```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; game start power-on
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


                LD      A,#00                           ; A := 0
                LD      (REG_VBLANK_ENABLED),A          ; disable interrupts
                JP      Init                            ; skip ahead

;
; RST     #8
; if there are credits or the game is being played it returns immediately.  if not, it returns to higher subroutine
;

0008  3A0760    LD      A,(NoCredits)   ; load A with 1 when no credits have been inserted, 0 if any credits exist or game is
being played
000B  0F        RRCA                    ; any credits in the game ?
000C  D0        RET     NC              ; yes, return

000D  33        INC     SP
000E  33        INC     SP
000F  C9        RET                     ; else return to higher subroutine

;
; RST     #10
; if mario is alive, it returns.  if mario is dead, it returns to the higher subroutine.
;

0010  3A0062    LD      A,(#6200)       ; 1 when mario is alive, 0 when dead
0013  0F        RRCA                    ; is mario alive?
0014  D8        RET     C               ; yes, return

0015  33        INC     SP              ; no, increase SP by 2 and return
0016  33        INC     SP
0017  C9        RET                     ; effectively returns twice

;
; RST     #18
;

0018  210960    LD      HL,WaitTimerMSB ; load timer that counts down
001B  35        DEC     (HL)            ; Count it down...
001C  C8        RET     Z               ; Return if zero

001D  33        INC     SP              ; otherwise Increase SP twice
001E  33        INC     SP
001F  C9        RET                     ; and return - effectively returns to higher subroutine

;
; RST     #20
;

0020  210860    LD      HL,WaitTimerLSB ; load HL with timer
0023  35        DEC     (HL)            ; count it down
0024  28F2      JR      Z,#0018         ; If zero skip up and count down the other timer

0026  E1        POP     HL              ; else move stack pointer up and return to higher subroutine
0027  C9        RET                     ; return


;
; RST     #28
; jumps program to (2*A + Next program address)
; used in conjuction with a jump table after the call
;

0028  87        ADD     A,A             ; A := A * 2
0029  E1        POP     HL              ; load HL with address of jump table
002A  5F        LD      E,A             ; load E with A
002B  1600      LD      D,#00           ; D := 0
002D  C33200    JP      #0032           ; skip ahead

;
; RST #30
;

0030  1812      JR      #0044           ; this core sub is actually at #0044

;
; continuation of RST #28 from #002D above
;

0032  19        ADD     HL,DE           ; HL is now 2A more than it was
0033  5E        LD      E,(HL)          ; load E with low byte from the table
0034  23        INC     HL              ; next table entry
0035  56        LD      D,(HL)          ; load D with high byte from table
0036  EB        EX      DE,HL           ; DE <> HL
0037  E9        JP      (HL)            ; jump to the address in HL

;
; RST     #38
; HL and C are preloaded
; updates #A (10 decimal) by adding C from each location from HL to HL + #40 by 4
```

```
; [the bytes affected are offset by 4 bytes each]
;
; Also #003D is called from several places. used for updating girl's sprite
;

0038  110400    LD      DE,#0004        ; load offset of 4 to add
003B  060A      LD      B,#0A           ; for B = 1 to #A (10 decimal)

003D  79        LD      A,C             ; Load A with C
003E  86        ADD     A,(HL)          ; Add the contents of HL into A
003F  77        LD      (HL),A          ; put back into HL, this increases the value in HL by C
0040  19        ADD     HL,DE           ; next HL to do will be 4 more than previous
0041  10FA      DJNZ    #003D           ; next B

0043  C9        RET                     ; return

; continuation of rst #30
; used to check a screen number.  if it doesn't match, the 2nd level of subroutine is returned
; A is preloaded with the check value, in binary

0044  212762    ld      HL,#6227        ; Load HL with address of Screen #
0047  46        ld      B,(HL)          ; load B with Screen #, For B = 1 to screen # (1, 2, 3 or 4)

0048  0F        RRCA                    ; Rotate A right with carry
0049  10Fd      DJNZ    #0048           ; Next B

004B  d8        RET    c                ; return if carry

004C  E1        POP     HL              ; otherwise HL gets the stack = return to higher subroutine
004D  C9        RET                     ; return

; HL is preloaded with source data of kong sprites values
; this subroutine copies the memory values of HL to HL + #28 into #6908 through #6908 + #28
; used to set all the kong sprites

004E  110869    LD      DE,#6908        ; Kong's Sprites start
0051  012800    LD      BC,#0028        ; #28 bytes to copy
0054  EDB0      LDIR                    ; copy
0056  C9        RET                     ; return

; this subroutine takes the value of RngTimer1 and adds into it the values from FrameCounter and RngTimer2
; it returns with A loaded with this result and also RngTimer1 with the answer.
; random number generator

0057  3A1860    LD      A,(RngTimer1)       ; load A with timer
005A  211A60    LD      HL,FrameCounter        ; load HL with other timer address
005D  86        ADD     A,(HL)          ; add
005E  211960    LD      HL,RngTimer2        ; load HL with yet another timer address
0061  86        ADD     A,(HL)          ; add
0062  321860    LD      (RngTimer1),A       ; store
0065  C9        RET                     ; return

; interrupt routine

0066  F5        PUSH    AF
0067  C5        PUSH    BC
0068  D5        PUSH    DE
0069  E5        PUSH    HL
006A  DDE5      PUSH    IX
006C  FDE5      PUSH    IY              ; save all registers

006E  AF        XOR     A               ; A := 0
006F  32847D    LD      (REG_VBLANK_ENABLE),A       ; disable interrupts
0072  3A007D    LD      A,(IN2)         ; load A with Credit/Service/Start Info
0075  E601      AND     #01             ; is the Service button being pressed?
0077  C20040    JP      NZ,#4000        ; yes, jump to #4000 [??? this would cause a crash ???]

007A  213801    LD      HL,#0138        ; load HL with start of table data
007D  CD4101    CALL    #0141           ; refresh the P8257 Control registers / refresh sprites to hardware
0080  3A0760    LD      A,(NoCredits)   ; load the credit indicator
0083  A7        AND     A               ; are there credits present / is a game being played ?
0084  C2B500    JP      NZ,#00B5        ; No, jump ahead

0087  3A2660    LD      A,(UprightCab)       ; yes, load A with upright/cocktail
008A  A7        AND     A               ; upright ?
008B  C29800    JP      NZ,#0098        ; yes, jump ahead

008E  3A0E60    LD      A,(PlayerTurnB)       ; else load A with player number
0091  A7        AND     A               ; is this player 2 ?
0092  3A807C    LD      A,(IN1)         ; load A with raw input from player 2
0095  C29B00    JP      NZ,#009B        ; yes, skip next step

0098  3A007C    LD      A,(IN0)         ; load A with raw input from player 1
009B  47        LD      B,A             ; copy to B
009C  E60F      AND     #0F             ; mask left 4 bits to zero
009E  4F        LD      C,A             ; copy this to C
009F  3A1160    LD      A,(RawInput)       ; load A with player input
00A2  2F        CPL                     ; The contents of A are inverted (one’s complement).
00A3  A0        AND     B               ; logical and with raw input - checks for jump button
00A4  E610      AND     #10             ; mask all bits but 4.  if jump was pressed it is there
00A6  17        RLA
00A7  17        RLA
00A8  17        RLA                     ; rotate left 3 times
00A9  B1        OR      C               ; mix back into masked input
00AA  60        LD      H,B             ; load H with B = raw input
```

```
00AB  6F        LD      L,A             ; load L with A = modified input
00AC  221060    LD      (InputState),HL      ; store into input memories, InputState and RawInput
00AF  78        LD      A,B             ; load A with raw input
00B0  CB77      BIT     6,A             ; is the bit 6 set for reset?
00B2  C20000    JP      NZ,#0000        ; if reset, jump back to #0000 for a reboot

00B5  211A60    LD      HL,FrameCounter        ; else load HL with Timer constantly counts down from FF to 00 and then FF to
00 again and again ... 1 count per frame
00B8  35        DEC     (HL)            ; decrease this timer
00B9  CD5700    CALL    #0057           ; update the random number gen
00BC  CD7B01    CALL    #017B           ; check for credits being inserted and handle them
00BF  CDE000    CALL    #00E0           ; update all sounds
00C2  21D200    LD      HL,#00D2        ; load HL with return address
00C5  E5        PUSH    HL              ; push to stack so any RETs go there (#00D2)
00C6  3A0560    LD      A,(GameMode1)   ; load A with game mode1

; GameMode1 is 0 when game is turned on, 1 when in attract mode.  2 when credits in waiting for start, 3 when playing game

                RST     #28             ; jump based on above:

00CA  C3 01                             ; #01C3 = startup
00CC  3C 07                             ; #073C = attract mode
00CE  B2 08                             ; #08B2 = credits, waiting
00D0  FE 06                             ; #06FE = playing game

; return here from any of the jumps above, based on return address pushed to stack at #00C5

00D2  FDE1      POP     IY
00D4  DDE1      POP     IX
00D6  E1        POP     HL
00D7  D1        POP     DE
00D8  C1        POP     BC              ; restore all registers except AF

00D9  3E01      LD      A,#01           ; A := 1
00DB  32847D    LD      (REG_VBLANK_ENABLE),A       ; enable interrupts
00DE  F1        POP     AF              ; restore AF
00DF  C9        RET                     ; return from interrupt

; called from #00BF
; updates all sounds

00E0  218060    LD      HL,#6080        ; source data at sound buffer
00E3  11007D    LD      DE,REG_SFX      ; set destination to sound outputs
00E6  3A0760    LD      A,(NoCredits)   ; load A with credit indicator
00E9  A7        AND     A               ; have credits been inserted / is there a game being played ?
00EA  C0        RET     NZ              ; no, return [change to NOP to enable sound in demo ]

; this sub writes the sound buffer to the hardware
; sounds have durations to play in the buffer

00EB  0608      LD      B,#08           ; yes, there was a credit or a game is being played.  For B = 1 to 8 Do:

00ED  7E        LD      A,(HL)          ; load A with sound duration / sound effect for the sound
00EE  A7        AND     A               ; is there a sound to play ?
00EF  CAF500    JP      Z,#00F5         ; no, skip next 2 steps

00F2  35        DEC     (HL)            ; yes, decrease the duration
00F3  3E01      LD      A,#01           ; A := 1

00F5  12        LD      (DE),A          ; store sound to output (play sound)
00F6  1C        INC     E               ; next output address
00F7  2C        INC     L               ; next source address
00F8  10F3      DJNZ    #00ED           ; Next B

00FA  218B60    LD      HL,#608B        ; load HL with music timer
00FD  7E        LD      A,(HL)          ; load A with this value
00FE  A7        AND     A               ; == 0 ?
00FF  C20801    JP      NZ,#0108        ; no, skip ahead 4 steps

0102  2D        DEC     L               ; else
0103  2D        DEC     L               ; HL := #6089
0104  7E        LD      A,(HL)          ; load A with this value to use for music
0105  C30B01    JP      #010B           ; skip next 3 steps

0108  35        DEC     (HL)            ; decrease timer
0109  2D        DEC     L               ; HL := #608A
010A  7E        LD      A,(HL)          ; load A with this tune to use

010B  32007C    LD      (REG_MUSIC),A   ; play music
010E  218860    LD      HL,#6088        ; load HL with address/counter for mario dying sound
0111  AF        XOR     A               ; A := 0
0112  BE        CP      (HL)            ; compare.  is mario dying ?
0113  CA1801    JP      Z,#0118         ; no, skip next 2 steps

0116  35        DEC     (HL)            ; else decrease the counter
0117  3C        INC     A               ; A := 1

0118  32807D    LD      (REG_SFX_DEATH),A   ; store A into digital sound trigger -death (?)
011B  C9        RET                     ; return

; clear all sounds
; called from several places

011C  0608      LD      B,#08           ; For B = 1 to 8
011E  AF        XOR     A               ; A := 0
```

```
011F  21007D    LD      HL,REG_SFX      ; [REG_SFX..REG_SFX+7] get all zeros
0122  118060    LD      DE,#6080        ; #6080-#6088 get all zeros - clears sound buffer

0125  77        LD      (HL),A          ; clear this memory - clears sound outputs
0126  12        LD      (DE),A          ; clear this memory
0127  2C        INC     L               ; next memory
0128  1C        INC     E               ; next memory
0129  10FA      DJNZ    #0125           ; Next B

012B  0604      LD      B,#04           ; For B = 1 to 4

012D  12        LD      (DE),A          ; #6088-#608B get all zeros
012E  1C        INC     E               ; next DE
012F  10FC      DJNZ    #012D           ; Next B

0131  32807D    LD      (REG_SFX_DEATH),A   ; clear the digital sound trigger (death)
0134  32007C    LD      (REG_MUSIC),A   ; clear the sound output
0137  C9        RET                     ; return

; data used in sub below

0138  53 00 69 80 41 00 70 80
0140  81

; called from #007D
; HL is preloaded with #0138
; This copies the sprite data from $6900 to $7000
; Presumably the reason sprite data isn't stored in $7000 in the first place is to ensure it's updated only during vblank.

0141  AF        XOR     A               ; A := 0
0142  32857D    LD      (REG_DMA),A     ; store into P8257 DRQ DMA Request
0145  7E        LD      A,(HL)          ; load table data (#53)
0146  320878    LD      (#7808),A       ; store into P8257 control register
0149  23        INC     HL              ; next table entry
014A  7E        LD      A,(HL)          ; load table data (#00)
014B  320078    LD      (#7800),A       ; store into P8257 control register
014E  23        INC     HL              ; next table entry
014F  7E        LD      A,(HL)          ; load table data (#69)
0150  320078    LD      (#7800),A       ; store into P8257 control register
0153  23        INC     HL              ; next table entry
0154  7E        LD      A,(HL)          ; load table data (#80)
0155  320178    LD      (#7801),A       ; store into P8257 control register
0158  23        INC     HL              ; next table entry
0159  7E        LD      A,(HL)          ; load table data (#41)
015A  320178    LD      (#7801),A       ; store into P8257 control register
015D  23        INC     HL              ; next table entry
015E  7E        LD      A,(HL)          ; load table data (#00)
015F  320278    LD      (#7802),A       ; store into P8257 control register
0162  23        INC     HL              ; next table entry
0163  7E        LD      A,(HL)          ; load table data (#70)
0164  320278    LD      (#7802),A       ; store into P8257 control register
0167  23        INC     HL              ; next table entry
0168  7E        LD      A,(HL)          ; load table data (#80)
0169  320378    LD      (#7803),A       ; store into P8257 control register
016C  23        INC     HL              ; next table entry
016D  7E        LD      A,(HL)          ; load table data (#81)
016E  320378    LD      (#7803),A       ; store into P8257 control register
0171  3E01      LD      A,#01           ; A := 1
0173  32857D    LD      (REG_DMA),A     ; store into P8257 DRQ DMA Request
0176  AF        XOR     A               ; A := 0
0177  32857D    LD      (REG_DMA),A     ; store into P8257 DRQ DMA Request
017A  C9        RET                     ; return

; called from #00BC
; checks for and handles credits

017B  3A007D    LD      A,(IN2)         ; load A with IN2
017E  CB7F      BIT     7,A             ; is the coin switch active?
0180  210360    LD      HL,CoinSwitch   ; load HL with pointer to coin switch indicator
0183  C28901    JP      NZ,#0189        ; yes, skip next 2 steps

0186  3601      LD      (HL),#01        ; otherwise store 1 into coin switch indicator  -  this is for coin insertion
0188  C9        RET                     ; return

0189  7E        LD      A,(HL)          ; Load A with coin switch indicator
018A  A7        AND     A               ; has a coin been inserted ?
018B  C8        RET     Z               ; no, return

; coin has been inserted

018C  E5        PUSH    HL              ; else save HL to stack
018D  3A0560    LD      A,(GameMode1)   ; load A with game mode1
0190  FE03      CP      #03             ; is someone playing?
0192  CA9D01    JP      Z,#019D         ; yes, skip ahead and don't play the sound

0195  CD1C01    CALL    #011C           ; no, then clear all sounds
0198  3E03      LD      A,#03           ; load sound duration
019A  328360    LD      (#6083),A       ; plays the coin insert sound

019D  E1        POP     HL              ; restore HL from stack
019E  3600      LD      (HL),#00        ; store 0 into coin switch indicator - no more coins
01A0  2B        DEC     HL              ; HL := CoinCounter
01A1  34        INC     (HL)            ; increase this counter
01A2  112460    LD      DE,CoinsPerCredit2  ; load DE with # of coins needed per credit
01A5  1A        LD      A,(DE)          ; load A with coins needed
```

```
01A6  96        SUB     (HL)            ; has the player inserted enough coins for a new credit?
01A7  C0        RET     NZ              ; yes, return (CoinCounter is now zero)

01A8  77        LD      (HL),A          ; no; restore CoinCounter
01A9  13        INC     DE              ; DE := CreditsPerCoin
01AA  2B        DEC     HL              ; HL := NumCredits
01AB  EB        EX      DE,HL           ; DE := NumCredits, HL := CreditsPerCoin
01AC  1A        LD      A,(DE)          ; load A with number of credits in BCD
01AD  FE90      CP      MAX_CREDITS     ; is the number of credits already maxed out?
01AF  D0        RET     NC              ; yes; return

01B0  86        ADD     A,(HL)          ; add number of credits with # of credits per coin
01B1  27        DAA                     ; decimal adjust
01B2  12        LD      (DE),A          ; store result in credits
01B3  110004    LD      DE,#0400        ; load task #4 - draws credits on screen if any are present
01B6  CD9F30    CALL    #309F           ; insert task
01B9  C9        RET                     ; return

; table data used below in 01C6

01BA  00 37 00 AA AA AA 50 76 00

; this is called when the game is first turned on or reset from #00C9

01C3  CD7408    CALL    #0874           ; clears the screen and sprites
01C6  21BA01    LD      HL,#01BA        ; start of table data above
01C9  11B260    LD      DE,#60B2        ; set destination
01CC  010900    LD      BC,#0009        ; set counter to 9
01CF  EDB0      LDIR                    ; copy 9 bytes above into #60B2-#60BB
01D1  3E01      LD      A,#01           ; A := 1
01D3  320760    LD      (NoCredits),A   ; store into credit indicator == no credits exist
01D6  322962    LD      (#6229),A       ; initialize level to 1
01D9  322862    LD      (#6228),A       ; set number of lives remaining to 1
01DC  CDB806    CALL    #06B8           ; if a game is played or credits exist, display remaining lives-1 and level
01DF  CD0702    CALL    #0207           ; set all dip switch settings and create default high score table from ROM
01E2  3E01      LD      A,#01           ; A := 1
01E4  32827D    LD      (REG_FLIPSCREEN),A      ; store into flip screen setting
01E7  320560    LD      (GameMode1),A   ; store into game mode 1
01EA  322762    LD      (#6227),A       ; initialize screen to 1 (girders)
01ED  AF        XOR     A               ; A := 0
01EE  320A60    LD      (GameMode2),A       ; store into game mode 2
01ED  C3303F    JP      #3F30           ; jump to additional code to set game mode 2 to 6 - make game start with title screen
01EE  00        NOP                     ; no operation
01F1  CD530A    CALL    #0A53           ; draw "1UP" on screen
01F4  110403    LD      DE,#0304        ; load task data to draw "HIGH SCORE"
01F7  CD9F30    CALL    #309F           ; insert task to draw text
01FA  110202    LD      DE,#0202        ; load task #2, parameter 2 to display the high score
01FD  CD9F30    CALL    #309F           ; insert task
0200  110002    LD      DE,#0200        ; load task #2, parameter 0 to display player 1 score
0203  CD9F30    CALL    #309F           ; insert task
0206  C9        RET                     ; return

; this sub reads and sets the dip switch settings, and creates the default high score table

0207  3A807D    LD      A,(DSW1)        ; load A with Dip switch settings
020A  4F        LD      C,A             ; copy to C
020B  212060    LD      HL,StartingLives        ; set destination address to initial number of lives
020E  E603      AND     #03             ; mask bits, now between 0 and 3 inclusive
0210  C603      ADD     A,#03           ; Add 3, now between 3 and 6 inclusive
0212  77        LD      (HL),A          ; store in initial number of lives
0213  23        INC     HL              ; next HL, now at ExtraLifeThreshold = score needed for extra life
0214  79        LD      A,C             ; load A with original value of dip switches
0215  0F        RRCA                    ;
0216  0F        RRCA                    ; rotate right twice
0217  E603      AND     #03             ; mask bits, now between 0 and 3
0219  47        LD      B,A             ; copy to B.  used in minisub below for loop counter
021A  3E07      LD      A,#07           ; A := 7 = default score for extra life
021C  CA2602    JP      Z,#0226         ; on zero, jump ahead and use 7

021F  3E05      LD      A,#05           ; A : = 5

0221  C605      ADD     A,#05           ; add 5
0223  27        DAA                     ; decimal adjust
0224  10FB      DJNZ    #0221           ; loop until done

0226  77        LD      (HL),A          ; store the result in score for extra life
0227  23        INC     HL              ; HL := CoinsPerCredit
0228  79        LD      A,C             ; load A with dipswitch
0229  010101    LD      BC,#0101        ; B := 1, C := 1
022C  110201    LD      DE,#0102        ; D := 1, E := 2
022F  E670      AND     #70             ; mask bits.  turns off all except the 3 used for coins/credits
0231  17        RLA
0232  17        RLA
0233  17        RLA
0234  17        RLA                     ; rotate left 4 times.  now in lower 3 bits
0235  CA4702    JP      Z,#0247         ; if zero, skip ahead and leave BC and DE alone

0238  DA4102    JP      C,#0241         ; if there was a carry, skip ahead

023B  3C        INC     A               ; increase A
023C  4F        LD      C,A             ; store into C
023D  5A        LD      E,D             ; E := 1
023E  C34702    JP      #0247           ; skip ahead

0241  C602      ADD     A,#02           ; else A := 2
```

```
0243  47        LD      B,A             ; B := 2
0244  57        LD      D,A             ; D := 2
0245  87        ADD     A,A             ; A := 4
0246  5F        LD      E,A             ; E := 4

0247  72        LD      (HL),D          ; store D into CoinsPerCredit
0248  23        INC     HL              ; HL := CoinsPer2Credits
0249  73        LD      (HL),E          ; store E into CoinsPer2Credits
024A  23        INC     HL              ; HL := CoinsPerCredit2
024B  70        LD      (HL),B          ; store B into CoinsPerCredit2
024C  23        INC     HL              ; HL := CreditsPerCoin
024D  71        LD      (HL),C          ; store DE and BC into coins/credits
024E  23        INC     HL              ; HL := UprightCab = memory for upright/cocktail
024F  3A807D    LD      A,(DSW1)        ; load A with dipswitch settings
0252  07        RLCA                    ; rotate left
0253  3E01      LD      A,#01           ; A := 1
0255  DA5902    JP      C,#0259         ; if carry, skip next step

0258  3D        DEC     A               ; A := 0

0259  77        LD      (HL),A          ; store into upright / cocktail
025A  216535    LD      HL,#3565        ; source = #3565 = default high score table
025D  110061    LD      DE,#6100        ; dest = #6100 = high score RAM
0260  01AA00    LD      BC,#00AA        ; byte counter = #AA
0263  EDB0      LDIR                    ; copy high score table into RAM
0265  C9        RET                     ; return

; come here from game power-on
; first, clear system RAM
Init:
0266  0610      LD      B,#10           ; for B = 0 to #10
0268  210060    LD      HL,RAM          ; set destination
026B  AF        XOR     A               ; A := 0

026C  4F        LD      C,A             ; For C = 0 to #FF

026D  77        LD      (HL),A          ; store 0 into memory
026E  23        INC     HL              ; next location
026F  0D        DEC     C               ; Next C
0270  20FB      JR      NZ,#026D        ; Loop until done

0272  10F8      DJNZ    #026C           ; Next B

; clears sprite memory

0274  0604      LD      B,#04           ; For B = 1 to 4
0276  210070    LD      HL,SPRITE_RAM   ; load HL with start address
0279  4F        LD      C,A             ; For C = 0 to #FF

027A  77        LD      (HL),A          ; Clear this memory
027B  23        INC     HL              ; next memory
027C  0D        DEC     C               ; Next C
027D  20FB      JR      NZ,#027A        ; loop until done

027F  10F8      DJNZ    #0279           ; Next B

; this subroutine clears the VIDEO RAM with #10 (clear shape)

0281  0604      LD      B,#04           ; for B = 1 to 4
0283  3E10      LD      A,#10           ; #10 is the code for clear on the screen
0285  210074    LD      HL,#7400        ; load HL with beginning of graphics memory

0288  0E00      LD      C,#00           ; For C = 1 to #FF

028A  77        LD      (HL),A          ; load clear into video RAM
028B  23        INC     HL              ; next location
028C  0D        DEC     C               ;
028D  20FB      JR      NZ,#028A        ; Next C

028F  10F7      DJNZ    #0288           ; Next B

; Loads #60C0 to #60FF (task list) with #FF

0291  21C060    LD      HL,#60C0        ; HL points to start of task list
0294  0640      LD      B,#40           ; For B = 1 to #40
0296  3EFF      LD      A,#FF           ; load A with code for no task

0298  77        LD      (HL),A          ; store into task location
0299  23        INC     HL              ; next location
029A  10FC      DJNZ    #0298           ; Next B

; reset some memories to 0 and 1

029C  3EC0      LD      A,#C0           ; load A with #C0 for the #60B0 and #60B1 timers
029E  32B060    LD      (#60B0),A       ; store into timer
02A1  32B160    LD      (#60B1),A       ; store into timer
02A4  AF        XOR     A               ; A := 0
02A5  32837D    LD      (REG_SPRITE),A  ; Clear dkong_spritebank_w  /* 2 PSL Signal */

02A8  32867D    LD      (REG_PALETTE_A),A       ; clear palette bank selector
02AB  32877D    LD      (REG_PALETTE_B),A       ; clear palette bank selector
02AE  3C        INC     A               ; A: = 1
02AF  32827D    LD      (REG_FLIPSCREEN),A       ; set flip screen setting
02B2  31006C    LD      SP,#6C00        ; set Stack Pointer to #6C00
02B5  CD1C01    CALL    #011C           ; clear all sounds
```

```
02B8  3E01      LD      A,#01           ; A := 1
02BA  32847D    LD      (REG_VBLANK_ENABLE),A       ; enable interrupts

;
; arrive after RET encountered after #0306 jump
; check for tasks and do them if they exist
;

02BD  2660      LD      H,#60           ; H := #60
02BF  3AB160    LD      A,(#60B1)       ; load A with task pointer
02C2  6F        LD      L,A             ; copy to L.  HL now has #60XX which is the current task
02C3  7E        LD      A,(HL)          ; load A with task
02C4  87        ADD     A,A             ; double.  Is there a task to do ?
