```
;RAMTest logic is extensive and has been placed in it's own file for clari
;Note: it is included (rather than linked) due to the linker's inability
;      to support commands from a file and 132 character command line limi


    .globl TestRAM


;-------------------------------------------------------------------------
;TestRAM
;
;Do endless walking bit tests and display error range to user on screen
;
;Important it is prohibited from using RAM in this logic so registers must
;and function calls eliminated.
;
;all registers affected as follows
;
;         Normal                          Alternate
;       ----------                        ----------
;           A       general purpose           A'      store previous RAM
;           B       general purpose           B'      store MSB error lo
;           C       general purpose           C'      store LSB error lo
;           D       MSB #of bytes to check    D'      MSB end fail addre
;           E       LSB #of bytes to check    E'      LSB end fail addre
;           H       MSB ptr to byte to check  H'      MSB start fail add
;           L       LSB ptr to byte to check  L'      LSB start fail add
;           IX      sim call return address
;           IY      ---
;-------------------------------------------------------------------------

TestRAM:
    exx                                 ;setup alternate bank register
    ld      hl,#0xffff                  ;start of failures
    ld      de,#0x0000                  ;end of failures
    exx

    ld      a,#DISPHOME                 ;reset the cursor to the home
    or      #DISPWRBIT                  ;
    out     (DISPPORT), a               ;
    xor     a                           ;
    out     (DISPPORT), a               ;

TRLoop:                                 ;repeat
    ld      hl,(RAMHighAddress)         ;  get RAM locations found by
```

```
    ld      de,(RAMLowAddress)          ;
    xor     a                           ;  clear c flag
    sbc     hl,de                       ;  hl = #of RAM bytes found
    inc     hl                          ;  adjust up by one to include
    ex      de,hl                       ;  hl=start of RAM address, de
TRFullLoop:                             ;  repeat
    ld      a,(hl)                      ;
    ex      af,af'                      ;    save the prior RAM conten

    exx                                 ;
    ld      bc,#0000                    ;    set bc' to zero indicatin
    exx

    ;----------------------
    ;Walking bit test left
    ;----------------------

    ld      b,#8                        ;
    ld      c,#0x01                     ;    setup pattern
TRWBLLoop:                              ;    repeat
    ld      a,c                         ;
    ld      (hl),a                      ;      write pattern
    cp      (hl)                        ;      check pattern written
    jr      z,TRWBLOK                   ;      if (pattern failed)
    ld      a,#0x4c                     ;        get the 'L' character
    jr      TRWBFail                    ;        *abort this byte as i
TRWBLOK:                                ;      endif
    rlca                                ;      rotate pattern left
    ld      c,a                         ;
    djnz    TRWBLLoop                   ;    until (all bits tested)

    ;----------------------
    ;Walking bit test right
    ;----------------------

    ld      b,#8                        ;
    ld      c,#0x80                     ;    setup pattern
TRWBRLoop:                              ;    repeat
    ld      a,c                         ;
    ld      (hl),a                      ;      write pattern
    cp      (hl)                        ;      check pattern written
    jr      z,TRWBROK                   ;      if (pattern failed)
    ld      a,#0x52                     ;        get the 'R' character
    jr      TRWBFail                    ;        *abort this byte as i
TRWBROK:                                ;      endif
```

```
    rrca                                    ;       rotate pattern right
    ld      c,a                             ;
    djnz    TRWBRLoop                       ;    until (all bits tested)
    jr      TRWBDone

TRWBFail:                                   ;    if (byte failed any of th

                                            ;       note: this destroys the
                                            ;             but that really d
                                            ;             is faulty anyway.
