```
                ;-----------------------------------------------------------------
                ;  PONG for Altair front panel.
                ;     May 2014, Mike Douglas
                ;
                ;  Left player quickly toggles A15 to hit the "ball." Right
                ;  player toggles A8. Score is kept in memory locations
                ;  80h and 81h (left and right). Score is missed balls, so
                ;  the lower number wins.
                ;-----------------------------------------------------------------

                ; Parameters:
                ;    SPEED determines how fast the ball moves. Higher values
                ;       move faster. Speed can easily be patched at address 1.
                ;
                ;    HITMSK, HITEDG determines how easy it is to hit the ball.
                ;       These are best changed by re-assembling the program.
                ;       Frankly, even the medium setting is too easy. Probably
                ;       best to stick with "hard" and change difficulty by
                ;       adjusting SPEED.
                ;
                ;    DEMO mode can be set by patching 35h and 65h to zero
                ;       and raising A15 and A8.

000E =          SPEED   equ     0eh             ;higher value is faster

0001 =          HITMSKR equ     01h             ;01h=hard, 03h=med, 07h=easy
0002 =          HITEDGR equ     02h             ;02h=hard, 04h=med, 08h=easy
                                                ;00h=demo with A15,A8 up

0010 =          HITMSKL equ     10h             ;10h=hard, 18h=med, 1ch=easy
0008 =          HITEDGL equ     08h             ;08h=hard, 04h=med, 02h=easy
                                                ;00h=demo with A15,A8 up

                ;-----------------------------------------------------------------
                ; Initialize
                ;-----------------------------------------------------------------
0000                    org     0




0000 010E00             lxi     b,SPEED         ;BC=adder for speed
0003 317D00             lxi     sp,stack        ;init stack pointer
0006 210000             lxi     h,0             ;zero the score
0009 228000             shld    scoreL
000C 110080             lxi     d,8000h         ;D=ball bit, E=switch status
000F C31E00             jmp     rLoop           ;begin moving right

                ;-----------------------------------------------------------------
                ; ledOut - Write D to LEDs A15-A8.
                ;   Loop accessing the address in DE which causes the proper LED
                ;   to light on the address lights. This routine is placed low
                ;   in memory so that address light A7-A5 remain off to make
                ;   A15-A8 look better.
                ;-----------------------------------------------------------------
0012 210000     ledOut  lxi     h,0             ;HL=16 bit counter
0015 1A         ledLoop ldax    d               ;display bit pattern on
0016 1A                 ldax    d               ;...upper 8 address lights
0017 1A                 ldax    d
0018 1A                 ldax    d
0019 09                 dad     b               ;increment display counter
001A D21500             jnc     ledLoop
001D C9                 ret

                ;-----------------------------------------------------------------
                ;  Moving Right
                ;-----------------------------------------------------------------
001E CD1200     rLoop   call    ledOut          ;output to LEDs A15-A8 from D

                ; Record the current right paddle state (A8) in the bit position
                ;   in E corresponding to the present ball position.

0021 DBFF               in      0ffh            ;A=front panel switches
0023 E601               ani     01h             ;get A8 alone
0025 CA2D00             jz      chkRt           ;switch not up, bit already zero
0028 7A                 mov     a,d             ;set bit in E corresponding to...
0029 B3                 ora     e               ;   the current ball position
002A E61F               ani     1fh             ;keep b7-b5 zero
002C 5F                 mov     e,a
```

```
                ; See if at the right edge. If so, see if A8 "paddle" has a hit

002D 7A         chkRt   mov     a,d             ;is ball at right edge?
002E E601               ani     1
0030 CA4500             jz      moveRt          ;no, continue moving right
0033 7B                 mov     a,e             ;switch state for each ball position
0034 E602               ani     HITEDGR         ;test edge for switch too early
0036 C23F00             jnz     missRt          ;hit too early
0039 7B                 mov     a,e             ;test for hit
003A E601               ani     HITMSKR
003C C27300             jnz     moveLfR         ;have a hit, switch direction

                ; missRt - right player missed the ball. Increment count

003F 218100     missRt  lxi     h,scoreR        ;increment right misses
0042 34                 inr     m

                ; moveRt - Move the ball right again.

0043 1E00       moveRtR mvi     e,0             ;reset switch state
0045 7A         moveRt  mov     a,d             ;move right again
0046 0F                 rrc
0047 57                 mov     d,a
0048 C31E00             jmp     rLoop

                ;-----------------------------------------------------------------
                ;  Moving left
                ;-----------------------------------------------------------------
004B CD1200     lLoop   call    ledOut          ;output to LEDs A15-A8 from D

                ; Record the current left paddle state (A15) in the bit position
                ;   in E corresponding to the present ball position.

004E DBFF               in      0ffh            ;A=front panel switches
0050 E680               ani     80h             ;get A15 alone
0052 CA5D00             jz      chkLft          ;switch not up, bit already zero
0055 7A                 mov     a,d             ;A=ball position
0056 0F                 rrc                     ;move b7..b3 to b4..b0
0057 0F                 rrc                     ;   so LEDs A7-A5 stay off

0058 0F                 rrc
0059 B3                 ora     e               ;form switch state in E
005A E61F               ani     1fh             ;keep b7-b5 zero
005C 5F                 mov     e,a

                ; See if at the left edge. If so, see if A15 "paddle" has a hit

005D 7A         chkLft  mov     a,d             ;is ball at left edge?