02C5  301C      JR      NC,#02E3        ; yes, skip ahead to handle task

02C7  CD1503    CALL    #0315           ; else flash the "1UP" above the score when it is time to do so
02CA  CD5003    CALL    #0350           ; check for and handle awarding extra lives
02CD  211960    LD      HL,RngTimer2        ; load HL with timer
02D0  34        INC     (HL)            ; increase the timer
02D1  218363    LD      HL,#6383        ; load HL with address of memory used to track tasks
02D4  3A1A60    LD      A,(FrameCounter)       ; load A with timer that constantly counts down from #FF to 0
02D7  BE        CP      (HL)            ; equal ?
02D8  28E3      JR      Z,#02BD         ; yes, loop back to check for more tasks

02DA  77        LD      (HL),A          ; else store A into the memory, for next time
02DB  CD7F03    CALL    #037F           ; check for updating of difficulty
02DE  CDA203    CALL    #03A2           ; check for releasing fires on girders and conveyors
02E1  18DA      JR      #02BD           ; loop back to check for more tasks

; arrive from #02C5
; loads data from the task list at #60C0 through #60CF
; tasks are loaded in subroutine at #309F
; HL is preloaded with task pointer
; A is preloaded with 2x the task number

02E3  E61F      AND     #1F             ; mask bits.  A now between 0 and #1F
02E5  5F        LD      E,A             ; copy to E
02E6  1600      LD      D,#00           ; D := 0
02E8  36FF      LD      (HL),#FF        ; overwrite the task with empty entry
02EA  2C        INC     L               ; next HL
02EB  4E        LD      C,(HL)          ; load C with the 2nd byte of the task (parameter)
02EC  36FF      LD      (HL),#FF        ; overwrite the task with empty entry
02EE  2C        INC     L               ; next HL
02EF  7D        LD      A,L             ; load A with low byte of the address
02F0  FEC0      CP      #C0             ; < #C0 ?
02F2  3002      JR      NC,#02F6        ; no, skip next step

02F4  3EC0      LD      A,#C0           ; reset low byte to #C0

02F6  32B160    LD      (#60B1),A       ; store into the task pointer
02F9  79        LD      A,C             ; load A with the 2nd byte of the task
02FA  21BD02    LD      HL,#02BD        ; load HL with return address
02FD  E5        PUSH    HL              ; push to stack so RET will go to #02BD = task list
02FE  210703    LD      HL,#0307        ; load HL with data from table below
0301  19        ADD     HL,DE           ; add the offset based on byte 1 of the task
0302  5E        LD      E,(HL)          ; load E with the low byte from the table below
0303  23        INC     HL              ; next HL
0304  56        LD      D,(HL)          ; load D with the high byte from the table
0305  EB        EX      DE,HL           ; DE <> HL
0306  E9        JP      (HL)            ; jump to address from the table

; data for jump table used above
; task table

0307  1C 05                              ; #051C ; 0, for adding to score.  parameter is score in hundreds
0309  9B 05                              ; #059B ; 1, clears and displays scores.  parameter 0 for p1, 1 for p2
030B  C6 05                              ; #05C6 ; 2, displays score.  0 for p1, 1 for p2, 2 for highscore
030D  E9 05                              ; #05E9 ; 3, used to draw text.  parameter is code for text to draw
030F  11 06                              ; #0611 ; 4, draws credits on screen if any are present
0311  2A 06                              ; #062A ; 5, parameter 0 adds bonus to player's score , parameter 1 update onscreen
bonus timer and play sound & change to red if below 1000
0313  B8 06                              ; #06B8 ; 6, draws remaining lives and level number.  parameter 1 to draw lives-1

; called from #02C7
; flashes 1UP or 2UP

0315  3A1A60    LD      A,($601A)       ; load A with timer constantly counts down from FF to 00 and then FF to 00 again and
again ... 1 count per frame
0318  47        LD      B,A             ; copy to B
0319  E60F      AND     #0F             ; mask bits, now between 0 and #F.  Is it zero ?
031B  C0        RET     NZ              ; no, return

031C  CF        RST     #8              ; if credits exist or someone is playing, continue.  else RET

031D  3A0D60    LD      A,(PlayerTurnA)       ; Load A with player # (0 for player 1, 1 for player 2)
0320  CD4703    CALL    #0347           ; Loads HL with location for score (either player 1 or 2)
0323  11E0FF    LD      DE,#FFE0        ; load DE with offset for each column
0326  CB60      BIT     4,B             ; test bit 4 of timer.  Is it zero ?
0328  2814      JR      Z,#033E         ; yes, skip ahead

032A  3E10      LD      A,#10           ; A := #10 = blank character
032C  77        LD      (HL),A          ; clear the text "1" from "1UP" or "2" from "2UP"
032D  19        ADD     HL,DE           ; add offset for next column
032E  77        LD      (HL),A          ; clear the text "U" from "1UP"
032F  19        ADD     HL,DE           ; next column
```

```
0330  77        LD      (HL),A          ; clear the text "P" from "1UP"
0331  3A0F60    LD      A,(TwoPlayerGame)       ; load A with # of players in game
0334  A7        AND     A               ; is this a 1 player game?
0335  C8        RET     Z               ; yes, return

0336  3A0D60    LD      A,(PlayerTurnA)       ; Load current player #
0339  EE01      XOR     #01             ; change player from 1 to 2 or from 2 to 1
033B  CD4703    CALL    #0347           ; Loads HL with location for score (either player 1 or 2)

033E  3C        INC     A               ; increase A, now it has the number of the player
033F  77        LD      (HL),A          ; draw player number on screen
0340  19        ADD     HL,DE           ; next column
0341  3625      LD      (HL),#25        ; draw "U" on screen
0343  19        ADD     HL,DE           ; next column
0344  3620      LD      (HL),#20        ; draw "P" on screen
0346  C9        RET                     ; return

; called from #033B

0347  214077    LD      HL,#7740        ; for player 1 HL gets #7740 VRAM address
034A  A7        AND     A               ; is this player 2?
034B  C8        RET     Z               ; no, then return

034C  21E074    LD      HL,#74E0        ; player 2 gets #74E0 location on screen
034F  C9        RET                     ; return

; called from #02CA
; checks for and handles extra life

0350  3A2D62    LD      A,(#622D)       ; load A with high score indicator
0353  A7        AND     A               ; has this player already been awarded extra life?
0354  C0        RET     NZ              ; yes, return

0355  21B360    LD      HL,#60B3        ; load HL with address for player 1 score
0358  3A0D60    LD      A,(PlayerTurnA)       ; load A with 0 when player 1 is up, 1 when player 2 is up
035B  A7        AND     A               ; player 1 up ?
035C  2803      JR      Z,#0361         ; yes, skip next step

035E  21B660    LD      HL,#60B6        ; else load HL with address of player 2 score

0361  7E        LD      A,(HL)          ; load A with a byte of the player's score
0362  E6F0      AND     #F0             ; mask bits
0364  47        LD      B,A             ; copy to B
0365  23        INC     HL              ; next score byte
0366  7E        LD      A,(HL)          ; load A with byte of player's score
0367  E60F      AND     #0F             ; mask bits
0369  B0        OR      B               ; mix together the 2 score bytes
036A  0F        RRCA
036B  0F        RRCA
036C  0F        RRCA
036D  0F        RRCA                    ; rotate right 4 times, this swaps the high and low bytes
036E  212160    LD      HL,ExtraLifeThreshold        ; load HL with score needed for extra life
0371  BE        CP      (HL)            ; compare player's score to high score.  is it greater?
0372  D8        RET     C               ; no, return

0373  3E01      LD      A,#01           ; A := 1
0375  322D62    LD      (#622D),A       ; store into extra life indicator
0378  212862    LD      HL,#6228        ; load HL with address of number of lives remaining
037B  34        INC     (HL)            ; increase
037C  C3B806    JP      #06B8           ; skip ahead and update # of lives on the screen

; called from #02DB
; checks timers and increments difficulty if needed

; [timer_6384++ ; IF timer_6384 != 256 THEN RETURN ; timer_6384 := 0 ; ]

037F  218463    LD      HL,#6384        ; load HL with timer address
0382  7E        LD      A,(HL)          ; load A with the timer
0383  34        INC     (HL)            ; increase the timer
0384  A7        AND     A               ; was the timer at zero?
0385  C0        RET     NZ              ; no, return

; [timer_6381++ ; IF (timer_6381/8) != INT(timer_6381/8) THEN RETURN]

0386  218163    LD      HL,#6381        ; load HL with timer
0389  7E        LD      A,(HL)          ; load A with timer value
038A  47        LD      B,A             ; copy to B
038B  34        INC     (HL)            ; increase timer
038C  E607      AND     #07             ; mask bits.  are right 3 bits == #000 ? does for every 8 steps of #6381
038E  C0        RET     NZ              ; no, return

; increase difficulty if not at max

; [ difficulty := (timer_6381 div 8) + level ; IF difficulty > 5 THEN difficulty := 5 ; RETURN]

038F  78        LD      A,B             ; load A with original timer value
0390  0F        RRCA                    ; roll right 3 times... (div 8)
0391  0F        RRCA
0392  0F        RRCA
0393  47        LD      B,A             ; store result into B
0394  3A2962    LD      A,(#6229)       ; load A with level number
0397  80        ADD     A,B             ; add B to A
0398  FE05      CP      #05             ; is this answer > 5 ?
039A  3802      JR      C,#039E         ; no, skip next step
```

```
039C  3E05      LD      A,#05           ; otherwise A := 5

039E  328063    LD      (#6380),A       ; store result into difficulty
03A1  c9        RET                     ; return to #02DE

; called from #02DE

03A2  3E03      LD      A,#03           ; A := 3 = 0011 binary
03A4  F7        RST     #30             ; only continue if level is girders or conveyors, else RET

03A5  D7        RST     #10             ; if mario is alive, continue, else RET

03A6  3A5063    LD      A,(#6350)       ; load A with 1 when an item has been hit with hammer
03A9  0F        RRCA                    ; has an item been hit with the hammer ?
03AA  D8        RET     C               ; yes, return, we don't do anything here while hammer hits occur

03AB  21B862    LD      HL,#62B8        ; load HL with this counter
03AE  35        DEC     (HL)            ; decrease.  at zero?
03AF  C0        RET     NZ              ; no, return

03B0  3604      LD      (HL),#04        ; yes, reset counter to 4
03B2  3AB962    LD      A,(#62B9)       ; load A with fire release indicator
03B5  0F        RRCA                    ; roll right.  carry?  Is there a fire onscreen or is it time to release a new fire?
03B6  D0        RET     NC              ; no, return

; a fire is onscreen or to be released

03B7  21296A    LD      HL,#6A29        ; load HL with sprite for fire above oil can
03BA  0640      LD      B,#40           ; B := #40
03BC  DD21A066  LD      IX,#66A0        ; load IX with fire array start ?
03C0  0F        RRCA                    ; roll A right again.  carry ?  Is it time to release another fire?
03C1  D2E403    JP      NC,#03E4        ; no, skip ahead, animate oilcan, reset timer and return

; release a fire

03C4  DD360902  LD      (IX+#09),#02    ; store 2 into sprite +9 indicator (size ???)
03C8  DD360A02  LD      (IX+#0A),#02    ; store 2 into sprite +#A indicator (size ???)
03CC  04        INC     B
03CD  04        INC     B               ; B := #42 = extra fire oilcan sprite value
03CE  CDF203    CALL    #03F2           ; randomly store B or B+1 into (HL) - animates the oilcan fire with extra fire
03D1  21BA62    LD      HL,#62BA        ; load HL with this timer.  usually it is set at #10 when a level begins
03D4  35        DEC     (HL)            ; decrease timer.  zero ?
03D5  C0        RET     NZ              ; no, return

; release a fire, or do something when fires already exist

03D6  3E01      LD      A,#01           ; A := 1
03D8  32B962    LD      (#62B9),A       ; store into fire release indicator
03DB  32A063    LD      (#63A0),A       ; store into other fireball release indicator

03DE  3E10      LD      A,#10           ; A := #10
03E0  32BA62    LD      (#62BA),A       ; reset timer back to #10
03E3  C9        RET                     ; return

03E4  DD360902  LD      (IX+#09),#02    ; set +9 to 2 (size ???)
03E8  DD360A00  LD      (IX+#0A),#00    ; set +A to 0 (size ???)
03EC  CDF203    CALL    #03F2           ; randomly store B or B+1 into (HL) - animates the oilcan fire
03EF  C3DE03    JP      #03DE           ; skip back, reset timer, and return

; called from #03CE and #03EC above
; animates the oilcan fire

03F2  70        LD      (HL),B          ; store B into (HL) - set the oilcan fire sprite
03F3  3A1960    LD      A,(RngTimer2)       ; load A with random number
03F6  0F        RRCA                    ; rotate right.  carry ?
03F7  D8        RET     C               ; yes, return

03F8  04        INC     B               ; else increase B
03F9  70        LD      (HL),B          ; store B into (HL) - set the oilcan fire sprite with higher value
03FA  C9        RET                     ; return

; called from main routine at #19B0
; animates kong, checks for kong beating chest, animates girl and her screams for help

03FB  3A2762    LD      A,(#6227)       ; load A with screen number
03FE  FE02      CP      #02             ; are we on the conveyors?
0400  C21304    JP      NZ,#0413        ; no, skip ahead

; conveyors

0403  210869    LD      HL,#6908        ; load HL with kongs sprite start
0406  3AA363    LD      A,(#63A3)       ; load A with kongs direction
0409  4F        LD      C,A             ; copy to C for subroutine below
040A  FF        RST     #38             ; move kong
040B  3A1069    LD      A,(#6910)       ; load A with kong's X position
040E  D63B      SUB     #3B             ; subtract #3B (59 decimal)
0410  32B763    LD      (#63B7),A       ; store into kong's position

; #6390 - counts from 0 to 7F periodically
; #6391 - is 0, then changed to 1 when timer in #6390 is counting up

0413  3A9163    LD      A,(#6391)       ; load A with indicator
0416  A7        AND     A               ; == 0 ?
0417  C22604    JP      NZ,#0426        ; no, skip next 5 steps
```

```
041A  3A1A60    LD      A,(FrameCounter)       ; else load A with this clock counts down from #FF to 00 over and over...
041D  A7        AND     A               ; == 0 ?
041E  C28604    JP      NZ,#0486        ; no, skip ahead

0421  3E01      LD      A,#01           ; else A := 1
0423  329163    LD      (#6391),A       ; store into indicator

0426  219063    LD      HL,#6390        ; load HL with timer
0429  34        INC     (HL)            ; increase
042A  7E        LD      A,(HL)          ; load A with timer value
042B  FE80      CP      #80             ; == #80 ?
042D  CA6404    JP      Z,#0464         ; yes, skip ahead

0430  3A9363    LD      A,(#6393)       ; else get barrel deployment
0433  A7        AND     A               ; is a barrel deployment in progress?
0434  C28604    JP      NZ,#0486        ; yes, jump ahead

0437  7E        LD      A,(HL)          ; else load A with timer
0438  47        LD      B,A             ; copy to B
0439  E61F      AND     #1F             ; mask bits, now == 0 ?
043B  C28604    JP      NZ,#0486        ; no, skip ahead

043E  21CF39    LD      HL,#39CF        ; else load HL with start of table data
0441  CB68      BIT     5,B             ; is bit 5 turned on timer ?  (1/8 chance???)
0443  2003      JR      NZ,#0448        ; no, skip ahead

; kong is beating his chest

0445  21F739    LD      HL,#39F7        ; start of table data
0448  CD4E00    CALL    #004E           ; update kong's sprites
044B  3E03      LD      A,#03           ; load sound duration of 3
044D  328260    LD      (#6082),A       ; play boom sound using sound buffer

0450  3A2762    LD      A,(#6227)       ; load A with screen number
0453  0F        RRCA                    ; is this the girders or the elevators ?
0454  D27804    JP      NC,#0478        ; no, skip ahead

0457  0F        RRCA                    ; else is this the rivets ?
0458  DA8604    JP      C,#0486         ; yes, skip ahead

; else pie factory

045B  210B69    LD      HL,#690B        ; load HL with start of Kong sprite data
045E  0EFC      LD      C,#FC           ; C := #FC.  used in sub below to move kong by -4
0460  FF        RST     #38             ; move kong
0461  C38604    JP      #0486           ; skip ahead

; arrive here from #042D when timer in #6390 is #80

0464  AF        XOR     A               ; A := 0
0465  77        LD      (HL),A          ; clear timer
0466  23        INC     HL              ; increase address to #6391
0467  77        LD      (HL),A          ; clear this one too
0468  3A9363    LD      A,(#6393)       ; Load Barrel deployment indicator
046B  A7        AND     A               ; is a deployment in progress?
046C  C28604    JP      NZ,#0486        ; yes, jump ahead

046F  215C38    LD      HL,#385C        ; else load HL with start of table data for kongs sprites
0472  CD4E00    CALL    #004E           ; update kong's sprites
0475  C35004    JP      #0450           ; jump back

; arrive here from #0454 when on rivets and conveyors
; moves kong, updates girl and her screams for help

0478  210869    LD      HL,#6908        ; load HL with start of kong sprite X position
047B  0E44      LD      C,#44           ; set offset to #44, used only on rivets
047D  0F        RRCA                    ; roll screen number right (again).  is this the conveyors screen?
