extract 1; display(val); sign:=val=1
end;

begin comment display zero;
```

```
      val:=zero extract 1; display(val); zero:=val=1
    end;

    begin comment display parity;
      val:=parity extract 1; display(val); parity:=val=1
    end;

    comment display program counter; display(PC);

    comment command o for out;goto outsim;

    comment command g; display(SP);


  end case action;

  goto execute;
outsim:

end program



**********************************

AFSLUTTET NORMALT
tid     :        10
linier  :       806
```

```
s2=genass intel8080 bs1 list.yes

    0       mvi  a,1
    2       mov  a,c
    3       inr  c
    4       add  c
    5   p:  1,2,3,4,5
   10       6,7,8,9,10
   15
nd   402
=algol sim8080
gin

lgol end 56


le:   bs2 15

*R


*A
0

*B
0  1

*C
0  2

*D
0  3

*E
0  4

*H
0  5

*L
0  6

*X0
ddr     0    reg    0    1    2    3    4    5    6    0    0    0    0    0
*G
000  6

*W 1
0

*X3
ddr     3    reg    2    1    2    3    4    5    6    0    0    0    0    0
*K
0  1

*M
0  1

*N
0  1

*O
0  1
```