    ld      a,h                             ;
    ex      af,af'                          ;       a' = error MSN
    ld      a,l                             ;       a = error LSN
    exx                                     ;       SWAP REGISTER SETS
    ld      c,a                             ;       hl' = start of error ra
    ex      af,af'                          ;
    ld      b,a                             ;       bc' = error address

    ;-------------------------------------------------------------
    ;Update range start (hl') if error is lower than old start
    ;-------------------------------------------------------------


;   ld      a,b                             ;       get error MSB
    cp      h                               ;       compare it to range sta
    jr      c,TRWBFlow                      ;       if (error MSB < start M
    jr      nz,TRWBFnlow                    ;       if (error MSB != start
    ld      a,c                             ;       get error LSB
    cp      l                               ;
    jr      nc,TRWBFnlow                    ;       if (error LSB >= start
TRWBFlow:                                   ;       if (new low address has
    ld      a,b                             ;
    ld      h,a                             ;         update hl' with the n
    ld      a,c                             ;
    ld      l,a                             ;
TRWBFnlow:                                  ;       endif

    ;-------------------------------------------------------------
    ;Update range end (de') if error is higher than old end
    ;-------------------------------------------------------------

    ld      a,b                             ;       get error MSB
    cp      d                               ;       compare it to range end
    jr      c,TRWBFnhigh                    ;       if (error MSB < end MSB
    jr      nz,TRWBFhigh                    ;       if (error MSB != end MS
```

```
    ld      a,c                             ;       get error LSB
    cp      e                               ;
    jr      c,TRWBFnhigh                    ;       if (error LSB < end LSB
TRWBFhigh:                                  ;       if (new high address ha
    ld      a,b                             ;
    ld      d,a                             ;         update de' with the n
    ld      a,c                             ;
    ld      e,a                             ;
TRWBFnhigh:                                 ;       endif
    exx                                     ;       RETURN TO ORIGINAL REGI


TRWBDone:                                   ;     endif

    ex      af,af'                          ;
    ld      (hl),a                          ;     restore the prior RAM con

    inc     hl                              ;     increment pointer

    dec     de                              ;     decrement count
    ld      a,d                             ;
    or      e                               ;
    jr      z,TRComplete                    ;
    jp      TRFullLoop                      ;   until (all RAM bytes are te
TRComplete:

    ;-----------------------------------------------------
    ;All of initially detected memory has now been tested
    ;Display range of errors found
    ;-----------------------------------------------------


    exx                                     ;   SWAP REGISTER SETS

    ld      a,b                             ;   check bc' so see if it has
    or      c                               ;   indicating no errors found
    jr      nz,TREF                         ;   if (no errors have been fou
    ld      a,#0x2b                         ;
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;     display a '+'
    xor     a                               ;
    out     (DISPPORT), a                   ;
    jr      TRNOEF
TREF:                                       ;   else
```

```
    ;--------------------------------------------
    ;Display '('
    ;--------------------------------------------

    ld      a,#0x28                         ;
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;
    xor     a                               ;
    out     (DISPPORT), a                   ;

    ;--------------------------------------------
    ;Display start of error range as 4 hex bytes
    ;--------------------------------------------

    ld      a,h                             ;    get MSB
    ld      ix,#TRFR1                       ;
    jr      TRDispA                         ;    simulate 'call' to TRDisp
TRFR1:                                      ;
    ld      a,l                             ;    get LSB
    ld      ix,#TRFR2                       ;
    jr      TRDispA                         ;    simulate 'call' to TRDisp
TRFR2:                                      ;

    ;--------------------------------------------
    ;Display '-'
    ;--------------------------------------------

    ld      a,#0x2d                         ;
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;
    xor     a                               ;
    out     (DISPPORT), a                   ;

    ;--------------------------------------------
    ;Display end of error range as 4 hex bytes
    ;--------------------------------------------

    ld      a,d                             ;    get MSB
    ld      ix,#TRFR3                       ;
    jr      TRDispA                         ;    simulate 'call' to TRDisp
TRFR3:                                      ;
    ld      a,e                             ;    get LSB
    ld      ix,#TRFR4                       ;
    jr      TRDispA                         ;    simulate 'call' to TRDisp
TRFR4:                                      ;
```

```
    ;-----------------------------------------
    ;Display ')'
    ;-----------------------------------------

    ld      a,#0x29                         ;
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;
    xor     a                               ;
    out     (DISPPORT), a                   ;

TRNOEF:                                     ;  endif

    exx                                     ;  RETURN TO ORIGINAL REGISTER

    jp      TRLoop                          ;until (forever)

    ret


;-----------------------------------------
;Display A as a 2 byte hex value on screen
;return to address in IX
;  (as stack can't be used)
;
; DO NOT USE A CALL - MUST JUMP, w/return in IX
;
; DESTROYS 'c' REGISTER IN ALTERNATE REGISTER SET
; only af altered in current register set
;-----------------------------------------

TRDispA:
    exx
    ld      c,a                             ;store value in alternate regi
    exx

    rra
    rra
    rra
    rra

    and     #0x0f                           ;display low nibble of A on th
    cp      #10                             ;
    jr      nc, TF1Alpha                    ;if (A < 10)
    add     #0x30                           ;  offset value to '0'
```

```
    jr      TF1Done                         ;
TF1Alpha:                                   ;else
    add     #(0x41 - 10)                    ;  offset value to 'A'
TF1Done:                                    ;endif
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;write it to the display port
    xor     a                               ;
    out     (DISPPORT), a                   ;

    exx
    ld      a,c                             ;restore value from alternate
    exx

    and     #0x0f                           ;display low nibble of A on th
    cp      #10                             ;
    jr      nc, TF2Alpha                    ;if (A < 10)
    add     #0x30                           ;  offset value to '0'
    jr      TF2Done                         ;
TF2Alpha:                                   ;else
    add     #(0x41 - 10)                    ;  offset value to 'A'
TF2Done:                                    ;endif
    or      #DISPWRBIT                      ;
    out     (DISPPORT), a                   ;write it to the display port
    xor     a                               ;
    out     (DISPPORT), a                   ;

    jp      (ix)                            ;simulated 'return'
```