047E  D28504    JP      NC,#0485        ; no, skip next 2 steps

0481  3AB763    LD      A,(#63B7)       ; load A with kong's position
0484  4F        LD      C,A             ; copy to C for sub below, controls position of kong

0485  FF        RST     #38             ; move kong to his position

0486  3A9063    LD      A,(#6390)       ; load A with timer
0489  4F        LD      C,A             ; copy to C
048A  112000    LD      DE,#0020        ; DE := #20, used for offset in call at #04A6
048D  3A2762    LD      A,(#6227)       ; load A with screen number
0490  FE04      CP      #04             ; are we on the rivets level?
0492  CABE04    JP      Z,#04BE         ; yes, jump ahead to handle

0495  79        LD      A,C             ; load A with the timer
0496  A7        AND     A               ; == 0 ?
0497  CAA104    JP      Z,#04A1         ; yes, skip next 3 steps

049A  3EEF      LD      A,#EF           ; else A := #EF
049C  CB71      BIT     6,C             ; is bit 6 of the timer set ?
049E  C2A304    JP      NZ,#04A3        ; no, skip next step

04A1  3E10      LD      A,#10           ; A := #10

04A3  21C475    LD      HL,#75C4        ; load HL with address of a location in video RAM where girl yells "HELP"
04A6  CD1405    CALL    #0514           ; update girl yelling "HELP"
04A9  3A0569    LD      A,(#6905)       ; load A with girl's sprite
```

```
04AC  320569    LD      (#6905),A       ; store girl's sprite
04AF  CB71      BIT     6,C             ; is bit 6 of the timer set ?
04B1  C8        RET     Z               ; yes, return

04B2  47        LD      B,A             ; else B := A
04B3  79        LD      A,C             ; A := C (timer)
04B4  E607      AND     #07             ; mask bits, now betwen 0 and 7.  zero ?
04B6  C0        RET     NZ              ; no, return

04B7  78        LD      A,B             ; restore A which has girl's sprite
04B8  EE03      XOR     #03             ; toggle bits 0 and 1
04BA  320569    LD      (#6905),A       ; store into girl's sprite
04BD  C9        RET                     ; return to #19B3 - main routine

; arrive here when we are on the rivets level

04BE  3E10      LD      A,#10           ; A := #10 = code for clear space
04C0  212376    LD      HL,#7623        ; load HL with video RAM for girl location
04C3  CD1405    CALL    #0514           ; clear the "help" the girl yells on the left side
04C6  218375    LD      HL,#7583        ; load HL with video RAM right of girl
04C9  CD1405    CALL    #0514           ; clear the "help" the girl yells on the right side
04CC  CB71      BIT     6,C             ; check timer bit 6.  zero?
04CE  CA0905    JP      Z,#0509         ; yes, skip ahead

04D1  3A0362    LD      A,(#6203)       ; load A with mario X position
04D4  FE80      CP      #80             ; is mario on left side of screen ?
04D6  D2F104    JP      NC,#04F1        ; yes, skip ahead

04D9  3EDF      LD      A,#DF           ; else A := #DF
04DB  212376    LD      HL,#7623        ; load HL with video RAM for girl location
04DE  CD1405    CALL    #0514           ; draw "help" on the left side

04E1  3A0169    LD      A,(#6901)       ; load A with sprite used for girl
04E4  F680      OR      #80             ; set bit 7
04E6  320169    LD      (#6901),A       ; store into sprite used for girl
04E9  3A0569    LD      A,(#6905)       ; load A with girl's sprite
04EC  F680      OR      #80             ; set bit 7
04EE  C3AC04    JP      #04AC           ; jump back and animate girl

04F1  3EEF      LD      A,#EF           ; A := #EF
04F3  218375    LD      HL,#7583        ; load HL with video RAM for girl location
04F6  CD1405    CALL    #0514           ; draw "help" on the right side

04F9  3A0169    LD      A,(#6901)       ; load A with sprite used for girl
04FC  E67F      AND     #7F             ; mask bits, turns off bit 7
04FE  320169    LD      (#6901),A       ; store result
0501  3A0569    LD      A,(#6905)       ; load A with girl's sprite
0504  E67F      AND     #7F             ; mask bits, turns off bit 7
0506  C3AC04    JP      #04AC           ; jump back and store into girl's sprite and check for animation and RET

; jump from #04CE

0509  3A0362    LD      A,(#6203)       ; load A with mario X position
050C  FE80      CP      #80             ; is mario on left side of screen?
050E  D2F904    JP      NC,#04F9        ; yes, jump back

0511  C3E104    JP      #04E1           ; else jump back

;
; this sub gets called a lot
; HL is preloaded with an address of video RAM ?
; DE is preloaded with an offset to add
; A is preloaded with a value to write
; writes A into HL, A-1 into HL+DE, A-2 into HL+2DE
;

0514  0603      LD      B,#03           ; for B = 1 to 3

0516  77        LD      (HL),A          ; store A into memory
0517  19        ADD     HL,DE           ; next memory
0518  3D        DEC     A               ; decrease A
0519  10FB      DJNZ    #0516           ; next B

051B  C9        RET                     ; return

;
; Task #0, arrive from jump at #0306
; adds score
; parameter in A is the score to add in hundreds
;

051C  4F        LD      C,A             ; copy score to C
051D  CF        RST     #8              ; only continue if credits exist or someone is playing, else RET
051E  CD5F05    CALL    #055F           ; load DE with address of player score
0521  79        LD      A,C             ; load score
0522  81        ADD     A,C             ; double
0523  81        ADD     A,C             ; triple
0524  4F        LD      C,A             ; C is now 3 times A for use in the scoring table
0525  212935    LD      HL,#3529        ; #3529 holds table data for scoring
0528  0600      LD      B,#00           ; B := 0
052A  09        ADD     HL,BC           ; add offset for scoring table
052B  A7        AND     A               ; clear carry flag
052C  0603      LD      B,#03           ; for B = 1 to 3

052E  1A        LD      A,(DE)          ; load A with current score
```

```
052F  8E        ADC     A,(HL)          ; add the amount the player just scored
0530  27        DAA                     ; decimal adjust
0531  12        LD      (DE),A          ; store result in score
0532  13        INC     DE              ; next byte of score
0533  23        INC     HL              ; next byte of score to add
0534  10F8      DJNZ    #052E           ; Next B

0536  D5        PUSH    DE              ; save DE
0537  1B        DEC     DE              ; DE is now the last byte of score
0538  3A0D60    LD      A,(PlayerTurnA)       ; 0 for player 1, 1 for player 2
053B  CD6B05    CALL    #056B           ; update onscreen score
053E  D1        POP     DE              ; restore DE
053F  1B        DEC     DE              ; decrement
0540  21BA60    LD      HL,#60BA        ; load HL with high score address
0543  0603      LD      B,#03           ; for B = 1 to  3

0545  1A        LD      A,(DE)          ; load A with player score
0546  BE        CP      (HL)            ; compare to high score
0547  D8        RET     C               ; if less, then return

0548  C25005    JP      NZ,#0550        ; if greater, then skip ahead to update

054B  1B        DEC     DE              ; next score byte
054C  2B        DEC     HL              ; next highscore byte
054D  10F6      DJNZ    #0545           ; next B

054F  C9        RET                     ; return

0550  CD5F05    CALL    #055F           ; load DE with address of player score
0553  21B860    LD      HL,#60B8        ; load HL with high score address

0556  1A        LD      A,(DE)          ; load A with player score byte
0557  77        LD      (HL),A          ; store into high score byte
0558  13        INC     DE              ; next address
0559  23        INC     HL              ; next address
055A  10FA      DJNZ    #0556           ; next B

055C  C3DA05    JP      #05DA           ; skip ahead to update high score onscreen

; called from #051E and #0550
; loads DE with address of current player's score

055F  11B260    LD      DE,#60B2        ; load DE with player 1 score
0562  3A0D60    LD      A,(PlayerTurnA)       ; load number of players
0565  A7        AND     A               ; is this player 2 ?
0566  C8        RET     Z               ; no, return

0567  11B560    LD      DE,#60B5        ; else load DE with player 2 score
056A  C9        RET                     ; return

; called from #053B
; update onscreen score

056B  DD218177  LD      IX,#7781        ; load IX with the start of the score in video RAM (100,000's place)
056F  A7        AND     A               ; is this player 1?
0570  280A      JR      Z,#057C         ; Yes, jump ahead

0572  DD212175  LD      IX,#7521        ; else load IX with #7521 - the start of player 2 score (100,000's place)
0576  1804      JR      #057C           ; skip next step

0578  DD214176  LD      IX,#7641        ; #7641 is the start of high score 100,000 place

057C  EB        EX      DE,HL           ; DE <> HL
057D  11E0FF    LD      DE,#FFE0        ; offset is inverse of 20 ?  to add to next column in scoreboard
0580  010403    LD      BC,#0304        ; For B = 1 to 3

; can arrive here from #0627 to draw number of credits

0583  7E        LD      A,(HL)          ; get digit
0584  0F        RRCA
0585  0F        RRCA
0586  0F        RRCA
0587  0F        RRCA                    ; rotate right 4 times
0588  CD9305    CALL    #0593           ; draw to screen
058B  7E        LD      A,(HL)          ; get digit
058C  CD9305    CALL    #0593           ; draw to screen
058F  2B        DEC     HL              ; next digit
0590  10F1      DJNZ    #0583           ; Next B

0592  C9        RET                     ; return

; called from #0588 and #058C above

0593  E60F      AND     #0F             ; mask out left 4 bits of A
0595  DD7700    LD      (IX+#00),A      ; store A on screen
0598  DD19      ADD     IX,DE           ; adjust to next location
059A  C9        RET                     ; return

;
; task #1
; called from #0306
; parameter is 0 when 1 player game, 1 when 2 player game
; clears score and runs task #2 as well
;
```

```
059B  FE03      CP      #03             ; task parameter < 3 ?
059D  D2BD05    JP      NC,#05BD        ; yes, skip ahead [when would it do this???  A always 0 or 1 ???]

; #60B2, #60B3, #60B4 - player 1 score

; #60B5, #60B6, #60B7 - player 2 score

05A0  F5        PUSH    AF              ; save AF
05A1  21B260    LD      HL,#60B2        ; load HL with player 1 score
05A4  A7        AND     A               ; parameter == 0 ?
05A5  CAAB05    JP      Z,#05AB         ; yes, skip next step

05A8  21B560    LD      HL,#60B5        ; else load HL with player 2 score
05AB  FE02      CP      #02             ; parameter == 2 ? [when would it do this ??? A always 0 or 1 ??? ]
05AD  C2B305    JP      NZ,#05B3        ; no, skip next step

05B0  21B860    LD      HL,#60B8        ; load HL with high score

05B3  AF        XOR     A               ; A := 0
05B4  77        LD      (HL),A          ; clear score
05B5  23        INC     HL              ; next score memory
05B6  77        LD      (HL),A          ; clear score
05B7  23        INC     HL              ; next score memory
05B8  77        LD      (HL),A          ; clear score
05B9  F1        POP     AF              ; restore AF
05BA  C3C605    JP      #05C6           ; jump ahead to task 2

; never arrive here ???

05BD  3D        DEC     A               ; decrease A
05BE  F5        PUSH    AF              ; save AF
05BF  CD9B05    CALL    #059B           ; ???  call myself ???
05C2  F1        POP     AF              ; restore AF
05C3  C8        RET     Z               ; return if Zero

05C4  18F7      JR      #05BD           ; else loop again

;
; task #2 - displays score
; called from #0306 and at end of task #1, from #05BA
; parameter is 0 for player 1, 1 for player 2, and 3 for high score
;

05C6  FE03      CP      #03             ; task parameter == 3 ?
05C8  CAE005    JP      Z,#05E0         ; yes, skip ahead to handle high score

05CB  11B460    LD      DE,#60B4        ; load DE with player 1 score
05CE  A7        AND     A               ; parameter == 0 ? (1 player game)
05CF  CAD505    JP      Z,#05D5         ; yes, skip next step

05D2  11B760    LD      DE,#60B7        ; else load DE with player 2 score

05D5  FE02      CP      #02             ; parameter == 2 ?
05D7  C26B05    JP      NZ,#056B        ; no, jump back and display score

; arrive here from #055C

05DA  11BA60    LD      DE,#60BA        ; yes, load DE with high score
05DD  C37805    JP      #0578           ; jump back and display high score

05E0  3D        DEC     A               ; decrease A
05E1  F5        PUSH    AF              ; save AF
05E2  CDC605    CALL    #05C6           ; call this sub again for the lower parameter
05E5  F1        POP     AF              ; restore AF.  A == 0 ?  are we done?
05E6  C8        RET     Z               ; yes, return

05E7  18F7      JR      #05E0           ; else loop back again

; task #3
; draws text to screen
; called from #0306 with code for text to draw in A

05E9  214B36    LD      HL,#364B        ; start of table data
05EC  87        ADD     A,A             ; double the parameter
05ED  F5        PUSH    AF              ; save AF to stack
05EE  E67F      AND     #7F             ; mask bits
05F0  5F        LD      E,A             ; copy to E
05F1  1600      LD      D,#00           ; D := 0
05F3  19        ADD     HL,DE           ; add to table to get pointer
05F4  5E        LD      E,(HL)          ; load E with first byte from table
05F5  23        INC     HL              ; next table entry
05F6  56        LD      D,(HL)          ; load D with 2nd byte from table
05F7  EB        EX      DE,HL           ; DE <> HL
05F8  5E        LD      E,(HL)          ; load E with 1st byte from dereferenced table
05F9  23        INC     HL              ; next table entry
05FA  56        LD      D,(HL)          ; load D with 2ndy byte from derefernced table
05FB  23        INC     HL              ; next table entry
05FC  01E0FF    LD      BC,#FFE0        ; load BC with offset to print characters across
05FF  EB        EX      DE,HL           ; DE <> HL.  HL now has screen destination, DE has table pointer

0600  1A        LD      A,(DE)          ; load A with table data
0601  FE3F      CP      #3F             ; end code reached?
0603  CA2600    JP      Z,#0026         ; yes, return to program.  This will effectively RET twice

0606  77        LD      (HL),A          ; draw letter to screen
```

```
0607  F1        POP     AF              ; restore AF from stack.  is there a carry?
0608  3002      JR      NC,#060C        ; no, skip next step

060A  3610      LD      (HL),#10        ; yes, write a blank space to the screen

060C  F5        PUSH    AF              ; save AF
060D  13        INC     DE              ; next table data
060E  09        ADD     HL,BC           ; add screen offset for next column
060F  18EF      JR      #0600           ; loop again

;
; task #4
; jump from #0306
; draws credits on screen if any are present
;

0611  3A0760    LD      A,(NoCredits)   ; 1 when no credits have been inserted; 0 if any credits exist
0614  0F        RRCA                    ; credits in game ?
0615  D0        RET     NC              ; yes, return

; called from #08F0

0616  3E05      LD      A,#05           ; load text code for "CREDIT"
0618  cde905    CALL    #05E9           ; draw to screen
061B  210160    LD      HL,NumCredits   ; load HL with pointer to number of credits
061E  11E0Ff    LD      DE,#ffe0        ; load DE with #ffe0 = offset for columns?
0621  dd21Bf74  LD      IX,#74Bf        ; load IX with screen address to draw
0625  0601      LD      B,#01           ; B := 1
0627  c38305    JP      #0583           ; jump back to draw number of credits on screen and return

;
; task #5
; called from #0306
; parameter 0 = adds bonus to player's score
; parameter 1 = update onscreen bonus timer and play sound & change to red if below 1000

062A  A7        AND     A               ; parameter == 0 ?
062B  cA9106    JP      Z,#0691         ; yes, skip ahead and add bonus to player's score

062E  3A8C63    LD      A,(#638C)       ; else load onscreen timer
0631  A7        AND     a               ; timer == 0 ?
0632  c2A806    JP      NZ,#06A8        ; no, jump ahead

0635  3Ab863    LD      A,(#63B8)       ; else load A with timer expired indicator
0638  A7        AND     a               ; has timer expired ?
0639  c0        RET     NZ              ; yes, return

; the following code sets up the on screen timer initial value

063A  3Ab062    LD      A,(#62B0)       ; load a with value from #62B0 (expects a decimal number here)
063D  010A00    LD      BC,#000A        ; B := 0, C := #0A (10 decimal)

0640  04        INC     b               ; increment b
0641  91        SUB     c               ; subtract 10 decimal from A
0642  c24006    JP      NZ,#0640        ; loop again if not zero; counts how many tens there are

0645  78        LD      A,b             ; load a with the number of tens in the counter
0646  07        RLCA                    ; rotate left (x2)
0647  07        RLCA                    ; rotate left (x4)
0648  07        RLCA                    ; rotate left (x8)
0649  07        RLCA                    ; rotate left (x16)
064A  328C63    LD      (#638C),A       ; load on screen timer with result.  hex value converts to decimal.


064D  214A38    LD      HL,#384A        ; load HL with #384A - table data
0650  116574    LD      DE,#7465        ; load DE with #7465 - screen location for bonus timer
0653  3E06      LD      A,#06           ; For A = 1 to 6

; draws timer box on screen with all zeros

0655  dd211D00  LD      IX,#001D        ; load IX with #001D offset used for each column
0659  010300    LD      BC,#0003        ; counter := 3
065C  edb0      LDIR                    ; transfer (HL) to (DE) 3 times
065E  dd19      ADD     IX,DE           ; add offset DE to IX
0660  dde5      PUSH    IX              ;
0662  d1        POP     de              ; load DE with IX
0663  3D        DEC     a               ; decrease counter
0664  c25506    JP      NZ,#0655        ; loop again if not zero

; check to see if timer is below 1000

0667  3A8C63    LD      A,(#638C)       ; load a with value from on screen timer

066A  4f        LD      c,A             ; copy to C
066b  e60F      AND     #0F             ; zeroes out left 4 bits
066D  47        LD      B,A             ; store result in B
066E  79        LD      A,C             ; restore a with original value from timer
066f  0F        RRCA                    ; rotate right 4 times.  divides by 16
0670  0F        RRCA
0671  0F        RRCA
0672  0F        RRCA
0673  e60F      AND     #0F             ; and with #0F - zero out left 4 bits
0675  c28906    JP      NZ,#0689        ; jump if not zero to #0689

; arrive here when timer runs below 1000
```

```
0678  3E03      LD      A,#03           ; else load A with warning sound
067A  328960    LD      (#6089),A       ; set warning sound
067D  3E70      LD      A,#70           ; A := #70 = color code for red?
067f  328674    LD      (#7486),A       ; store A into #7486 = paint score red (MSB) ?
0682  32A674    LD      (#74A6),A       ; store A into #74A6 = paint score red (LSB) ?
0685  80        ADD     A,b             ; A = A + B
0686  47        LD      B,A             ; B := A
0687  3E10      LD      A,#10           ; A = #10 = code for blank space

0689  32E674    LD      (#74E6),A       ; draw timer to screen (MSB)
068C  78        LD      A,b             ; A := B
068D  32C674    LD      (#74C6),A       ; draw timer to screen (LSB)
0690  c9        RET                     ; return

;
; continuation of task #5 when parameter = 0 from #062B
; adds bonus to player's score
;

0691  3A8C63    LD      A,(#638C)       ; load A with timer value from #638C
0694  47        LD      B,A             ; copy to B
0695  e60F      AND     #0F             ; and with #0F - mask four left bits.  how has low byte of bonus
0697  c5        PUSH    BC              ; save BC
0698  cd1C05    CALL    #051C           ; add to score
069b  c1        POP     BC              ; restore BC
069C  78        LD      A,b             ; load A with timer
069D  0F        RRCA                    ; rotate right 4 times
069E  0F        RRCA
069f  0F        RRCA
06A0  0F        RRCA
06A1  e60F      AND     #0F             ; mask four left bits to zero
06A3  c60A      ADD     A,#0A           ; add #0A (10 decimal) - this indicates scores of thousands to add
06A5  c31C05    JP      #051C           ; jump to add score (thousands) and RET

; jump here from #0632

06A8  d601      SUB     #01             ; subtract 1 from bonus timer
06Aa  2005      JR      NZ,#06b1        ; If not zero, skip next 2 steps

; timer at zero

06Ac  21B863    LD      HL,#63B8        ; load HL with mario dead flag
06Af  3601      LD      (HL),#01        ; store 1 - mario will die soon on next timer click

06b1  27        DAA                     ; Decimal adjust
06b2  328C63    LD      (#638C),A       ; store A into timer
06b5  c36A06    JP      #066A           ; jump back

;
; task #6
; called from #01DC and #0306.  also jump here from #037C after high score has been exceeded
; parameter used to subtract the number of lives to draw
;

06B8  4F        LD      C,A             ; load C with the task parameter
06B9  CF        RST     #8              ; is the game being played or credits exists?  If so, continue.  Else RET

06BA  0606      LD      B,#06           ; For B = 1 to 6
06BC  11E0FF    LD      DE,#FFE0        ; load DE with offset for next column
06BF  218377    LD      HL,#7783        ; load HL with screen location where mario extra lives drawn

06C2  3610      LD      (HL),#10        ; clear this area of screen
06C4  19        ADD     HL,DE           ; add offset for next column
06C5  10FB      DJNZ    #06C2           ; next B

06C7  3A2862    LD      A,(#6228)       ; load A with number of lives remaining
06CA  91        SUB     C               ; subtract the task parameter.  zero lives to draw?
06CB  CAD706    JP      Z,#06D7         ; yes, skip next 5 steps

06CE  47        LD      B,A             ; For B = 1 to A
06CF  218377    LD      HL,#7783        ; load HL with screen location to draw remaining lives

06D2  36FF      LD      (HL),#FF        ; draw the extra mario
06D4  19        ADD     HL,DE           ; add offset for next column
06D5  10FB      DJNZ    #06D2           ; next B

06D7  210375    LD      HL,#7503        ; load HL with screen location for "L="
06DA  361C      LD      (HL),#1C        ; draw "L"
06DC  21E374    LD      HL,#74E3        ; next location
06DF  3634      LD      (HL),#34        ; draw "="
06E1  3A2962    LD      A,(#6229)       ; load A with level #
06E4  fe64      CP      #64             ; level < #64 (100 decimal) ?