005E E680               ani     80h
0060 CA7500             jz      moveLf          ;no, continue moving left
0063 7B                 mov     a,e             ;switch state for each ball position
0064 E608               ani     HITEDGL         ;test edge for switch too early
0066 C26F00             jnz     missLf          ;hit too early
0069 7B                 mov     a,e             ;test for hit
006A E610               ani     HITMSKL
006C C24300             jnz     moveRtR         ;have a hit, switch direction

                ; missLf - left player missed the ball. Increment count

006F 218000     missLf  lxi     h,scoreL        ;increment left misses
0072 34                 inr     m

                ; moveLf - Move the ball left again.

0073 1E00       moveLfR mvi     e,0             ;reset switch state
0075 7A         moveLf  mov     a,d             ;move right again
0076 07                 rlc
0077 57                 mov     d,a
0078 C34B00             jmp     lLoop

                ;-----------------------------------------------------------------
                ; Data Storage
                ;-----------------------------------------------------------------
007B                    ds      2               ;stack space
007D =          stack   equ     $

0080                    org     80h             ;put at 80h and 81h
0080            scoreL  ds      1               ;score for left paddle
0081            scoreR  ds      1               ;score for right paddle
```

```
 0082                   end
```

Here is PONG in octal if you really want to enter it manually!

```
   0: 001 016 000 061 175 000 041 000   000 042 200 000 021 000 200 303
  20: 036 000 041 000 000 032 032 032   032 011 322 025 000 311 315 022
  40: 000 333 377 346 001 312 055 000   172 263 346 037 137 172 346 001
  60: 312 105 000 173 346 002 302 077   000 173 346 001 302 163 000 041
 100: 201 000 064 036 000 172 017 127   303 036 000 315 022 000 333 377
 120: 346 200 312 135 000 172 017 017   017 263 346 037 137 172 346 200
 140: 312 165 000 173 346 010 302 157   000 173 346 020 302 103 000 041
 160: 200 000 064 036 000 172 007 127   303 113 000
```

```
                ;**************************************************************************
                ;
                ;  Paper tape binary file loader
                ;       May 2014, Mike Douglas
                ;
                ;       Load this program through the front panel switches to read in
                ;       a binary file from paper tape through an SIO board at
                ;       I/O address 0/1.
                ;
                ;       As written, the program runs at 0200h (octal 1000) and loads
                ;       the paper tape file starting at location zero.
                ;
                ;       Position the tape in blank leader and start it reading, then
                ;       run this program. Stop/reset the computer after the tape is read.
                ;
                ;**************************************************************************
0000 =          sioCtl  equ     0               ;control/status port
0001 =          sioDat  equ     1               ;data port

0200                    org     0200h           ;A9 up, all other switches down

                ; Skip leading zeros.

0200 DB01       skip    in      sioDat          ;read most recent character
0202 B7                 ora     a
0203 CA0002             jz      skip            ;still null

                ; Valid data coming in. Loop storing incoming data in memory.

0206 210000             lxi     h,0             ;start at address zero

0209 77         loop    mov     m,a             ;store byte in next location
020A 23                 inx     h

020B DB00       wtChar  in      sioCtl          ;wait for a character
020D 0F                 rrc
020E DA0B02             jc      wtChar          ;negative logic (0=data present)









0211 DB01               in      sioDat          ;get the character
0213 C30902             jmp     loop

0216                    end


1000: 333 001 267 312 000 002 041 000           ;Octal dump for front panel entry
1010: 000 167 043 333 000 017 332 013
1020: 002 333 001 303 011 002
```

```
                ;*****************************************************************************
                ;
                ;   Save memory to paper tape
                ;       May 2014, Mike Douglas
                ;
                ;       Load this program through the front panel switches to punch a
                ;       binary tape from memory through an SIO board at I/O address 0/1.
                ;
                ;       As written, the program loads at 0200h (octal 1000) and
                ;       writes the tpe starting from address zero. Patch location 0201h
                ;       with the number of bytes to write (zero = 256 bytes). The program
                ;       executes a HALT instruction to stop, after which the computer
                ;       must be reset.
                ;
                ;*****************************************************************************
0000 =          sioCtl  equ     0               ;control/status port
0001 =          sioDat  equ     1               ;data port

0200                    org     0200h           ;A9 up, all other switches down

0200 0680               mvi     b,128           ;number of bytes to write
0202 210000             lxi     h,0             ;write from address zero

0205 DB00       loop    in      sioCtl          ;wait for ready to xmit
0207 07                 rlc
0208 DA0502             jc      loop            ;negative logic (0=ready to xmit)

020B 7E                 mov     a,m             ;get the byte to write
020C D301               out     sioDat          ;send it

020E 23                 inx     h               ;move to next location
020F 05                 dcr     b               ;byte counter
0210 C20502             jnz     loop

0213 76                 hlt                     ;stop

0214                    end




1000: 006 200 041 000 000 333 000 007           ;octal dump for front panel entry
1010: 332 005 002 176 323 001 043 005
1020: 302 005 002 166
```