06E6  3805      JR      c,#06Ed         ; yes, skip next 2 steps

06E8  3E63      LD      A,#63           ; otherwise A := #63 (99 decimal)
06Ea  322962    LD      (#6229),A       ; store into level #

06Ed  010Aff    LD      BC,#ff0A        ; B: = #FF, C := #0A (10 decimal)

06f0  04        INC     b               ; increment B
06f1  91        SUB     c               ; subtract 10 decimal
06f2  d2f006    JP      NC,#06f0        ; not carry, loop again (counts tens)

06f5  81        ADD     A,C             ; add 10 back to A to get a number from 0 to 9
```

```
06f6  32A374    LD      (#74A3),A       ; draw level to screen (low byte)
06f9  78        LD      A,b             ; load a with b (number of tens)
06fa  32C374    LD      (#74C3),A       ; draw level to screen (high byte)
06FA  C36F13    JP      #136F           ; jump to additional code to draw level and sub level on screen
06fd  c9        RET                     ; return

; start of main routine when playing a game
; arrive here from #00C9

06FE  3A0A60    LD      A,(GameMode2)       ; load A with game mode2
0701  EF        RST     #28             ; jump based on what the game state is

0702  86 09                             ; (0) #0986     ; game start = clears screen, clears sounds, sets screen flip if
needed
0704  AB 09                             ; (1) #09AB     ; copy player data, set screen, set next game mode based on number of
players
0706  D6 09                             ; (2) #09D6     ; clears palettes, draws "PLAYER <I>", draws player2 score, draws
"2UP" (2 player game only)
0708  FE 09                             ; (3) #09FE     ; copy player data into correct area (2 player game only)
070A  1B 0A                             ; (4) #0A1B     ; clears palletes, draws "PLAYER <II>", update player2 score, draw
"2UP" to screen (2 player game only)
070C  37 0A                             ; (5) #0A37     ; updates high score, player score, remaining lives, level, 1UP
070E  63 0A                             ; (6) #0A63     ; clears screen and sprites, check for intro screen to run
0710  76 0A                             ; (7) #0A76     ; kong clims ladders and scary music played
0712  DA 0B                             ; (8) #0BDA     ; draw goofy kongs, how high can you get, play music
0714  00 00                             ; (9)           ; unused
0716  91 0C                             ; (A) #0C91     ; clears screen, update timers, draws current screen, sets background
music
0718  3C 12                             ; (B) #123C     ; set initial mario sprite position and draw remaining lives and
level
071A  7A 19                             ; (C) #197A     ; for when playing a game.  this is the main routine
071C  7C 12                             ; (D) #127C     ; mario died.  handle mario dying animations
071E  F2 12                             ; (E) #12F2     ; clear sounds, decrease life, check for and handle game over
0720  44 13                             ; (F) #1344     ; clear sounds, clear game start flag, draw game over if needed PL2,
set game mode2 accordingly
0722  8F 13                             ; (10) #138F    ; check for game over status on a 2 player game
0724  A1 13                             ; (11) #13A1    ; check for game over status on a 2 player game
0726  AA 13                             ; (12) #13AA    ; flip screen if needed, reset game mode2 to zero, set player 2
0728  BB 13                             ; (13) #13BB    ; set player 1, reset game mode2 to zero, set screen flip to not
flipped
072A  1E 14                             ; (14) #141E    ; draw credits on screen, clears screen and sprites, checks for high
score, flips screen if necessary
072C  86 14                             ; (15) #1486    ; player enters initials in high score table
072E  15 16                             ; (16) #1615    ; handle end of level animations
0730  6B 19                             ; (17) #196B    ; clear screen and all sprites, set game mode2 to #12 for player1 or
#13 for player2
0732  00 00 00 00 00 00 00 00 00 00                     ; unused

; arrive from #00C9 when attract mode starts

073C  210A60    LD      HL,GameMode2        ; load HL with game mode2 address
073F  3A0160    LD      A,(NumCredits)  ; load A with number of credits
0742  A7        AND     A               ; any credits exist ?
0743  C25C07    JP      NZ,#075C        ; yes, skip ahead, zero out game mode2, increase game mode1, and RET

0746  7E        LD      A,(HL)          ; else load A with game mode2
0747  EF        RST     #28             ; jump based on A

0748  79 07                     0       ; #0779         ; clear screen, set color palettes, draw attract mode text and high
score table,
                                                        ; [continued] increase game mode2, clear sprites, ; draw "1UP" on
screen , draws number of coins needed for play
074A  63 07                     1       ; #0763         ;
074C  3C 12                     2       ; #123C         ; set initial mario sprite position and draw remaining lives and
level
074E  77 19                     3       ; #1977         ; set artificial input for demo play [change to #197A to enable
playing in demo part 1/2]
0750  7C 12                     4       ; #127C         ; handle mario dying animations
0752  C3 07                     5       ; #07C3         ; clears the screen and sprites and increase game mode2
0754  CB 07                     6       ; #07CB         ; handle intro splash screen ?
0756  4B 08                     7       ; #084B         ; counts down a timer then resets game mode2 to 0

0758  00 00 00 00                       ; unused

; arrive from #0743 when credits exist

075C  3600      LD      (HL),#00        ; set game mode2 to zero
075E  210560    LD      HL,GameMode1    ; load HL with game mode1
0761  34        INC     (HL)            ; increase
0762  C9        RET                     ; return

; arrive here from #0747 during attract mode when GameMode2 == 1

0763  E7        RST     #20             ; only continue here once per frame, else RET

0764  AF        XOR     A               ; A := 0
0765  329263    LD      (#6392),A       ; clear barrel deployment indicator
0768  32A063    LD      (#63A0),A       ; clear fireball release indicator
076B  3E01      LD      A,#01           ; A := 1
076D  322762    LD      (#6227),A       ; load screen number with 1
0770  322962    LD      (#6229),A       ; load level # with 1
0773  322862    LD      (#6228),A       ; load number of lives with 1
0776  C3920C    JP      #0C92           ; skip ahead

; arrive from #0747 when GameMode2 == 0
```

```
; clear screen, set color palettes, draw attract mode text and high score table, increase game mode2, clear sprites, ; draw
"1UP" on screen , draws number of coins needed for play

0779  21867D    LD      HL,REG_PALETTE_A
077C  3600      LD      (HL),#00        ; clear palette bank selector
077E  23        INC     HL
077F  3600      LD      (HL),#00        ; clear palette bank selector
0781  111B03    LD      DE,#031B        ; load task data for text "INSERT COIN"
0784  CD9F30    CALL    #309F           ; insert task to draw text
0787  1C        INC     E               ; load task data for text "PLAYER    COIN"
0788  CD9F30    CALL    #309F           ; insert task to draw text
078B  CD6509    CALL    #0965           ; draws credits on screen if any are present and displays high score table
078E  210960    LD      HL,WaitTimerMSB ; load HL with timer address
0791  3602      LD      (HL),#02        ; set timer at 2
0793  23        INC     HL              ; load HL with game mode2
0794  34        INC     (HL)            ; increase
0795  CD7408    CALL    #0874           ; clears the screen and sprites
0798  CD530A    CALL    #0A53           ; draw "1UP" on screen
079B  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players in game
079E  FE01      CP      #01             ; 2 player game?
07A0  CCEE09    CALL    Z,#09EE         ; yes, skip ahead to handle

07A3  ED5B2260  LD      DE,(CoinsPerCredit)      ; D := CoinsPer2Credits; E := CoinsPerCredit
07A7  216C75    LD      HL,#756C        ; load HL with screen RAM location
07AA  CDAD07    CALL    #07AD           ; run this sub below twice

07AD  73        LD      (HL),E          ; draw to screen number of coins needed for 1 player game
07AE  23        INC     HL              ;
07AF  23        INC     HL              ; next screen location 2 rows down
07B0  72        LD      (HL),D          ; draw to screen number of coins needed for 2 player game
07B0  00        NOP                     ; no operation - two player game removed, so don't display message
07B1  7A        LD      A,D             ; A := D
07B2  D60A      SUB     #0A             ; subtract #A (10 decimal). result == 0 ?
07B4  C2BC07    JP      NZ,#07BC        ; no, skip next 3 steps

07B7  77        LD      (HL),A          ; else draw this zero to screen
07B8  3C        INC     A               ; increase A, A := 1 now
07B9  328E75    LD      (#758E),A       ; draw 1 to screen in front of the zero, so it draws "10" credits needed for 2
players

07BC  110102    LD      DE,#0201        ; D := 2, E := 1, used for next loop for 1 player and 2 players
07BF  218C76    LD      HL,#768C        ; set screen location to draw for next loop if needed
07C2  C9        RET                     ; return

; arrive from #0747 when GameMode2 == 5

07C3  CD7408    CALL    #0874           ; clears the screen and sprites
07C6  210A60    LD      HL,GameMode2       ; load HL with game mode 2
07C9  34        INC     (HL)            ; increase game mode2
07CA  C9        RET                     ; return

; arrive from jump at #0747 when GameMode2 == 6

07CB  3A8A63    LD      A,(#638A)       ; load A with kong screen flash counter
07CE  FE00      CP      #00             ; == 0 ?  time to flash?
07D0  C22D08    JP      NZ,#082D        ; no, skip ahead : load C with (#638B), decreases #638A, loads A with (#638A); loads
C with #638B, decreases #638A returns to #07DA

07D3  3E60      LD      A,#60           ; else A := #60
07D5  328A63    LD      (#638A),A       ; store into kong screen flash counter
07D8  0E5F      LD      C,#5F           ; C := #5F

; can arrive here from jump at #0838

07DA  FE00      CP      #00             ; A == 0 ? [why not AND A ?]
07DC  CA3B08    JP      Z,#083B         ; yes, skip ahead

07DF  21867D    LD      HL,REG_PALETTE_A        ; load pallete bank
07E2  3600      LD      (HL),#00        ; clear palette bank selector
07E4  79        LD      A,C             ; A := C
07E5  CB07      RLC     A               ; rotate left.  carry bit set?
07E7  3002      JR      NC,#07EB        ; no, skip next step

07E9  3601      LD      (HL),#01        ; set pallete bank selector to 1

07EB  23        INC     HL              ; HL := REG_PALETTE_B = 2nd pallete bank
07EC  3600      LD      (HL),#00        ; clear the pallete bank selector
07EE  CB07      RLC     A               ; rotate left again.  carry bit set ?
07F0  3002      JR      NC,#07F4        ; no, skip next step

07F2  3601      LD      (HL),#01        ; set pallete bank selector to 1

07F4  328B63    LD      (#638B),A       ; store A into ???

; draws DONKEY KONG logo to screen

07F7  21083D    LD      HL,#3D08        ; load HL with start of table data

07FA  3EB0      LD      A,#B0           ; A := #B0 = code for girder on screen
07FC  46        LD      B,(HL)          ; get first data.  this is used as a loop counter
07FD  23        INC     HL              ; next table entry
07FE  5E        LD      E,(HL)          ; load E with table data
07FF  23        INC     HL              ; next entry
0800  56        LD      D,(HL)          ; load D with table data.  DE now has an address
```

```
0801  12        LD      (DE),A          ; draw girder on screen
0802  13        INC     DE              ; next address
0803  10FC      DJNZ    #0801           ; Next B

0805  23        INC     HL              ; next table entry
0806  7E        LD      A,(HL)          ; get data
0807  FE00      CP      #00             ; done ?
0809  C2FA07    JP      NZ,#07FA        ; no, loop again

080C  111E03    LD      DE,#031E        ; load task data for text "(C) 1981"
080F  CD9F30    CALL    #309F           ; insert task to draw text
0812  13        INC     DE              ; load task data for text "NINTENDO OF AMERICA"
0813  CD9F30    CALL    #309F           ; insert task to draw text
0816  21CF39    LD      HL,#39CF        ; load HL with table data for kong beating chest
0819  CD4E00    CALL    #004E           ; update kong's sprites
081C  CD243F    CALL    #3F24           ; draw TM logo onscreen [patch? orig japanese had 3 NOPs here]
081C  C3233F    JP      #3F23           ; jump to additional code to display the romhack version
081F  00        NOP                     ; no operation
0820  210869    LD      HL,#6908        ; load HL with start of kong sprite X pos
0823  0E44      LD      C,#44           ; load C with offset to add X
0825  FF        RST     #38             ; draw kong in new position
0826  210B69    LD      HL,#690B        ; load HL with start of kong sprite Y pos
0829  0E78      LD      C,#78           ; load C with offset to add Y
0829  0E80      LD      C,#80           ; load C with offset - draw kong one position lower
082B  FF        RST     #38             ; draw kong
082C  C9        RET                     ; return

; jump here from #07D0
; loads C with #638B, decreases #638A

082D  3A8B63    LD      A,(#638B)       ; load A with ???
0830  4F        LD      C,A             ; copy to C
0831  3A8A63    LD      A,(#638A)       ; load A with kong intro flash counter
0834  3D        DEC     A               ; decrease
0835  328A63    LD      (#638A),A       ; store result
0838  C3DA07    JP      #07DA           ; jump back

; jump here from #07DC

083B  210960    LD      HL,WaitTimerMSB ; load HL with timer address
083E  3602      LD      (HL),#02        ; set timer to 2
0840  23        INC     HL              ; HL := GameMode2
0841  34        INC     (HL)            ; increase game mode2
0842  218A63    LD      HL,#638A        ; load HL with kong intro flash counter
0845  3600      LD      (HL),#00        ; clear counter
0847  23        INC     HL              ; HL := #638B = ???
0848  3600      LD      (HL),#00        ; clear this memory
084A  C9        RET                     ; return

; arrive from #0747 when GameMode2 == 7

084B  E7        RST     #20             ; update timer and continue here only when complete, else RET

084C  210A60    LD      HL,GameMode2        ; load HL with game mode2
084F  3600      LD      (HL),#00        ; set to 0
0851  C9        RET                     ; return

; called from #0986
; clears screen and all sprites

0852  210074    LD      HL,#7400        ; #7400 is beginning of video RAM
0855  0E04      LD      C,#04           ; for C= 1 to 4
0857  0600      LD      B,#00           ; for B = 1 to 256
0859  3E10      LD      A,#10           ; #10 is clear for screen in video RAM

085B  77        LD      (HL),A          ; clear this screen element
085C  23        INC     HL              ; next screen location
085D  10FC      DJNZ    #085B           ; Next B

085F  0D        DEC     C               ; Next C
0860  C25708    JP      NZ,#0857        ; loop until done

0863  210069    LD      HL,#6900        ; load HL with start of sprite RAM
0866  0E02      LD      C,#02           ; for C = 1 to 2
0868  06C0      LD      B,#C0           ; for B = 1 to #C0
086A  AF        XOR     A               ; A := 0

086B  77        LD      (HL),A          ; clear RAM
086C  23        INC     HL              ; next memory
086D  10FC      DJNZ    #086B           ; next B

086F  0D        DEC     C               ; next C
0870  C26808    JP      NZ,#0868        ; loop until done

0873  C9        RET                     ; return

; called from many places.  EG #08BA and #01C3 and #0C92 and other places
; clears the screen and sprites

0874  210474    LD      HL,#7404        ; load HL with start of video RAM
0877  0E20      LD      C,#20           ; For C = 1 to #20

0879  061C      LD      B,#1C           ; for B = 1 to #1C
087B  3E10      LD      A,#10           ; A := #10
087D  110400    LD      DE,#0004        ; DE = 4, used as offset to add later
```

```
0880  77        LD      (HL),A          ; store into memory
0881  23        INC     HL              ; next memory
0882  10FC      DJNZ    #0880           ; Next B

0884  19        ADD     HL,DE           ; add offset of 4
0885  0D        DEC     C               ; decrease counter
0886  C27908    JP      NZ,#0879        ; loop until zero

0889  212275    LD      HL,#7522        ; load HL with screen location
088C  112000    LD      DE,#0020        ; load DE with offset to use
088F  0E02      LD      C,#02           ; for C = 1 to 2
0891  3E10      LD      A,#10           ; A := #10 = clear screen byte

0893  060E      LD      B,#0E           ; for B = 1 to #0E
0895  77        LD      (HL),A          ; clear the screen element
0896  19        ADD     HL,DE           ; add offset for next
0897  10FC      DJNZ    #0895           ; Next B

0899  212375    LD      HL,#7523        ; load HL with next screen location
089C  0D        DEC     C               ; done ?
089D  C29308    JP      NZ,#0893        ; no, loop again

08A0  210069    LD      HL,#6900        ; load HL with start of sprite RAM
08A3  0600      LD      B,#00           ; For B = 0 to #FF
08A5  3E00      LD      A,#00           ; A := 0

08A7  77        LD      (HL),A          ; clear memory
08A8  23        INC     HL              ; next memory
08A9  10FC      DJNZ    #08A7           ; Next B

08AB  0680      LD      B,#80           ; For B = 0 to #80
08AD  77        LD      (HL),A          ; store memory
08AE  23        INC     HL              ; next memory
08AF  10FC      DJNZ    #08AD           ; Next B

08B1  C9        RET                     ; Return

; jump from #00C9
; arrive here when credits have been inserted, waiting for game to start

08B2  3A0A60    LD      A,(GameMode2)       ; load A with game mode2

; GameMode2 = 1 during attract mode, 7 during intro , A during how high can u get,
;         B right before play, C during play, D when dead, 10 when game over

08B5  EF        RST     #28             ; jump based on A

08B6  BA 08                             ; #08BA         ; display screen to press start etc.
08B8  F8 08                             ; #08F8         ; wait for start buttons to be pressed

08BA  CD7408    CALL    #0874           ; clear the screen and sprites
08BD  AF        XOR     A               ; A := 0
08BE  320760    LD      (NoCredits),A   ; store into credit indicator
08C1  110C03    LD      DE,#030C        ; load DE with task code to display "PUSH" onscreen
08C4  CD9F30    CALL    #309F           ; insert task
08C7  210A60    LD      HL,GameMode2        ; load A with game mode2
08CA  34        INC     (HL)            ; increase game mode2
08CB  CD6509    CALL    #0965           ; draw credits on screen if any are present and displays high score table
08CE  AF        XOR     A               ; A := 0
08CF  21867D    LD      HL,REG_PALETTE_A        ; load HL with pallete bank
08D2  77        LD      (HL),A          ; clear palette bank selector
08D3  2C        INC     L               ; next pallete bank
08D4  77        LD      (HL),A          ; clear palette bank selector

; called from #08F8

08D5  0604      LD      B,#04           ; B := 4 = 0100 binary
08D7  1E09      LD      E,#09           ; E := 9 , code for "ONLY 1 PLAYER BUTTON"
08D9  3A0160    LD      A,(NumCredits)  ; load A with number of credits
08DC  FE01      CP      #01             ; == 1 ?
~~08DE  CAE408    JP      Z,#08E4         ; yes, skip next 2 steps~~
08DE  C3E408    JP      #08E4           ; two player game removed, so always skip next 2 steps

08E1  060C      LD      B,#0C           ; B := #0C = 1100 binary
08E3  1C        INC     E               ; E := #0A, code for "1 OR 2 PLAYERS BUTTON"

08E4  3A1A60    LD      A,(FrameCounter)       ; load A with # Timer constantly counts down from FF to 00
08E7  E607      AND     #07             ; mask bits. zero ?
08E9  C2F308    JP      NZ,#08F3        ; no, skip next 3 steps

08EC  7B        LD      A,E             ; yes, load A with E for code of text to draw, for buttons to press to start
08ED  CDE905    CALL    #05E9           ; draw text to screen
08F0  CD1606    CALL    #0616           ; draw credits on screen

08F3  3A007D    LD      A,(IN2)         ; load A with IN2 [Credit/Service/Start Info]
08F6  A0        AND     B               ; mask bits with B
08F7  C9        RET                     ; return

; jump from #08B5 when GameMode2 == 1

08F8  CDD508    CALL    #08D5           ; draws press player buttons and loads A with IN2, masked by possible player numbers
08FB  FE04      CP      #04             ; is the player 1 button pressed ?
08FD  CA0609    JP      Z,#0906         ; yes, skip ahead
```

```
0900  FE08      CP      #08             ; is the player 2 button pressed ?
0902  CA1909    JP      Z,#0919         ; yes, skip ahead
0900  00        NOP                     ; no operation - two player game removed,so ignore P2 button
0901  00        NOP                     ; no operation - two player game removed,so ignore P2 button
0902  00        NOP                     ; no operation - two player game removed,so ignore P2 button
0903  00        NOP                     ; no operation - two player game removed,so ignore P2 button
0904  00        NOP                     ; no operation - two player game removed,so ignore P2 button

0905  C9        RET                     ; return to #00D2

; player 1 start

0906  CD7709    CALL    #0977           ; subtract 1 credit and update screen credit counter
0909  214860    LD      HL,P2NumLives       ; load HL with RAM used for player 2
090C  0608      LD      B,#08           ; for B = 1 to 8
090E  AF        XOR     A               ; A := 0

090F  77        LD      (HL),A          ; clear memory
0910  2C        INC     L               ; next memory
0911  10FC      DJNZ    #090F           ; Next B

0913  210000    LD      HL,#0000        ; clear HL
0916  C33809    JP      #0938           ; skip ahead

; 2 players start

0919  CD7709    CALL    #0977           ; subtract 1 credit and update screen credit counter
091C  CD7709    CALL    #0977           ; subtract 1 credit and update screen credit counter
091F  114860    LD      DE,P2NumLives       ; load DE with RAM location used for player 2
0922  3A2060    LD      A,(StartingLives)       ; load initial number of lives
0925  12        LD      (DE),A          ; store into number of lives player 2
0926  1C        INC     E               ; DE := Unk6049
0927  215E09    LD      HL,#095E        ; load HL with source data table start
092A  010700    LD      BC,#0007        ; counter = 7
092D  EDB0      LDIR                    ; copy #095E into Unk6049 for 7 bytes
092F  110101    LD      DE,#0101        ; load task #1, parameter 1.  clears player 1 and 2 scores and displays them.
0932  CD9F30    CALL    #309F           ; insert task
0935  210001    LD      HL,#0100        ; HL := #100

; code to draw the random ladders for the pies screen
; #0BC7 : draw girders screen, #0919 : draw pies screen, #135B : draw elevators screen, #1344 : draw rivets screen

0919  FE01      CP      #01             ; is this the girders screen?
091B  CAC70B    JP      Z,#0BC7         ; yes, jump to specific code to draw screen and ladders for girders screen
091E  FE03      CP      #03             ; is this the elevators screen?
0920  CA5B13    JP      Z,#135B         ; yes, jump to specific code to draw screen and ladders for elevators screen
0923  D5        PUSH    DE              ; save DE to the stack for later
0924  DD218B3B  LD      IX,#3B8B        ; load IX with pointer to ladder datastructure for pies screen
0928  21006B    LD      HL,#6B00        ; load HL with pointer to output datastructure to build ladder definitions
092B  CD210C    CALL    #0C21           ; create the random ladder definitions
092E  11006B    LD      DE,#6B00        ; load DE with pointer to output datastructure containing ladder definitions
0931  CDA70D    CALL    #0DA7           ; draw the random ladders
0934  C3F509    JP      #09F5           ; jump to continue
0937  00        NOP                     ; no operation

0938  220E60    LD      (PlayerTurnB),HL      ; store HL into PlayerTurnB and TwoPlayerGame.  TwoPlayerGame is the number of
players in the game
093B  CD7408    CALL    #0874           ; clear the screen and sprites
093E  114060    LD      DE,P1NumLives       ; load DE with address for number of lives player 1
0941  3A2060    LD      A,(StartingLives)       ; number of initial lives set with dip switches (3, 4, 5, or 6)
0944  12        LD      (DE),A          ; store into number of lives
0945  1C        INC     E               ; DE := Unk6041
0946  215E09    LD      HL,#095E        ; load HL with start of table data
0949  010700    LD      BC,#0007        ; counter = 7
094C  EDB0      LDIR                    ; copy #095E into Unk6041 for 7 bytes
094E  110001    LD      DE,#0100        ; load task #1, parameter 0.  clears player 1 score and displays it
0951  CD9F30    CALL    #309F           ; insert task
0954  AF        XOR     A               ; A := 0
0955  320A60    LD      (GameMode2),A       ; reset game mode2
0958  3E03      LD      A,#03           ; A := 3
095A  320560    LD      (GameMode1),A   ; store into game mode1
095D  C9        RET                     ; return

; table data use in code above - gets copied to Unk6041 to Unk6041+7

095E  01 65 3A 01 00 00 00              ; #3A65 is start of table data for screens/levels

; called from #08CB

0965  110004    LD      DE,#0400        ; set task #4 = draws credits on screen if any are present
0968  CD9F30    CALL    #309F           ; insert task
096B  111403    LD      DE,#0314        ; set task #3, parameter 14 through 1A.  For display of high score table
096E  0606      LD      B,#06           ; for B = 1 to 6

0970  CD9F30    CALL    #309F           ; insert task
0973  1C        INC     E               ; increase task parameter
0974  10FA      DJNZ    #0970           ; Next B

0976  C9        RET                     ; return

; subtract 1 credit and update screen credit counter

0977  210160    LD      HL,NumCredits   ; load HL with pointer to number of credits
097A  3E99      LD      A,#99           ; A := #99
097C  86        ADD     A,(HL)          ; add to number of credits.   equivalent of subtracting 1
```

```
097D  27        DAA                     ; decimal adjust
097E  77        LD      (HL),A          ; store into number of credits
097F  110004    LD      DE,#0400        ; set task #4 = draws credits on screen if any are present
0982  CD9F30    CALL    #309F           ; insert task
0985  C9        RET                     ; return

; arrive here when a game begins
; clears screen, clears sounds, sets screen flip if needed
; jump from #0701 when GameMode2 == 0

0986  CD5208    CALL    #0852           ; clear screen and all sprites
0989  CD1C01    CALL    #011C           ; clear all sounds
098C  11827D    LD      DE,REG_FLIPSCREEN       ; load DE with flip screen setting
098F  3E01      LD      A,#01           ; A := 1
0991  12        LD      (DE),A          ; store
0992  210A60    LD      HL,GameMode2        ; load HL with game mode 2 address
0995  3A0E60    LD      A,(PlayerTurnB)       ; load A with 0 when player 1 is up, = 1 when player 2 is up
0998  A7        AND     A               ; is player 1 up?
0999  C29F09    JP      NZ,#099F        ; no, skip next 2 steps

099C  3601      LD      (HL),#01        ; set game mode 2 to 1
099E  C9        RET                     ; return

099F  3A2660    LD      A,(UprightCab)       ; load A with upright/cocktail
09A2  3D        DEC     A               ; is this cocktail mode ?
09A3  CAA809    JP      Z,#09A8         ; no, skip next 2 steps

09A6  AF        XOR     A               ; A := 0
09A7  12        LD      (DE),A          ; set screen to flipped

09A8  3603      LD      (HL),#03        ; set game mode 2 to 3
09AA  C9        RET                     ; return

; jump from #0701 when GameMode2 == 1
; copy player data, set screen, set next game mode based on number of players

09AB  214060    LD      HL,P1NumLives        ; load HL with source data location
09AE  112862    LD      DE,#6228        ; load DE with destination data location.  start with remaining lives
09B1  010800    LD      BC,#0008        ; byte counter set to 8
09B4  EDB0      LDIR                    ; copy (HL) into (DE) from P1NumLives to P2NumLives into #6228 to #622F
09B6  2A2A62    LD      HL,(#622A)      ; EG #3A65.  start of table data for screens/levels
09B9  7E        LD      A,(HL)          ; load screen number from table
09BA  322762    LD      (#6227),A       ; store screen number
09BD  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
09C0  A7        AND     A               ; 1 player game?
09C1  210960    LD      HL,WaitTimerMSB ; load HL with timer address
09C4  110A60    LD      DE,GameMode2        ; load DE with game mode2 address
09C7  CAD009    JP      Z,#09D0         ; if 1 player game, skip ahead

; 2 player game

09CA  3678      LD      (HL),#78        ; store #78 into timer
09CC  EB        EX      DE,HL           ; DE <> HL.  HL now has game mode2
09CD  3602      LD      (HL),#02        ; GameMode2 := 2
09CF  C9        RET                     ; return

; 1 player game

09D0  3601      LD      (HL),#01        ; store 1 into timer
09D2  EB        EX      DE,HL           ; DE <> HL.  HL now has game mode2
09D3  3605      LD      (HL),#05        ; GameMode2 := 5
09D5  C9        RET                     ; return


; used to draw players during 2 player game
; jump here from #0701
; clears palettes, draws "PLAYER <1>", draws player2 score, draws "2UP"

09D6  AF        XOR     A               ; A := 0
09D7  32867D    LD      (REG_PALETTE_A),A       ; clear palette bank selector
09DA  32877D    LD      (REG_PALETTE_B),A       ; clear palette bank selector
09DD  110203    LD      DE,#0302        ; load task data for text #2 "PLAYER <1>"
09E0  CD9F30    CALL    #309F           ; insert task to draw
09E3  110102    LD      DE,#0201        ; load task #2, parameter 1 to display player 2 score
09E6  CD9F30    CALL    #309F           ; insert task
09E9  3E05      LD      A,#05           ; A := 5
09EB  320A60    LD      (GameMode2),A       ; store into game mode2

09EE  3E02      LD      A,#02           ; load A with "2"
09F0  32E074    LD      (#74E0),A       ; write to screen
09F3  3E25      LD      A,#25           ; load A with "U"
09F5  32C074    LD      (#74C0),A       ; write to screen
09F8  3E20      LD      A,#20           ; load A with "P"
09FA  32A074    LD      (#74A0),A       ; write to screen
09FD  C9        RET                     ; return

; arrive from #0701 when GameMode2 == 3

09FE  214860    LD      HL,P2NumLives        ; source location is ???
0A01  112862    LD      DE,#6228        ; destination is player lives remaining plus other player variables
0A04  010800    LD      BC,#0008        ; byte counter set to 8
0A07  EDB0      LDIR                    ; copy
0A09  2A2A62    LD      HL,(#622A)      ; load HL with table for screens/levels
0A0C  7E        LD      A,(HL)          ; load A with screen number from table
0A0D  322762    LD      (#6227),A       ; store A into screen number
```

```
0A10  3E78      LD      A,#78           ; A := #78
0A12  320960    LD      (WaitTimerMSB),A    ; store into timer
0A15  3E04      LD      A,#04           ; A := 4
0A17  320A60    LD      (GameMode2),A       ; store into game mode2
0A1A  C9        RET                     ; return

; arrive from #0701 when GameMode2 == 4
; clears palletes, draws "PLAYER <II>", update player2 score, draw "2UP" to screen

0A1B  AF        XOR     A               ; A := 0
0A1C  32867D    LD      (REG_PALETTE_A),A       ; clear palette bank selector
0A1F  32877D    LD      (REG_PALETTE_B),A       ; clear palette bank selector
0A22  110303    LD      DE,#0303        ; load task data for text #3 "PLAYER <II>"
0A25  CD9F30    CALL    #309F           ; insert task to draw text
0A28  110102    LD      DE,#0201        ; load task #2, parameter 1 to display player 2 score
0A2B  CD9F30    CALL    #309F           ; insert task
0A2E  CDEE09    CALL    #09EE           ; draw "2UP" to screen
0A31  3E05      LD      A,#05           ; A := 5
0A33  320A60    LD      (GameMode2),A       ; store into game mode2

09BD  210960    LD      HL,#6009        ; load HL with timer address WaitTimerMSB (original code from #09C1)
09C0  110A60    LD      DE,#600A        ; load DE with game mode2 address (original code from #09C4)
09C3  3601      LD      (HL),#01        ; store 1 into timer (original code from #09D0)
09C5  EB        EX      DE,HL           ; DE <> HL. HL now has game mode2 (original code from #09D2)
09C6  3605      LD      (HL),#05        ; game mode2 := 5 (original code from #09C6)
09C8  C9        RET                     ; return

; continued code to create random ladder definitions (continued from #0C75)

09C9  DD7E00    LD      A,(IX+#00)      ; load C with y-value top of ladder
09CC  DD23      INC     IX              ; increase pointer to input data
09CE  57        LD      D,A             ; store y-value top  of ladder to D
09CF  DD4E00    LD      C,(IX+#00)      ; load C with y-value bottom of ladder
09D2  DD23      INC     IX              ; increase pointer to input data
09D4  C5        PUSH    BC              ; store BC to the stack for later
09D5  3AA166    LD      A,(#66A1)       ; load the ladder counter
09D8  3C        INC     A               ; increase the ladder counter
09D9  32A166    LD      (#66A1),A       ; store the ladder counter
09DC  47        LD      B,A             ; store ladder counter to B
09DD  3AA066    LD      A,($66A0)       ; load broken ladder number
09E0  90        SUB     B               ; is ladder counter equal to broken ladder number?
09E1  3E00      LD      A,#00           ; load A with normal ladder type
09E3  2002      JR      NZ,#09E7        ; no, skip next step, don't make it a broken ladder
09E5  3E01      LD      A,#01           ; load A with broken ladder type
09E7  77        LD      (HL),A          ; store ladder type in data structure
09E8  23        INC     HL              ; next element
09E9  C1        POP     BC              ; restore BC from the stack
09EA  70        LD      (HL),B          ; store x-value top of ladder in data structure
09EB  23        INC     HL              ; next element
09EC  72        LD      (HL),D          ; store y-value top of ladder in data structure
09ED  23        INC     HL              ; next element
09EE  70        LD      (HL),B          ; store x-value bottom of ladder in data structure
09EF  23        INC     HL              ; next element
09F0  71        LD      (HL),C          ; store y-value bottom of ladder in data structure
09F1  23        INC     HL              ; next element
09F2  C3300C    JP      #0C30           ; loop again

; continued code to draw the random ladders for the pies screen (continued from #0934)

09F5  D1        POP     DE              ; restore DE from the stack
09F6  CDA70D    CALL    #0DA7           ; draw the screen
09F9  C9        RET                     ; return

; code to activate the ladders

09FA  CD4124    CALL    #2441           ; call code to activate the random ladders
09FD  3A2762    LD      A,(#6227)       ; load A with screen number
0A00  FE02      CP      #02             ; is this the pies screen?
0A02  C2650D    JP      NZ,#0D65        ; no, jump back
0A05  21AF3B    LD      HL,#3BAF        ; load HL with data table static elements
0A08  DD210963  LD      IX,#6309        ; load IX with index ladder activation
0A0C  CD7524    CALL    #2475           ; call code to activate the static ladders
0A0F  C3650D    JP      #0D65           ; jump back

0A12  00000000000000000000000000000000  ; no operations - free space
0A22  000000                            ; no operations - free space

; code to get a random value between 0 and 6

0A25  CD7B0A    CALL    #0A7B           ; load A with random value (copy in B)
0A28  E601      AND     #01             ; keep bit 0 - vaue between 0 and 1
0A2A  4F        LD      C,A             ; save to C
0A2B  78        LD      A,B             ; restore random value from B
0A2C  1F        RRA                     ; rotate right
0A2D  E601      AND     #01             ; keep bit 0 - value between 0 and 1
0A2F  81        ADD     A,C             ; add C
0A30  4F        LD      C,A             ; save to C
0A31  78        LD      A,B             ; restore random value from B
0A32  1F        RRA                     ; rotate right
0A33  E603      AND     #03             ; keep bits 0 and 1 - value between 0 and 3
0A35  81        ADD     A,C             ; add C, A is now value between 0 and 5
0A36  C9        RET                     ; return

; arrive from #0701 when GameMode2 == 5
; updates high score, player score, remaining lives, level, 1UP
```

```
0A37  110403    LD      DE,#0304        ; load task data for text #4 "HIGH SCORE"
0A3A  CD9F30    CALL    #309F           ; insert task to draw text
0A3D  110202    LD      DE,#0202        ; load task #2, parameter 2 to display high score
0A40  CD9F30    CALL    #309F           ; insert task
0A43  110002    LD      DE,#0200        ; load task #2, parameter 0 to display player 1 score
0A46  CD9F30    CALL    #309F           ; insert task
0A49  110006    LD      DE,#0600        ; load task #6 parameter 0 to display lives remaining and level
0A4C  CD9F30    CALL    #309F           ; insert task
0A4F  210A60    LD      HL,GameMode2        ; load HL with game mode2 address
0A52  34        INC     (HL)            ; increase game mode

;  called from #01F1 , #0798, and other places
; draw "1UP" on screen

0A53  3E01      LD      A,#01           ; load A with "1"
0A55  324077    LD      (#7740),A       ; write to screen
0A58  3E25      LD      A,#25           ; load A with "U"
0A5A  322077    LD      (#7720),A       ; write to screen
0A5D  3E20      LD      A,#20           ; load A with "P"
0A5F  320077    LD      (#7700),A       ; write to screen
0A62  C9        RET                     ; return

; arrive from #0701 when GameMode2 == 6
; clears screen and sprites, check for intro screen to run

0A63  DF        RST     #18             ; count down WaitTimerMSB and only continue here if == 0, else return to higher sub.
0A64  CD7408    CALL    #0874           ; clears the screen and sprites
0A67  210960    LD      HL,WaitTimerMSB ; load HL with timer
0A67  CD713F    CALL    #3F71           ; initialize a cycling pointer into the code (#0100-#3FFF)
0A6A  3601      LD      (HL),#01        ; set timer to 1
0A6C  2C        INC     L               ; HL := GameMode2
0A6D  34        INC     (HL)            ; increase game mode2 to 7
0A6E  112C62    LD      DE,#622C        ; load DE with game start flag address
0A71  1A        LD      A,(DE)          ; load A with game start flag
0A72  A7        AND     A               ; is this game just beginning?
0A73  C0        RET     NZ              ; yes, return

0A74  34        INC     (HL)            ; else increase game mode2 to 8 - skip kong intro to begin
0A75  C9        RET                     ; return

; arrive from #0701 when GameMode2 == 7

0A76  3A8563    LD      A,(#6385)       ; varies from 0 to 7 while the intro screen runs, when kong climbs the dual ladders
and scary music is played
0A79  EF        RST     #28             ; jump based on A

0A7A  8A 0A                     0       ; #0A8A
0A7C  BF 0A                     1       ; #0ABF
0A7E  E8 0A                     2       ; #0AE8
0A80  69 30                     3       ; #3069
0A82  06 0B                     4       ; #0B06
0A84  69 30                     5       ; #3069
0A86  68 0B                     6       ; #0B68
0A88  B3 0B                     7       ; #0BB3

; arrive from #0A79 when intro screen indicator == 0

0A8A  AF        XOR     A               ; A := 0
0A8B  32867D    LD      (REG_PALETTE_A),A       ; clear palette bank selector
0A8E  3C        INC     A               ; A := 1
0A8F  32877D    LD      (REG_PALETTE_B),A       ; store into palette bank selector
0A92  110D38    LD      DE,#380D        ; load DE with start of table data
0A95  CDA70D    CALL    #0DA7           ; draw the screen
0A98  3E10      LD      A,#10           ; A := #10
0A9A  32A376    LD      (#76A3),A       ; erase a graphic near top of screen
0A9D  326376    LD      (#7663),A       ; erase a graphic near top of screen
0AA0  3ED4      LD      A,#D4           ; A := #D4
0AA2  32AA75    LD      (#75AA),A       ; draw a ladder at top of screen
0AA5  AF        XOR     A               ; A := 0
0AA6  32AF62    LD      (#62AF),A       ; store into kong climbing counter
0AA9  21B438    LD      HL,#38B4        ; load HL with start of table data
0AAC  22C263    LD      (#63C2),HL      ; store
0AAF  21CB38    LD      HL,#38CB        ; load HL with start of table data
0AB2  22C463    LD      (#63C4),HL      ; store
0AB5  3E40      LD      A,#40           ; A := #40
0AB7  320960    LD      (WaitTimerMSB),A    ; set timer to #40
0ABA  218563    LD      HL,#6385        ; load HL with intro screen counter
0ABD  34        INC     (HL)            ; increase
0ABE  C9        RET                     ; return

; arrive from #0A79 when intro screen indicator == 1

0ABF  DF        RST     #18             ; count down timer and only continue here if zero, else RET
0AC0  218C38    LD      HL,#388C        ; load HL with start of table data for kong
0AC3  CD4E00    CALL    #004E           ; update kong's sprites
0AC6  210869    LD      HL,#6908        ; load HL with start of Kong sprite
0AC9  0E30      LD      C,#30           ; load offset to add
0ACB  FF        RST     #38             ; move kong
0ACC  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
0ACF  0E99      LD      C,#99           ; load offset to add
0AD1  FF        RST     #38             ; move kong
0AD2  3E1F      LD      A,#1F           ; A := #1F
0AD4  328E63    LD      (#638E),A       ; store into kong ladder climb counter
0AD7  AF        XOR     A               ; A := 0
```

```
0AD8  320C69    LD      (#690C),A       ; store into kong's right arm sprite
0ADB  218A60    LD      HL,#608A        ; load HL with music buffer
0ADE  3601      LD      (HL),#01        ; play scary music for start of game sound
0AE0  23        INC     HL              ; load HL with duration
0AE1  3603      LD      (HL),#03        ; set duration to 3
0AE3  218563    LD      HL,#6385        ; load HL with intro screen counter
0AE6  34        INC     (HL)            ; increase
0AE7  C9        RET                     ; return

; arrive from #0A79 when intro screen indicator == 2

0AE8  CD6F30    CALL    #306F           ; animate kong climbing up the ladder with girl under arm
0AEB  3AAF62    LD      A,(#62AF)       ; load A with kong climbing counter
0AEE  E60F      AND     #0F             ; mask bits, now between 0 and #F.  zero?
0AF0  CC4A30    CALL    Z,#304A         ; yes, roll up kong's ladder behind him

0AF3  3A0B69    LD      A,(#690B)       ; load HL with start of Kong sprite
0AF6  FE5D      CP      #5D             ; < #5D ?
0AF8  D0        RET     NC              ; no, return

0AF9  3E20      LD      A,#20           ; A := #20
0AFB  320960    LD      (WaitTimerMSB),A    ; set timer to #20
0AFE  218563    LD      HL,#6385        ; load HL with intro screen counter
0B01  34        INC     (HL)            ; increase
0B02  22C063    LD      (#63C0),HL      ; store HL into ???
0B05  C9        RET                     ; return

; arrive from #0A79 when intro screen indicator == 4

0B06  3A1A60    LD      A,(FrameCounter)        ; load A with this clock counts down from #FF to 00 over and over...
0B09  0F        RRCA                    ; rotate right.  carry bit?
0B0A  D8        RET     C               ; yes, return

0B0B  2AC263    LD      HL,(#63C2)      ; load HL with ??? EG HL = #38B4
0B0E  7E        LD      A,(HL)          ; load table data
0B0F  FE7F      CP      #7F             ; end of data ?
0B11  CA1E0B    JP      Z,#0B1E         ; yes, jump ahead

0B14  23        INC     HL              ; next HL
0B15  22C263    LD      (#63C2),HL      ; store
0B18  4F        LD      C,A             ; C := A
0B19  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
0B1C  FF        RST     #38             ; move kong
0B1D  C9        RET                     ; return

0B1E  215C38    LD      HL,#385C        ; load HL with start of kong graphic table data
0B21  CD4E00    CALL    #004E           ; update kong's sprites
0B24  110069    LD      DE,#6900        ; load destination with girl sprite
0B27  010800    LD      BC,#0008        ; set counter to 8
0B2A  EDB0      LDIR                    ; draw the girl after kong takes her up the ladder
0B2C  210869    LD      HL,#6908        ; load HL with kong sprite start address
0B2F  0E50      LD      C,#50           ; C := #50
0B31  FF        RST     #38             ; move kong
0B32  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
0B35  0EFC      LD      C,#FC           ; C := #FC
0B37  FF        RST     #38             ; move kong

0B38  CD4A30    CALL    #304A           ; roll up kong's ladder behind him
0B3B  3A8E63    LD      A,(#638E)       ; load A with kong ladder climb counter
0B3E  FE0A      CP      #0A             ; == #A ? (all done)
0B40  C2380B    JP      NZ,#0B38        ; no, loop again

0B43  3E03      LD      A,#03           ; set boom sound duration
0B45  328260    LD      (#6082),A       ; play boom sound
0B48  112C39    LD      DE,#392C        ; load DE with table data start for first angled girder
0B4B  CDA70D    CALL    #0DA7           ; draw the angled girder
0B4E  3E10      LD      A,#10           ; A := #10 = clear character
0B50  32AA74    LD      (#74AA),A       ; clear the right end of the top girder
0B53  328A74    LD      (#748A),A       ; clear the right end of the top girder
0B56  3E05      LD      A,#05           ; A := 5
0B58  328D63    LD      (#638D),A       ; store into kong bounce counter
0B5B  3E20      LD      A,#20           ; A := #20
0B5D  320960    LD      (WaitTimerMSB),A    ; set timer to #20
0B60  218563    LD      HL,#6385        ; load HL with intro screen counter
0B63  34        INC     (HL)            ; increase
0B64  22C063    LD      (#63C0),HL      ; store into ???
0B67  C9        RET                     ; return

; arrive from #0A79 when intro screen indicator == 6

0B68  3A1A60    LD      A,(FrameCounter)        ; load A with this clock counts down from #FF to 00 over and over...
0B6B  0F        RRCA                    ; rotate right.  carry bit set?
0B6C  D8        RET     C               ; yes, return

; make kong jump to the left during intro

0B6D  2AC463    LD      HL,(#63C4)      ; load HL with ??? (table data?)
0B70  7E        LD      A,(HL)          ; get table data
0B71  FE7F      CP      #7F             ; done ?
0B73  CA860B    JP      Z,#0B86         ; yes, jump ahead

0B76  23        INC     HL              ; next table entry
0B77  22C463    LD      (#63C4),HL      ; store for next
0B7A  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
0B7D  4F        LD      C,A             ; C := A
```

```
0B7E  FF        RST     #38             ; move kong
0B7F  210869    LD      HL,#6908        ; load HL with start of Kong sprite
0B82  0EFF      LD      C,#FF           ; C := #FF (negative 1)
0B84  FF        RST     #38             ; move kong
0B85  C9        RET                     ; return

0B86  21CB38    LD      HL,#38CB        ; load HL with start of table data
0B89  22C463    LD      (#63C4),HL      ; store into ???
0B8C  3E03      LD      A,#03           ; set boom sound duration
0B8E  328260    LD      (#6082),A       ; play boom sound
0B91  21DC38    LD      HL,#38DC        ; load HL with start of table data
0B94  3A8D63    LD      A,(#638D)       ; load A with kong bounce counter
0B97  3D        DEC     A               ; decrease
0B98  07        RLCA
0B99  07        RLCA
0B9A  07        RLCA
0B9B  07        RLCA                    ; rotate left 4 times (mult by 16)
0B9C  5F        LD      E,A             ; copy to E
0B9D  1600      LD      D,#00           ; D := 0
0B9F  19        ADD     HL,DE           ; add to HL
0BA0  EB        EX      DE,HL           ; DE <> HL
0BA1  CDA70D    CALL    #0DA7           ; draw the screen
0BA4  218D63    LD      HL,#638D        ; load HL with kong bounce counter
0BA7  35        DEC     (HL)            ; decrease.  done bouncing?
0BA8  C0        RET     NZ              ; no, return

0BA9  3EB0      LD      A,#B0           ; else A := #B0
0BAB  320960    LD      (WaitTimerMSB),A    ; store into counter
0BAE  218563    LD      HL,#6385        ; load HL with intro screen counter
0BB1  34        INC     (HL)            ; increase
0BB2  C9        RET                     ; return

; arrive from #0A79   last part of the intro to the game ?

0BB3  218A60    LD      HL,#608A        ; load HL with music sound address
0BB6  3A0960    LD      A,(WaitTimerMSB)    ; load A with timer value
0BB9  FE90      CP      #90             ; == #90 ?
0BBB  200B      JR      NZ,#0BC8        ; no, skip ahead

0BBD  360F      LD      (HL),#0F        ; play sound #0F = X X X kong sound
0BBF  23        INC     HL              ; HL := GameMode2
0BC0  3603      LD      (HL),#03        ; set game mode2 to 3
0BC2  211969    LD      HL,#6919        ; load HL with kong's face sprite
0BC5  34        INC     (HL)            ; increase - kong is now showing teeth
0BC6  1809      JR      #0BD1           ; skip ahead

0BC8  FE18      CP      #18             ; timer == #18 ?
0BCA  2005      JR      NZ,#0BD1        ; no, skip ahead

0BCC  211969    LD      HL,#6919        ; load HL with kong's face sprite
0BCF  35        DEC     (HL)            ; decrease - kong is normal face
0BD0  00        NOP                     ; no operation [?]

0BD1  DF        RST     #18             ; count down timer and only continue here if zero, else RET.  HL is loaded with
WaitTimerMSB address
0BD2  AF        XOR     A               ; A := 0
0BD3  328563    LD      (#6385),A       ; reset intro screen counter to zero
0BD6  34        INC     (HL)            ; increase timer in WaitTimerMSB
0BD7  23        INC     HL              ; HL := GameMode2
0BD8  34        INC     (HL)            ; increase game mode2 (to 8?)
0BD9  C9        RET                     ; return


0A76  210A60    LD      HL,#600A        ; load HL with the address of the game mode2
0A79  34        INC     (HL)            ; increase the game mode2, skip the ladder climbing sequence
0A7A  C9        RET                     ; return

; code to get a random value based on the opcode read at a cycling pointer into the code (#0100-#3FFF)

0A7B  3A3262    LD      A,(#6232)       ; get stored MSB of pointer into the code
0A7E  67        LD      H,A             ; store in H
0A7F  3A3362    LD      A,(#6233)       ; get stored LSB of pointer into the code
0A82  3C        INC     A               ; increase the LSB, advance to next byte in the code
0A83  6F        LD      L,A             ; store in L
0A84  2008      JR      NZ,#0A8E        ; if L is not equal to #00, no rollover LSB, skip next steps
0A86  24        INC     H               ; rollover of LSB, so increase the MSB
0A87  7C        LD      A,H             ; load A with the MSB
0A88  FE40      CP      #40             ; is the MSB equal to #40?
0A8A  2002      JR      NZ,#0A8E        ; no, no rollover MSB, skip the next step
0A8C  2601      LD      H,#01           ; reset H
0A8E  7E        LD      A,(HL)          ; load A with opcode from HL
0A8F  47        LD      B,A             ; store copy of opcode in B
0A90  7C        LD      A,H             ; load A with H
0A91  323262    LD      (#6232),A       ; save updated MSB of pointer into the code
0A94  7D        LD      A,L             ; load A with L
0A95  323362    LD      (#6233),A       ; save updated LSB of pointer into the code
0A98  78        LD      A,B             ; load A with stored opcode from B
0A99  C9        RET                     ; return

0A9A  00000000000000000000000000000000      ; no operations - free space

; code to get three different random values

0AA8  DD21A466  LD      IX,#66A4        ; set IX to memory location to store first random value
0AAC  CD250A    CALL    #0A25           ; get random value
```

```
0AAF  DD7700    LD      (IX+#00),A      ; store first random value
0AB2  CD250A    CALL    #0A25           ; get random value
0AB5  DD7701    LD      (IX+#01),A      ; store second random value
0AB8  CD250A    CALL    #0A25           ; get random value
0ABB  DD7702    LD      (IX+#02),A      ; store third random value
0ABE  DD7E00    LD      A,(IX+#00)      ; load A with first random value
0AC1  DD9601    SUB     (IX+#01)        ; is it equal to the second random value?
0AC4  28E2      JR      Z,#0AA8         ; yes, get three new random values
0AC6  DD7E00    LD      A,(IX+#00)      ; load A with first random value
0AC9  DD9602    SUB     (IX+#02)        ; is it equal to the third random value?
0ACC  28DA      JR      Z,#0AA8         ; yes, get three new random values
0ACE  DD7E01    LD      A,(IX+#01)      ; load A with second random value
0AD1  DD9602    SUB     (IX+#02)        ; is it equal to the third random value?
0AD4  28D2      JR      Z,#0AA8         ; yes, get three new random values
0AD6  C9        RET                     ; return

0AD7  6E 9A B0 DC                       ; data table with ladder y-positions offset for each area for the girders screen

; code to create random ladder definitions for the girders screen

0ADB  FD21D70A  LD      IY,#0AD7        ; load IY with start of data table with y-positions offsets
0ADF  3A1860    LD      A,(#6018)       ; load A with a random value based on RngTimer1
0AE2  E603      AND     #03             ; A is a random number between 0 and 3
0AE4  FE03      CP      #03             ; A == #03?
0AE6  2002      JR      NZ,#0AEA        ; no, skip next step
0AE8  3E01      LD      A,#01           ; A = #01
0AEA  C601      ADD     A,#01           ; A is a random number between 1 and 3
0AEC  32A266    LD      (#66A2),A       ; store in broken ladder skip
0AEF  21006B    LD      HL,#6B00        ; load HL with pointer to output data structure to build ladder definitions
0AF2  3E03      LD      A,#03           ; load A with #03
0AF4  32A066    LD      (#66A0),A       ; store in outer loop counter, repeat outer loop three times: #03, #02, #01
0AF7  E5        PUSH    HL              ; save HL to the stack for later
0AF8  DDE5      PUSH    IX              ; save IX to the stack for later
0AFA  CDA80A    CALL    #0AA8           ; calculate 3 different random values and store in #66A4, #66A5 and #66A6
0AFD  DDE1      POP     IX              ; restore IX from stack
0AFF  E1        POP     HL              ; restore HL from stack
0B00  3E03      LD      A,#03           ; load A with #03
0B02  32A166    LD      (#66A1),A       ; store in inner loop counter, repeat inner loop three times: #03, #02, #01
0B05  47        LD      B,A             ; store inner loop counter in B
0B06  DD21A466  LD      IX,#66A4        ; load IX with first ladder offset (from #66A4)(
0B0A  FE01      CP      #01             ; inner loop counter == #01?
0B0C  2808      JR      Z,#0B16         ; yes, jump forward
0B0E  DD23      INC     IX              ; load IX with second ladder offset (from #66A5)
0B10  FE02      CP      #02             ; inner loop counter == #02?
0B12  2802      JR      Z,#0B16         ; yes, jump forward
0B14  DD23      INC     IX              ; load IX with third ladder offset (from #66A6)
0B16  DD7E00    LD      A,(IX+#00)      ; load A with correct ladder offset
0B19  57        LD      D,A             ; store correct ladder offset in D
0B1A  78        LD      A,B             ; restore inner loop counter from B
0B1B  FE01      CP      #01             ; inner loop counter == #01 (last ladder)?
0B1D  3AA066    LD      A,(#66A0)       ; load A with outer loop counter
0B20  47        LD      B,A             ; store outer loop counter in B
0B21  200B      JR      NZ,#0B2E        ; no, not the last ladder, jump forward
0B23  3AA266    LD      A,(#66A2)       ; load A with broken ladder skip
0B26  90        SUB     B               ; outer loop counter == broken ladder skip?
0B27  CA9C0B    JP      Z,#0B9C         ; yes, skip this broken ladder, jump forward
0B2A  3E01      LD      A,#01           ; load A with #01 - ladder type is broken ladder
0B2C  1802      JR      #0B30           ; skip the next step
0B2E  3E00      LD      A,#00           ; load A with #00 - ladder type is normal ladder
0B30  77        LD      (HL),A          ; store ladder type in output data structure
0B31  23        INC     HL              ; set HL to next element in output data structure
0B32  7A        LD      A,D             ; restore ladder offset from D
0B33  CB27      SLA     A               ; shift left 1 time
0B35  CB27      SLA     A               ; shift left 2 times
0B37  CB27      SLA     A               ; shift left 3 times
0B39  CB27      SLA     A               ; shift left 4 times
0B3B  CB27      SLA     A               ; shift left 5 times - multiply ladder offset by #20
0B3D  4F        LD      C,A             ; store multiplied ladder offset in C
0B3E  3ED3      LD      A,#D3           ; load A with start x-value
0B40  91        SUB     C               ; subtract the multiplied ladder offset
0B41  32A766    LD      (#66A7),A       ; save ladder x-value to memory (determines segment on girder)
0B44  7A        LD      A,D             ; restore ladder offset from D
0B45  CB27      SLA     A               ; shift left - multiply by #02
0B47  32A866    LD      (#66A8),A       ; save ladder delta-value to memory (determines offset on slanted girder)
0B4A  FD7E00    LD      A,(IY+#00)      ; load A with start y-value
0B4D  4F        LD      C,A             ; store start y-value in C
0B4E  FD7E01    LD      A,(IY+#01)      ; load A with next start y-value
0B51  91        SUB     C               ; subtract C, A is now the difference between the two y-values
0B52  E603      AND     #03             ; is A a multiple of #04?
0B54  280F      JR      Z,#0B65         ; yes, jump forward
0B56  3AA766    LD      A,(#66A7)       ; load ladder x-value from memory
0B59  D610      SUB     #10             ; subtract offset #10 - move one girder segment to the left
0B5B  32A766    LD      (#66A7),A       ; save ladder x-value to memory
0B5E  3AA866    LD      A,(#66A8)       ; load ladder delta-value from memory
0B61  3C        INC     A               ; add offset #01 - move one girder segment to the left
0B62  32A866    LD      (#66A8),A       ; save ladder delta-value to memory
0B65  3AA766    LD      A,(#66A7)       ; load ladder x-value from memory
0B68  57        LD      D,A             ; store ladder x-value to D
0B69  3AA866    LD      A,(#66A8)       ; load ladder delta-value from memory
0B6C  4F        LD      C,A             ; store ladder delta-value to C
0B6D  78        LD      A,B             ; restore outer loop counter from B
0B6E  FE02      CP      #02             ; is outer loop counter #02?
0B70  2018      JR      NZ,#0B8A        ; no, jump forward to create bottom row or top row random ladder
0B72  7A        LD      A,D             ; yes, create middle row random ladder, restore ladder x-value from D
0B73  C608      ADD     A,#08           ; add offset to right on girder segment
```

```
0B75  57        LD      D,A             ; store adjusted ladder x-value to D
0B76  77        LD      (HL),A          ; store x-position top of ladder in data structure
0B77  23        INC     HL              ; next element
0B78  FD7E00    LD      A,(IY+#00)      ; load A with start y-value
0B7B  91        SUB     C               ; subtract delta-value stored in C to adjust for slanted girder
0B7C  77        LD      (HL),A          ; store y-position top of ladder in data structure
0B7D  23        INC     HL              ; next element
0B7E  7A        LD      A,D             ; restore ladder x-value from D
0B7F  77        LD      (HL),A          ; store x-position bottom of ladder in data structure
0B80  23        INC     HL              ; next element
0B81  FD7E01    LD      A,(IY+#01)      ; load A with start y-value
0B84  81        ADD     A,C             ; add ladder delta-value stored in C to adjust for slanted girder
0B85  77        LD      (HL),A          ; store y-position bottom of ladder in data structure
0B86  23        INC     HL              ; next element
0B87  C39C0B    JP      #0B9C           ; ready creating middle random ladder, jump forward
0B8A  7A        LD      A,D             ; create bottom row or top row random ladder, restore ladder x-value from D
0B8B  77        LD      (HL),A          ; store x-position top of ladder in data structure
0B8C  23        INC     HL              ; next element
0B8D  FD7E00    LD      A,(IY+#00)      ; load A with start y-value
0B90  81        ADD     A,C             ; add delta-value stored in C to adjust for slanted girder
0B91  77        LD      (HL),A          ; store y-position top of ladder in data structure
0B92  23        INC     HL              ; next element
0B93  7A        LD      A,D             ; restore ladder x-value from D
0B94  77        LD      (HL),A          ; store x-position bottom of ladder in data structure
0B95  23        INC     HL              ; next element
0B96  FD7E01    LD      A,(IY+#01)      ; load A with start y-value
0B99  91        SUB     C               ; subtract delta-value stored in C to adjust for slanted girder
0B9A  77        LD      (HL),A          ; store y-position bottom of ladder in data structure
0B9B  23        INC     HL              ; next element
0B9C  3AA166    LD      A,(#66A1)       ; load A with inner loop counter
0B9F  D601      SUB     #01             ; decrease inner loop counter
0BA1  32A166    LD      (#66A1),A       ; store in inner loop counter
0BA4  C2050B    JP      NZ,#0B05        ; repeat inner loop if not zero
0BA7  FD23      INC     IY              ; increae pointer into data table with y-positions offsets
0BA9  3AA066    LD      A,(#66A0)       ; load A with outer loop counter
0BAC  D601      SUB     #01             ; decrease outer loop counter
0BAE  32A066    LD      (#66A0),A       ; store in outer loop counter
0BB1  C2F70A    JP      NZ,#0AF7        ; repeat outer loop if not zero
0BB4  3EAA      LD      A,#AA           ; load A with #AA
0BB6  77        LD      (HL),A          ; store end marker in data structure
0BB7  C9        RET                     ; return

; code to activate all ladders for the girders screen

0BB8  CD7124    CALL    #2471           ; activate static ladders
0BBB  21006B    LD      HL,#6B00        ; load HL with data structure with build ladder definition
0BBE  C37524    JP      #2475           ; activate random ladders

0BC1  000000000000                      ; no operations - free space

; code to draw the screen and random ladders for the girders screen
; #0BC7 : draw girders screen, #0919 : draw pies screen, #135B : draw elevators screen, #1344 : draw rivets screen

0BC7  CDA70D    CALL    #0DA7           ; draw the screen
0BCA  3A2762    LD      A,(#6227)       ; load A with the screen number
0BCD  FE01      CP      #01             ; is this the girders?
0BCF  C0        RET     NZ              ; no, return
0BD0  CDDB0A    CALL    #0ADB           ; create the random ladder definitions
0BD3  11006B    LD      DE,#6B00        ; load DE with pointer to output datastructure containing ladder definitions
0BD6  CDA70D    CALL    #0DA7           ; draw the random ladders
0BD9  C9        RET                     ; return

; called after kong jump on the girders at start of game ?
; also after mario dies
; how high can you get ?
; draws goofy kongs and 25m, 50m, etc.
; plays music

0BDA  CD1C01    CALL    #011C           ; clear all sounds
0BDD  DF        RST     #18             ; count down timer and only continue here if zero, else RET

0BDE  CD7408    CALL    #0874           ; clear the screen and sprites
0BE1  1606      LD      D,#06           ; load task #6
0BE3  3A0062    LD      A,(#6200)       ; load A with 1 when mario is alive, 0 when dead
0BE6  5F        LD      E,A             ; store into task parameter
0BE7  CD9F30    CALL    #309F           ; insert task to display remaining lives and level number
0BEA  21867D    LD      HL,REG_PALETTE_A        ; load HL with palette bank
0BED  3601      LD      (HL),#01        ; set palette bank selector
0BEF  23        INC     HL              ; next pallete bank
0BF0  3600      LD      (HL),#00        ; clear palette bank selector
0BF2  218A60    LD      HL,#608A        ; load HL with tune address
0BF5  3602      LD      (HL),#02        ; play how high can you get sound?
0BF7  23        INC     HL              ; HL := #608B .  load HL with music timer ?
0BF8  3603      LD      (HL),#03        ; set to 3 units
0BFA  21A763    LD      HL,#63A7        ; load HL with address of counter
0BFD  3600      LD      (HL),#00        ; clear the counter
0BFF  21DC76    LD      HL,#76DC        ; load HL with screen address to draw the number of meters ?
0C02  22A863    LD      (#63A8),HL      ; store - used at #0C54
0C05  3A2E62    LD      A,(#622E)       ; load A with number of goofy kongs to draw
0C08  FE06      CP      #06             ; < 6 ?
0C0A  3805      JR      C,#0C11         ; yes, skip next 2 steps [BUG.  change to 0C0A  1805   JR #0C11 to fix]

0C0C  3E05      LD      A,#05           ; else A := 5
0C0E  322E62    LD      (#622E),A       ; store into number of goofy kongs to draw
```

```
0C11  3A2F62    LD      A,(#622F)       ; load A with current screen/level
0C14  47        LD      B,A             ; copy to B
0C15  3A2A62    LD      A,(#622A)       ; load A with the low byte of the pointer for lookup to screens/levels
0C18  B8        CP      B               ; are they the same ?
0C19  2804      JR      Z,#0C1F         ; yes, skip next 2 steps

0C1B  212E62    LD      HL,#622E        ; else load HL with number of goofys to draw
0C1E  34        INC     (HL)            ; increase

0C1F  322F62    LD      (#622F),A       ; store A into current screen/level
0C22  3A2E62    LD      A,(#622E)       ; load A with number of goofys to draw
0C25  47        LD      B,A             ; copy to B for use as loop counter, refer to #0C7E
0C26  21BC75    LD      HL,#75BC        ; load HL with screen location start for goofy kong

0C29  0E50      LD      C,#50           ; C := #50 = start graphic for goofy kong

0C2B  71        LD      (HL),C          ; draw part of goofy kong
0C2C  0C        INC     C               ; next graphic
0C2D  2B        DEC     HL              ; next screen location
0C2E  71        LD      (HL),C          ; draw part of goofy kong
0C2F  0C        INC     C               ; next graphic
0C30  2B        DEC     HL              ; next screen location
0C31  71        LD      (HL),C          ; draw part of goofy kong
0C32  0C        INC     C               ; next graphic
0C33  2B        DEC     HL              ; next screen location
0C34  71        LD      (HL),C          ; draw part of goofy kong
0C35  79        LD      A,C             ; load A with graphic number
0C36  FE67      CP      #67             ; -- #67 ? (are we done?)
0C38  CA430C    JP      Z,#0C43         ; yes, skip next 4 steps

0C3B  0C        INC     C               ; next C
0C3C  112300    LD      DE,#0023        ; load DE with offset
0C3F  19        ADD     HL,DE           ; add to screen location
0C40  C32B0C    JP      #0C2B           ; loop again

0C43  3AA763    LD      A,(#63A7)       ; load A with counter
0C46  3C        INC     A               ; increase
0C47  32A763    LD      (#63A7),A       ; store
0C4A  3D        DEC     A               ; decrease
0C4B  CB27      SLA     A
0C4D  CB27      SLA     A               ; shift left twice, it is now a usable offset
0C4F  E5        PUSH    HL              ; save HL
0C50  21F03C    LD      HL,#3CF0        ; load HL with start of table data for 25m, 50m, etc.
0C53  C5        PUSH    BC              ; save BC
0C54  DD2AA863  LD      IX,(#63A8)      ; load IX with screen VRAM address to draw number of meters
0C58  4F        LD      C,A             ; C := A, used for offset
0C59  0600      LD      B,#00           ; B := 0
0C5B  09        ADD     HL,BC           ; add offset
0C5C  7E        LD      A,(HL)          ; get table data
0C5D  DD7760    LD      (IX+#60),A      ; write to screen
0C60  23        INC     HL              ; next
0C61  7E        LD      A,(HL)          ; get data
0C62  DD7740    LD      (IX+#40),A      ; write to screen
0C65  23        INC     HL              ; next
0C66  7E        LD      A,(HL)          ; get table data
0C67  DD7720    LD      (IX+#20),A      ; write to screen
0C6A  DD36E08B  LD      (IX-#20),#8B    ; write "m" to screen
0C6E  C1        POP     BC              ; restore BC
0C6F  DDE5      PUSH    IX              ; transfer IX to HL (part 1/2)
0C71  E1        POP     HL              ; transfer IX to HL (part 2/2)
0C72  11FCFF    LD      DE,#FFFC        ; load offset for next screen location
0C75  19        ADD     HL,DE           ; add offset
0C76  22A863    LD      (#63A8),HL      ; store result
0C79  E1        POP     HL              ; restore HL
0C7A  115FFF    LD      DE,#FF5F        ; load DE with offset for goofy
0C7D  19        ADD     HL,DE           ; add offset to draw next goofy
0C7E  05        DEC     B               ; decrease B.  done drawing goofy kongs ?
0C7F  C2290C    JP      NZ,#0C29        ; no, loop and do another [why not use DJNZ ???]

0BFD  21B375    LD      HL,#75B3        ; load HL with screen location start for goofy kong (original code from #0C26)
0C00  0E50      LD      C,#50           ; C := #50 = start graphic for goofy kong (original code from #0C29)
0C02  71        LD      (HL),C          ; draw part of goofy kong (original code from #0C2B)
0C03  0C        INC     C               ; next graphic (original code from #0C2C)
0C04  2B        DEC     HL              ; next screen location (original code from #0C2D)
0C05  71        LD      (HL),C          ; draw part of goofy kong (original code from #0C2E)
0C06  0C        INC     C               ; next graphic (original code from #0C2F)
0C07  2B        DEC     HL              ; next screen location (original code from #0C30)
0C08  71        LD      (HL),C          ; draw part of goofy kong (original code from #0C31)
0C09  0C        INC     C               ; next graphic (original code from #0C32)
0C0A  2B        DEC     HL              ; next screen location (original code from #0C33)
0C0B  71        LD      (HL),C          ; draw part of goofy kong (original code from #0C34)
0C0C  79        LD      A,C             ; load A with graphic number (original code from #0C35)
0C0D  FE67      CP      #67             ; A == #67 (are we done?) (original code from #0C36)
0C0F  CA820C    JP      Z,#0C82         ; yes, skip forward (original code from #0C38)
0C12  0C        INC     C               ; next graphic (original code from #0C3B)
0C13  112300    LD      DE,#0023        ; load DE with offset (original code from #0C3C)
0C16  19        ADD     HL,DE           ; add offset to screen location (original code from #0C3F)
0C17  C3020C    JP      #0C02           ; loop again (original code from #0C40)

; code to draw the random ladders for the rivets screen

0C1A  DD21353B  LD      IX,#3B35        ; load IX with pointer to ladder datastructure for the rivets screen
0C1E  21006B    LD      HL,#6B00        ; load HL with pointer to output data structure to build ladder definitions

; code to create random ladder definitions
```

```
0C21  3A1860    LD      A,(#6018)       ; get random number for broken ladder number
0C24  E603      AND     #03             ; A is random number between 0 and 3
0C26  C601      ADD     A,#01           ; A is random number between 1 and 4
0C28  32A066    LD      (#66A0),A       ; store in broken ladder number
0C2B  3E00      LD      A,#00           ; A := #00
0C2D  32A166    LD      (#66A1),A       ; initialize ladder counter
0C30  DD7E00    LD      A,(IX+#00)      ; load data element
0C33  FEAA      CP      #AA             ; is this the last data element?
0C35  2002      JR      NZ,#0C39        ; no, skip the next steps
0C37  77        LD      (HL),A          ; yes, write end to data structure
0C38  C9        RET                     ; return
0C39  FEFF      CP      #FF             ; data element is x-value ladder, is this a mirror data element?
0C3B  2009      JR      NZ,#0C46        ; no, skip the next steps
0C3D  DD23      INC     IX              ; yes, increase pointer to input data
0C3F  3EFE      LD      A,#FE           ; load A with mirror offset
0C41  90        SUB     B               ; subtract the previous x-value stored in B
0C42  47        LD      B,A             ; load B with the mirrored x-value
0C43  C3D409    JP      #09D4           ; create the mirrored ladder
0C46  E5        PUSH    HL              ; store HL to the stack for later
0C47  C5        PUSH    BC              ; store BC to the stack for later
0C48  CD7B0A    CALL    #0A7B           ; get random value in A
0C4B  C1        POP     BC              ; restore BC from the stack
0C4C  E1        POP     HL              ; restore HL from the stack
0C4D  E603      AND     #03             ; A is random number between 0 and 3
0C4F  FE00      CP      #00             ; A = #00?
0C51  2005      JR      NZ,#0C58        ; no, skip next steps
0C53  DD4600    LD      B,(IX+#00)      ; use first x-value and store in B
0C56  1815      JR      #0C6D           ; jump forward
0C58  FE01      CP      #01             ; A = #01?
0C5A  2005      JR      NZ,#0C61        ; no, skip next steps
0C5C  DD4601    LD      B,(IX+#01)      ; use second x-value and store in B
0C5F  180C      JR      #0C6D           ; jump forward
0C61  FE02      CP      #02             ; A = #02?
0C63  2005      JR      NZ,#0C6A        ; no, skip next steps
0C65  DD4602    LD      B,(IX+#02)      ; use third x-value and store in B
0C68  1803      JR      #0C6D           ; jump forward
0C6A  DD4603    LD      B,(IX+#03)      ; use fourth x-value and store in B
0C6D  DD23      INC     IX              ; increase pointer to input data
0C6F  DD23      INC     IX              ; increase pointer to input data
0C71  DD23      INC     IX              ; increase pointer to input data
0C73  DD23      INC     IX              ; increase pointer to input data
0C75  C3C909    JP      #09C9           ; continue code to create non-mirrored ladder

0C78  00000000000000000000              ; no operations - free space

0C82  110703    LD      DE,#0307        ; load task data for text #7 "HOW HIGH CAN YOU GET?"
0C85  CD9F30    CALL    #309F           ; insert task to draw text
0C88  210960    LD      HL,WaitTimerMSB ; load HL with timer to wait
0C8B  36A0      LD      (HL),#A0        ; set timer for #A0 units
0C8D  23        INC     HL              ; HL := GameMode2
0C8E  34        INC     (HL)            ;
0C8F  34        INC     (HL)            ; increase game mode twice - starts game
0C90  C9        RET                     ; return

; arrive here from #0701 when game mode = 9
; clears screen, update timers, draws current screen, sets background music,

0C91  DF        RST     #18             ; count down WaitTimerMSB and only continue when 0

; arrive here from #0776 during attract mode

0C92  CD7408    CALL    #0874           ; clears the screen and sprites
0C95  AF        XOR     A               ; A := 0
0C96  328C63    LD      (#638C),A       ; reset onscreen timer
0C99  110105    LD      DE,#0501        ; load DE with task #5, parameter 1 update onscreen bonus timer and play sound &
change to red if below 1000
0C9C  CD9F30    CALL    #309F           ; insert task
0C9F  21867D    LD      HL,REG_PALETTE_A        ; load HL with palette bank selector
0CA2  3600      LD      (HL),#00        ; clear palette bank selector
0CA4  23        INC     HL              ; next bank
0CA5  3601      LD      (HL),#01        ; set palette bank selector
0CA7  3A2762    LD      A,(#6227)       ; load A with screen number
0CAA  3D        DEC     A               ; decrease by 1
0CAB  CAD40C    JP      Z,#0CD4         ; if zero jump to #0Cd4 - we were on girders - continue on #0CC6

0CAE  3D        DEC     A               ; if not decrease a again
0CAF  CADF0C    JP      Z,#0CDF         ; if zero jump to #0CDf - we were on pie - continue on #0CC6

0CB2  3D        DEC     A               ; if not decrease a again
0CB3  CAF20C    JP      Z,#0CF2         ; iF zero jump to #0CF2 - we were on elevators - continue on #0CC6

                                        ; else we are on rivets

0CB6  CD430D    CALL    #0D43           ; draws the blue vertical bars next to kong on rivets
0CB9  21867D    LD      HL,REG_PALETTE_A        ; load HL with palette bank selector
0CBC  3601      LD      (HL),#01        ; set palette bank selector
0CBE  3E0B      LD      A,#0B           ; load A with music code For rivets
0CC0  328960    LD      (#6089),A       ; set music
0CC3  118B3C    LD      DE,#3C8B        ; load DE with start of table data for rivets
0CC6  CDA70D    CALL    #0DA7           ; draw the screen
0CC3  116C3B    LD      DE,#3B6C        ; load DE with adapted start of table data for rivets
0CC6  CD4413    CALL    #1344           ; draw rivets screen including the random ladders

0CC9  3A2762    LD      A,(#6227)       ; load A with screen number
```

```
0CCC  FE04      CP      #04             ; screen is rivets level?
0CCE  CC000D    CALL    Z,#0D00         ; yes, call sub to draw the rivets

0CD1  C3A03F    JP      #3FA0           ; fix rectractable ladders for pie factory and returns to #0D5F. [orig code was JP
#0D5F ?]

; girders from #0CAB

0CD4  11E43A    LD      DE,#3AE4        ; Load DE with start of table data for girders
0CD7  3E08      LD      A,#08           ; A := 8 = music code for girders
0CD9  328960    LD      (#6089),A       ; set music for girders
0CDC  C3C60C    JP      #0CC6           ; jump back

; conveyors from #0CAF

0CDF  115D3B    LD      DE,#3B5D        ; load DE with start of table data for conveyors
0CDF  11AF3B    LD      DE,#3BAF        ; load DE with adapted start of table data for conveyors
0CE2  21867D    LD      HL,REG_PALETTE_A        ; load HL with palette bank selector
0CE5  3601      LD      (HL),#01        ; set palette bank selector
0CE7  23        INC     HL              ;
0CE8  3600      LD      (HL),#00        ; clear palette bank selector
0CEA  3E09      LD      A,#09           ; load A with conveyor music
0CEC  328960    LD      (#6089),A       ; set music for conveyors
0CEF  C3C60C    JP      #0CC6           ; jump back

; elevators from #0CB3

0CF2  CD270D    CALL    #0D27           ; draw elevator cables
0CF5  3E0A      LD      A,#0A           ; A := #A
0CF7  328960    LD      (#6089),A       ; set music for elevators
0CFA  11E53B    LD      DE,#3BE5        ; load DE with start of table data for the elevators
0CFA  115A3C    LD      DE,#3C5A        ; load DE with adapted start of table data for elevators
0CFD  C3C60C    JP      #0CC6           ; jump back

; For the rivets level only  - draw the rivets

0D00  0608      LD      B,#08           ; for B = 1 to 8 rivets to draw
0D02  21170D    LD      HL,#0D17        ; load HL with start of table data below

0D05  3EB8      LD      A,#B8           ; load A with #B8 = start code for rivet
0D07  0E02      LD      C,#02           ; For C = 1 to 2
0D09  5E        LD      E,(HL)          ; load E with the high byte of the address
0D0A  23        INC     HL              ; next HL
0D0B  56        LD      D,(HL)          ; load D with the low byte of the adddress
0D0C  23        INC     HL              ; next HL

0D0D  12        LD      (DE),A          ; draw rivet onscreen
0D0E  3D        DEC     A               ; next graphic
0D0F  13        INC     DE              ; next screen address
0D10  0D        DEC     C               ; Next C
0D11  C20D0D    JP      NZ,#0D0D        ; loop until done

0D14  10EF      DJNZ    #0D05           ; Next B

0D16  C9        RET                     ; return

; start of table data for rivets used above
; these are addresses in video RAM for the rivets

0D17  CA 76             ; #76CA
0D19  CF 76             ; #76CF
0D1B  D4 76             ; #76D4
0D1D  D9 76             ; #76D9
0D1F  2A 75             ; #752A
0D21  2F 75             ; #752F
0D23  34 75             ; #7534
0D25  39 75             ; #7539

; called from #0CF2 for elevators only
; draws the elevator cables

0D27  210D77    LD      HL,#770D        ; load HL with screen RAM location
0D2A  CD300D    CALL    #0D30           ; draw the left side elevator cable

0D2D  210D76    LD      HL,#760D        ; load HL with screen RAM location for right side cable

0D30  0611      LD      B,#11           ; for B = 1 to #11

0D32  36FD      LD      (HL),#FD        ; draw the cable to screen
0D34  23        INC     HL              ; next location
0D35  10FB      DJNZ    #0D32           ; Next B

0D37  110F00    LD      DE,#000F        ; load DE with offset [why here? should be before loop starts ?]
0D3A  19        ADD     HL,DE           ; add offset to location
0D3B  0611      LD      B,#11           ; for B = 1 to #11

0D3D  36FC      LD      (HL),#FC        ; draw cable to screen
0D3F  23        INC     HL              ; next location
0D40  10FB      DJNZ    #0D3D           ; Next B

0D42  C9        RET                     ; return

; called from #0CB6 for rivets only
; draws top light blue vertical bars next to Kong
```

```
0D43  218776    LD      HL,#7687        ; load HL with screen location (left side)
0D46  CD4C0D    CALL    #0D4C           ; draw the bars
0D49  214775    LD      HL,#7547        ; load HL with screen location (right side)
0D4C  0604      LD      B,#04           ; for B = 1 to 4

0D4E  36FD      LD      (HL),#FD        ; draw a bar
0D50  23        INC     HL              ; next screen location
0D51  10FB      DJNZ    #0D4E           ; Next B

0D53  111C00    LD      DE,#001C        ; load offset
0D56  19        ADD     HL,DE           ; add offset
0D57  0604      LD      B,#04           ; for B = 1 to 4

0D59  36FC      LD      (HL),#FC        ; draw a bar
0D5B  23        INC     HL              ; next screen location
0D5C  10FB      DJNZ    #0D59           ; next B

0D5E  C9        RET                     ; return

; jump here from #0CD1 (via #3FA3)

0D5F  CD560F    CALL    #0F56           ; clear and initialize RAM values, compute initial timer, draw all initial sprites
0D62  CD4124    CALL    #2441           ;
0D62  C3FA09    JP      #09FA           ; jump to additional code to activate the ladders
0D65  210960    LD      HL,WaitTimerMSB ; load HL with timer addr.
0D68  3640      LD      (HL),#40        ; set timer to #40
0D6A  23        INC     HL              ; HL := GameMode2
0D6B  34        INC     (HL)            ; increase game mode2
0D6C  215C38    LD      HL,#385C        ; load HL with start of kong graphic table data
0D6F  CD4E00    CALL    #004E           ; update kong's sprites

0D72  110069    LD      DE,#6900        ; set destination to girl sprite
0D75  010800    LD      BC,#0008        ; set counter to 8
0D78  EDB0      LDIR                    ; draw the girl on screen

0D7A  3A2762    LD      A,(#6227)       ; load a with screen number
0D7D  fe04      CP      #04             ; is this rivets screen?
0D7f  280A      JR      Z,#0D8b         ; if yes, jump ahead a bit

0D81  0F        RRCA                    ; no, roll right twice
0D82  0F        RRCA                    ; is this the conveyors or the elevators ?
0D83  d8        RET     c               ; yes, return

                                        ; else this is girders, kong needs to be moved

0D84  210B69    LD      HL,#690B        ; load HL with start of kong sprite
0D87  0EFC      LD      C,#FC           ; set to move by -4
0D89  FF        RST     #38             ; move kong
0D8A  C9        RET                     ; return

; on the rivets

0D8B  210869    LD      HL,#6908        ; load HL with kong sprite RAM
0D8E  0E44      LD      C,#44           ; set counter to #44 ?
0D90  FF        RST     #38             ; move kong

0D91  110400    LD      DE,#0004        ; load counters
0D94  011002    LD      BC,#0210        ; load counters
0D97  210069    LD      HL,#6900        ; load HL with start of sprite RAM (girl sprite first)
0D9A  CD3D00    CALL    #003D           ; move girl to right

0D9D  01F802    LD      BC,#02F8        ; load counters
0DA0  210369    LD      HL,#6903        ; load HL with Y value of girl -1
0DA3  CD3D00    CALL    #003D           ; move girl up

0DA6  C9        RET                     ; return [to #1983]

; part of routine which draws the screen
; DE is preloaded with address of table data
; called from many places

0DA7  1A        LD      A,(DE)          ; load a with DE - points to start of table data
0DA8  32B363    LD      (#63B3),A       ; save for later use
0DAB  FEAA      CP      #AA             ; is this the end of the data?
0DAD  C8        RET     Z               ; yes, return

; else draw screen stuff

0DAE  13        INC     DE              ; next table entry
0DAF  1A        LD      A,(DE)          ; load A with table data
0DB0  67        LD      H,A             ; copy to H
0DB1  44        LD      B,H             ; copy to B
0DB2  13        INC     DE              ; next table entry
0DB3  1A        LD      A,(DE)          ; load A with table data
0DB4  6F        LD      L,A             ; copy to L
0DB5  4D        LD      C,L             ; copy to C
0DB6  D5        PUSH    DE              ; save DE
0DB7  CDF02F    CALL    #2FF0           ; convert HL into VRAM address
0DBA  D1        POP     DE              ; restore DE
0DBB  22AB63    LD      (#63AB),HL      ; store the VRAM address into this location for later use.  starting point of
whatever we are drawing
0DBE  78        LD      A,B             ; A := B = original data item
0DBF  E607      AND     #07             ; mask bits, now between 0 and 7
0DC1  32B463    LD      (#63B4),A       ; store into ???
0DC4  79        LD      A,C             ; A := C = 2nd data item
```

```
0DC5  E607      AND     #07             ; mask bits, now between 0 and 7
0DC7  32AF63    LD      (#63AF),A       ; store into ???
0DCA  13        INC     DE              ; next table entry
0DCB  1A        LD      A,(DE)          ; load A with table data
0DCC  67        LD      H,A             ; copy to H
0DCD  90        SUB     B               ; subract the original data.  less than zero?
0DCE  D2D30D    JP      NC,#0DD3        ; no, skip next step

0DD1  ED44      NEG                     ; Negate A (A := #FF - A)

0DD3  32B163    LD      (#63B1),A       ; store into ???
0DD6  13        INC     DE              ; next table entry
0DD7  1A        LD      A,(DE)          ; load A with table data
0DD8  6F        LD      L,A             ; copy to L
0DD9  91        SUB     C               ; subtract the 2nd data item
0DDA  32B263    LD      (#63B2),A       ; store into ???
0DDD  1A        LD      A,(DE)          ; load A with same table data
0DDE  E607      AND     #07             ; mask bits, now between 0 and 7
0DE0  32B063    LD      (#63B0),A       ; store into ???
0DE3  D5        PUSH    DE              ; save DE
0DE4  CDF02F    CALL    #2FF0           ; convert HL into VRAM address
0DE7  D1        POP     DE              ; restore DE
0DE8  22AD63    LD      (#63AD),HL      ; store into ???
0DEB  3AB363    LD      A,(#63B3)       ; load A with first data item
0DEE  FE02      CP      #02             ; < 2 ? are we drawing a ladder or a broken ladder?
0DF0  F24F0E    JP      P,#0E4F         ; no, skip ahead [why P, instead of NC ?]

; else we are drawing a ladder

0DF3  3AB263    LD      A,(#63B2)       ; load A with ???
0DF6  D610      SUB     #10             ; subtract #10
0DF8  47        LD      B,A             ; copy answer to B
0DF9  3AAF63    LD      A,(#63AF)       ; load A with ???
0DFC  80        ADD     A,B             ; add B
0DFD  32B263    LD      (#63B2),A       ; store into ???
0E00  3AAF63    LD      A,(#63AF)       ; load A with ??? computed above
0E03  C6F0      ADD     A,#F0           ; add #F0
0E05  2AAB63    LD      HL,(#63AB)      ; load HL with VRAM address to begin drawing
0E08  77        LD      (HL),A          ; draw element to screen = girder above top of ladder ?
0E09  2C        INC     L               ; next location
0E0A  D630      SUB     #30             ; subtract #30.  now the element to draw is a ladder
0E0C  77        LD      (HL),A          ; draw element to screen = top of ladder
0E0D  3AB363    LD      A,(#63B3)       ; load A with original data item
0E10  FE01      CP      #01             ; == 1 ? (is this a broken ladder?)
0E12  C2190E    JP      NZ,#0E19        ; no, skip next 2 steps

0E15  AF        XOR     A               ; A := 0
0E16  32B263    LD      (#63B2),A       ; store into ???

0E19  3AB263    LD      A,(#63B2)       ; load A with ???
0E1C  D608      SUB     #08             ; subtract 8
0E1E  32B263    LD      (#63B2),A       ; store.  are we done?
0E21  DA2A0E    JP      C,#0E2A         ; yes, skip ahead

0E24  2C        INC     L               ; next HL
0E25  36C0      LD      (HL),#C0        ; draw ladder to screen
0E27  C3190E    JP      #0E19           ; loop again

0E2A  3AB063    LD      A,(#63B0)       ; load A with ???
0E2D  C6D0      ADD     A,#D0           ; add #D0
0E2F  2AAD63    LD      HL,(#63AD)      ;
0E2F  C34E3F    JP      #3F4E           ; jump to additional code to fix for small broken ladders that appear as normal
0E32  77        LD      (HL),A
0E33  3AB363    LD      A,(#63B3)       ; load A with original data item
0E36  FE01      CP      #01             ; == 1 ?  (is this a broken ladder ?)
0E38  C23F0E    JP      NZ,#0E3F        ; no, skip next 3 steps

; this is a broken ladder.  draw bottom part of ladder

0E3B  2D        DEC     L               ; decrease HL
0E3C  36C0      LD      (HL),#C0        ; set HL to #C0   draws bottom part of broken ladder to screen
0E3E  2C        INC     L               ; increase HL
0E3B  C3383F    JP      #3F38           ; jump to additional code to fix for small broken ladders that appear as normal
0E3E  00        NOP                     ; no operation

0E3F  3AB063    LD      A,(#63B0)       ; load A with ???
0E42  FE00      CP      #00             ; == 0 ?
0E44  CA4B0E    JP      Z,#0E4B         ; yes, skip next 3 steps

0E47  C6E0      ADD     A,#E0           ; add #E0
0E49  2C        INC     L               ; next HL
0E4A  77        LD      (HL),A          ; store into ???

0E4B  13        INC     DE              ; next table entry
0E4C  C3A70D    JP      #0DA7           ; loop again

; arrive from #0DF0

0E4F  3AB363    LD      A,(#63B3)       ; load A with original data item [why do this again ?  it was loaded just before
coming here]
0E52  FE02      CP      #02             ; == 2 ?
0E54  C2E80E    JP      NZ,#0EE8        ; no, skip ahead

; else data item type 2 = girder ???
```

```
0E57  3AAF63    LD      A,(#63AF)       ; load A with original data item #2, masked to be between 0 and 7
0E5A  C6F0      ADD     A,#F0           ; add #F0
0E5C  32B563    LD      (#63B5),A       ; store into ???
0E5F  2AAB63    LD      HL,(#63AB)      ; load HL with screen address to being drawing the item

0E62  3AB563    LD      A,(#63B5)       ; load A with ???
0E65  77        LD      (HL),A          ; draw element to screen
0E66  23        INC     HL              ; next screen location
0E67  7D        LD      A,L             ; A := L
0E68  E61F      AND     #1F             ; mask bits, now between 0 and #1F.  at zero ?
0E6A  CA780E    JP      Z,#0E78         ; yes, skip ahead

0E6D  3AB563    LD      A,(#63B5)       ; load A with ???
0E70  FEF0      CP      #F0             ; == #F0 ?
0E72  CA780E    JP      Z,#0E78         ; yes, skip next 2 steps

0E75  D610      SUB     #10             ; subtract #10
0E77  77        LD      (HL),A          ; store

0E78  011F00    LD      BC,#001F        ; load BC with offset
0E7B  09        ADD     HL,BC           ; add offset to HL
0E7C  3AB163    LD      A,(#63B1)       ; load A with ???
0E7F  D608      SUB     #08             ; subtract 8.  done?
0E81  DACF0E    JP      C,#0ECF         ; yes, skip ahead for next

0E84  32B163    LD      (#63B1),A       ; store A into ???
0E87  3AB263    LD      A,(#63B2)       ; load A with ???
0E8A  FE00      CP      #00             ; == 0 ? [why written this way?]
0E8C  CA620E    JP      Z,#0E62         ; yes, jump back and draw another [of same?]

0E8F  3AB563    LD      A,(#63B5)
0E92  77        LD      (HL),A          ; draw element to screen
0E93  23        INC     HL              ; next screen location
0E94  7D        LD      A,L             ; A := L
0E95  E61F      AND     #1F             ; mask bits, now between 0 and #1F.  at zero?
0E97  CAA00E    JP      Z,#0EA0         ; yes, skip next 3 steps

0E9A  3AB563    LD      A,(#63B5)       ; load A with ???
0E9D  D610      SUB     #10             ; subtract #10
0E9F  77        LD      (HL),A          ; store to screen.  draws bottom half of a girder

0EA0  011F00    LD      BC,#001F        ; load BC with offset
0EA3  09        ADD     HL,BC           ; add offset for next screen element
0EA4  3AB163    LD      A,(#63B1)       ; load A with ???
0EA7  D608      SUB     #08             ; subtract 8.  done?
0EA9  DACF0E    JP      C,#0ECF         ; yes, skip ahead for next

0EAC  32B163    LD      (#63B1),A       ; store A into ???
0EAF  3AB263    LD      A,(#63B2)       ; load A with ???
0EB2  CB7F      BIT     7,A             ; test bit 7.  is it zero?
0EB4  C2D30E    JP      NZ,#0ED3        ; no, skip ahead

0EB7  3AB563    LD      A,(#63B5)       ; load A with ???
0EBA  3C        INC     A               ; increase
0EBB  32B563    LD      (#63B5),A       ; store result
0EBE  FEF8      CP      #F8             ; == #F8 ?
0EC0  C2C90E    JP      NZ,#0EC9        ; no, skip next 3 steps

0EC3  23        INC     HL              ; next screen location
0EC4  3EF0      LD      A,#F0           ; A := #F0
0EC6  32B563    LD      (#63B5),A       ; store into ???

0EC9  7D        LD      A,L             ; A := L
0ECA  E61F      AND     #1F             ; mask bits.  now between 0 and #1F.  at zero?
0ECC  C2620E    JP      NZ,#0E62        ; no, jump back

0ECF  13        INC     DE              ; next table entry
0ED0  C3A70D    JP      #0DA7           ; loop back for more

0ED3  3AB563    LD      A,(#63B5)       ; load A with ???
0ED6  3D        DEC     A               ; decrease
0ED7  32B563    LD      (#63B5),A       ; store result
0EDA  FEF0      CP      #F0             ; compare to #F0.  is the sign positive?
0EDC  F2E50E    JP      P,#0EE5         ; yes, skip next 3 steps [why?  #0EE5 is a jump - it should jump directly instead]

0EDF  2B        DEC     HL              ;
0EE0  3EF7      LD      A,#F7           ; A := #F7
0EE2  32B563    LD      (#63B5),A       ; store into ???

0EE5  C3620E    JP      #0E62           ; jump back

; arrive from #0E54

0EE8  3AB363    LD      A,(#63B3)       ; load A with original data item [why load it again ? A already has #63B3]
0EEB  FE03      CP      #03             ; == 3?
0EED  C21B0F    JP      NZ,#0F1B        ; no, skip ahead

; we are drawing a conveyor

0EF0  2AAB63    LD      HL,(#63AB)      ; load HL with VRAM screen address to begin drawing
0EF3  3EB3      LD      A,#B3           ; A := #B3 = code graphic for conveyor
0EF5  77        LD      (HL),A          ; draw on screen
0EF6  012000    LD      BC,#0020        ; load BC with offset
0EF9  09        ADD     HL,BC           ; add offset to HL
0EFA  3AB163    LD      A,(#63B1)       ; load A with ???
```

```
0EFD  D610      SUB     #10             ; subtract #10.  done ?

0EFF  DA140F    JP      C,#0F14         ; yes, skip ahead

0F02  32B163    LD      (#63B1),A       ;
0F05  3EB1      LD      A,#B1           ; A := #B1
0F07  77        LD      (HL),A          ; store into ???
0F08  012000    LD      BC,#0020        ; load BC with offset
0F0B  09        ADD     HL,BC           ; add offset to HL
0F0C  3AB163    LD      A,(#63B1)       ; load A with ???
0F0F  D608      SUB     #08             ; subtract 8
0F11  C3FF0E    JP      #0EFF           ; loop again

0F14  3EB2      LD      A,#B2           ; A := #B2
0F16  77        LD      (HL),A          ; store (onscreen???)
0F17  13        INC     DE              ; next table entry
0F18  C3A70D    JP      #0DA7           ; loop back for more

; arrive from #0EED

0F1B  3AB363    LD      A,(#63B3)       ; load A with original data item [why load it again ? A already has #63B3]
0F1E  FE07      CP      #07             ; <= 7 ?
0F20  F2CF0E    JP      P,#0ECF         ; no, skip back and loop for next data item

0F23  FE04      CP      #04             ; first data item == 4 ?
0F25  CA4C0F    JP      Z,#0F4C         ; yes, skip ahead to handle

0F28  FE05      CP      #05             ; first data item == 5 ?
0F2A  CA510F    JP      Z,#0F51         ; yes, skip ahead to handle

; redraws screen when rivets has been completed

0F2D  3EFE      LD      A,#FE           ; A := #FE

0F2F  32B563    LD      (#63B5),A       ; store into ???
0F32  2AAB63    LD      HL,(#63AB)      ; load HL with ???

0F35  3AB563    LD      A,(#63B5)       ; load A with ???
0F38  77        LD      (HL),A          ; store into ???
0F39  012000    LD      BC,#0020        ; set offset to #20
0F3C  09        ADD     HL,BC           ; add offset for next
0F3D  3AB163    LD      A,(#63B1)       ; load A with ???
0F40  D608      SUB     #08             ; subtract 8
0F42  32B163    LD      (#63B1),A       ; store result.  done ?
0F45  D2350F    JP      NC,#0F35        ; no, loop again

0F48  13        INC     DE              ; else increase DE
0F49  C3A70D    JP      #0DA7           ; jump back

0F4C  3EE0      LD      A,#E0           ; A := #E0
0F4E  C32F0F    JP      #0F2F           ; jump back

0F51  3EB0      LD      A,#B0           ; A := #B0
0F53  C32F0F    JP      #0F2F           ; jump back

; called from #0D5F
; clears memories from #6200 - 6227 and #6280 to 6B00
; [why are #6280 - #6280+40 cleared?  they are set immediately after]
; computes initial timer
; initializes all sprites

0F56  0627      LD      B,#27           ; for B = 1 to #27
0F58  210062    LD      HL,#6200        ; load HL with start of address
0F5B  AF        XOR     A               ; A := #00

0F5C  77        LD      (HL),A          ; clear memory
0F5D  2C        INC     L               ; next
0F5E  10FC      DJNZ    #0F5C           ; next B

0F60  0E11      LD      C,#11           ; For C = 1 to 11
0F62  1680      LD      D,#80           ; load D with 80, used to reset B in inner loop
0F64  218062    LD      HL,#6280        ; start of memory to clear
0F67  42        LD      B,D             ; For B = 1 to #80

0F68  77        LD      (HL),A          ; clear (HL)
0F69  23        INC     HL              ; next memory
0F6A  10FC      DJNZ    #0F68           ; Next B

0F6C  0D        DEC     C               ; Next C
0F6D  20F8      JR      NZ,#0F67        ; loop until done

0F6F  219C3D    LD      HL,#3D9C        ; source addr. = #3D9C - table data
0F72  118062    LD      DE,#6280        ; Destination = #6280
0F75  014000    LD      BC,#0040        ; counter = #40 Bytes
0F78  EDB0      LDIR                    ; copy


;;; values are copied into #6280 through #6280 + #40
;;;     3D9C:                                       00 00 23 68
;;;     3DA0:  01 11 00 00 00 10 DB 68 01 40 00 00 08 01 01 01
;;;     3DB0:  01 01 01 01 01 01 00 00 00 00 00 00 80 01 C0 FF
;;;     3DC0:  01 FF FF 34 C3 39 00 67 80 69 1A 01 00 00 00 00
;;;     3DD0:  00 00 00 00 04 00 10 00 00 00 00 00
;;;
```

```
; set up initial timer
; timer is either 5000, 6000, 7000 or 8000 depending on level

0F7A  3A2962    LD      A,(#6229)       ; load level number
0F7D  47        LD      B,A             ; copy to B
0F7E  A7        AND     a               ; clear carry flag
0F7f  17        RLA                     ; rotate A left (double =2x)
0F80  A7        AND     a               ; clear carry flag
0F81  17        RLA                     ; rotate A left (double again =4x)
0F82  A7        AND     a               ; clear carry flag
0F83  17        RLA                     ; rotate A left (double again = 8x)
0F84  80        ADD     A,b             ; add B into A  (add once = 9x)
0F85  80        ADD     A,b             ; add B  into A  (add again = 10x)
0F86  C628      ADD     A,#28           ; add #28 (40 decimal) to A
0F88  FE51      CP      #51             ; < #51 ?
0F8A  3802      JR      c,#0F8E         ; yes, skip next step

0F8C  3E50      LD      A,#50           ; otherwise load A with #50 (80 decimal)

0F8E  21B062    LD      HL,#62B0        ; load HL with start of timers
0F91  0603      LD      B,#03           ; For B = 1 to 3

0F93  77        LD      (HL),A          ; store A into timer memory
0F94  2C        INC     l               ; next memory
0F95  10Fc      DJNZ    #0F93           ; Next B

0F97  87        ADD     A,A             ; add A with A (double a).  A is now #64, #78, #8C, or #A0
0F98  47        LD      B,A             ; copy to B
0F99  3EDC      LD      A,#DC           ; A := #DC (220 decimal)
0F9B  90        SUB     B               ; subtract B.  answers are #78, #64, #50, or #3C
0F9C  FE28      CP      #28             ; is this less than #28 (40 decimal) ?  (will never get this ... ???)
0F9E  3002      JR      NC,#0FA2        ; no, skip next step

0FA0  3E28      LD      A,#28           ; else load a with #28 (40). minimum value (never get this ... ?????)

0FA2  77        LD      (HL),A          ; store A into address of HL=#62B3 which controls timers
0FA3  2C        INC     L               ; HL := #62B4
0FA4  77        LD      (HL),A          ; store A into the timer control
0FA5  210962    LD      HL,#6209        ; load HL with #6209
0FA8  3604      LD      (HL),#04        ; store 4 into #6209
0FAA  2C        INC     L               ; HL := #620A
0FAB  3608      LD      (HL),#08        ; store 8 into #620A
0FAD  3A2762    LD      A,(#6227)       ; load A with screen number
0FB0  4F        LD      C,A             ; copy to C, used at #0FCB
0FB1  CB57      BIT     2,A             ; is this the rivets ?
0FB3  2016      JR      NZ,#0FCB        ; yes, skip ahead [would be better to jump to #1131, or JR to #0FCC]

; draw 3 black sprites above the top kongs ladder
; effect to erase the 2 girders at the top of kong's ladder

0FB5  21006A    LD      HL,#6A00        ; else load HL sprite RAM - used for blank space sprite
0FB8  3E4F      LD      A,#4f           ; A := #4F = X position of this sprite
0FBA  0603      LD      B,#03           ; For B = 1 to 3

0FBC  77        LD      (HL),A          ; set the sprite X position
0FBD  2C        INC     L               ; next address = sprite type
0FBE  363A      LD      (HL),#3A        ; set sprite type as blank square
0FC0  2C        INC     L               ; next address = sprite color
0FC1  360F      LD      (HL),#0F        ; set color to black
0FC3  2C        INC     L               ; next address = sprite Y position
0FC4  3618      LD      (HL),#18        ; set sprite Y position to #18
0FC6  2C        INC     L               ; next memory
0FC7  C610      ADD     A,#10           ; A := A + #10 to adjust for next X position
0FC9  10F1      DJNZ    #0FBC           ; Next B

0FCB  79        LD      A,C             ; load A with screen number
0FCC  EF        RST     #28             ; jump depending on the screen

; jump table data

0FCD  00 00                             ; unused
0FCF  D7 0F                             ; #0FD7 for girders
0FD1  1F 10                             ; #101F for conveyors
0FD3  87 10                             ; #1087 for elevators
0FD5  31 11                             ; #1131 for rivets

; arrive here when playing girders

0FD7  21DC3D    LD      HL,#3DDC        ; source - has the information about the barrel pile at #3DDC
0FDA  11A869    LD      DE,#69A8        ; destination = sprites
0FDD  011000    LD      BC,#0010        ; counter is #10
0FE0  EDB0      LDIR                    ; draws the barrels pile next to kong

0FE2  21EC3D    LD      HL,#3DEC        ; set up a copy job from table in #3DEC
0FE5  110764    LD      DE,#6407        ; destination in memory is #6407
0FE8  0E1C      LD      C,#1C           ; #1C is a secondary counter
0FEA  0605      LD      B,#05           ; #05 is a secondary counter
0FEC  CD2A12    CALL    #122A           ; copy

0FEF  21F43D    LD      HL,#3DF4        ; load HL with table data start for initial fire locations
0FF2  CDFA11    CALL    #11FA           ; ???

0FF5  21003E    LD      HL,#3E00        ; source table at #3E00 = oil can
0FF8  11FC69    LD      DE,#69FC        ; destination sprite at #69FC
0FFB  010400    LD      BC,#0004        ; 4 bytes
```

```
0FFE  EDB0      LDIR                    ; draw to screen

1000  210C3E    LD      HL,#3E0C        ; load HL with table data for hammers on girders
1003  CDA611    CALL    #11A6           ; ???

1006  211B10    LD      HL,#101B        ; set up copy job from table in #101B
1009  110767    LD      DE,#6707        ; set destination ?
100C  011C08    LD      BC,#081C        ; set counters ?
100F  CD2A12    CALL    #122A           ; copy

1012  110768    LD      DE,#6807        ; set destination ?
1015  0602      LD      B,#02           ; set counter to 2
1017  CD2A12    CALL    #122A           ; copy
101A  C9        RET                     ; return

; data used in sub at #1006

101B  00
101C  00
101D  02
101E  02

; arrive here when conveyors starts
; draws parts of the screen

101F  21EC3D    LD      HL,#3DEC        ; set up a copy job from table in #3DEC
1022  110764    LD      DE,#6407        ; desitnation in memory is #6407
1025  011C05    LD      BC,#051C        ; counters are #05 and #1C
1028  CD2A12    CALL    #122A           ; copy

102B  CD8611    CALL    #1186

102E  21183E    LD      HL,#3E18        ; set up copy job from table in #3E18
1031  11A765    LD      DE,#65A7        ; destination is #65A7
1034  010C06    LD      BC,#060C        ; counters are #05 and #0C
1037  CD2A12    CALL    #122A           ; copy

103A  DD21A065  LD      IX,#65A0        ; load IX with start of pies
103E  21B869    LD      HL,#69B8        ; load HL with sprites for pies
1041  111000    LD      DE,#0010        ; DE := #10
1044  0606      LD      B,#06           ; B := 6
1046  CDD311    CALL    #11D3

1049  21FA3D    LD      HL,#3DFA        ; load HL with start of table data
104C  CDFA11    CALL    #11FA           ; set fireball sprite

104F  21043E    LD      HL,#3E04        ; set up copy job from table in #3E04 = oil can sprite
1052  11FC69    LD      DE,#69FC        ; destination is #69FC = sprite
1055  010400    LD      BC,#0004        ; four bytes to copy
1058  EDB0      LDIR                    ; draw oil can

105A  211C3E    LD      HL,#3E1C        ; load HL with start of table data
105D  114469    LD      DE,#6944        ; load DE with sprite start for moving ladders
1060  010800    LD      BC,#0008        ; set byte counter to 8
1063  EDB0      LDIR                    ; draw moving ladders

1065  21243E    LD      HL,#3E24        ; set source table data
1068  11E469    LD      DE,#69E4        ; set destination RAM sprites
106B  011800    LD      BC,#0018        ; set counter
106E  EDB0      LDIR                    ; draw pulleys

1070  21103E    LD      HL,#3E10        ; load HL with table data for hammers on conveyors
1073  CDA611    CALL    #11A6           ; ???

1076  213C3E    LD      HL,#3E3C        ; load HL with table data for bonus items on conveyors
1079  110C6A    LD      DE,#6A0C        ; load DE with sprite destination
107C  010C00    LD      BC,#000C        ; 3 items x 4 bytes = 12 bytes (#0C)
107F  EDB0      LDIR                    ; draw bonus item sprites

1081  3E01      LD      A,#01           ; A := 1
1083  32B962    LD      (#62B9),A       ; store into fire release
1086  C9        RET                     ; return

; arrive here when elevators starts

1087  21EC3D    LD      HL,#3DEC        ; load HL with start of table data
108A  110764    LD      DE,#6407        ; set destination ???
108D  011C05    LD      BC,#051C        ; set counters
1090  CD2A12    CALL    #122A           ; copy ???

1093  CD8611    CALL    #1186

1096  210066    LD      HL,#6600        ; load HL with start of elevator sprites ???
1099  111000    LD      DE,#0010        ; load DE with offset to add
109C  3E01      LD      A,#01           ; A := 1
109E  0606      LD      B,#06           ; for B = 1 to 6

10A0  77        LD      (HL),A          ; write value into memory
10A1  19        ADD     HL,DE           ; add offset for next
10A2  10FC      DJNZ    #10A0           ; next B

10A4  0E02      LD      C,#02           ; For C = 1 to 2
10A6  3E08      LD      A,#08           ; A := 8
10A8  0603      LD      B,#03           ; for B = 1 to 3
10AA  210D66    LD      HL,#660D        ; load HL with ???
```

```
10AD  77        LD      (HL),A          ; write value into memory
10AE  19        ADD     HL,DE           ; add offset for next
10AF  10FC      DJNZ    #10AD           ; next B

10B1  3E08      LD      A,#08           ; A := 8 [why?  A is already 8]
10B3  0D        DEC     C               ; next C
10B4  C2A810    JP      NZ,#10A8        ; loop until done

; used to draw elevator platforms???

; #6600 - 665F  = the 6 elevator values.  6610, 6620, 6630, 6640 ,6650 are starting values
;       + 3 is the X position, + 5 is the Y position

10B7  21643E    LD      HL,#3E64        ; start of table data
10BA  110366    LD      DE,#6603        ; Destination sprite ? X positions ?
10BD  010E06    LD      BC,#060E        ; Counter = #06, offset = #0E
10C0  CDEC11    CALL    #11EC           ; set items from data table

10C3  21603E    LD      HL,#3E60        ; start of table data
10C6  110766    LD      DE,#6607        ; Destination sprite ?
10C9  010C06    LD      BC,#060C        ; B = 6 is loop variable, C = offset ?
10CC  CD2A12    CALL    #122A           ;

10CF  DD210066  LD      IX,#6600        ; load IX with ???
10D3  215869    LD      HL,#6958        ; load HL with elevator sprites start
10D6  0606      LD      B,#06           ; B := 6
10D8  111000    LD      DE,#0010        ; load offset with #10
10DB  CDD311    CALL    #11D3           ; ???

10DE  21483E    LD      HL,#3E48        ; source is data table for bonus items on elevators
10E1  110C6A    LD      DE,#6A0C        ; destination is RAM area for bonus items
10E4  010C00    LD      BC,#000C        ; counter set for #0C bytes
10E7  EDB0      LDIR                    ; copy

; set up the 2 fireballs

10E9  DD210064  LD      IX,#6400        ; load IX with start of fire #1
10ED  DD360001  LD      (IX+#00),#01    ; set fire active
10F1  DD360358  LD      (IX+#03),#58    ; set fire X position
10F5  DD360E58  LD      (IX+#0E),#58    ; set fire X position #2
10F9  DD360580  LD      (IX+#05),#80    ; set fire Y position
10FD  DD360F80  LD      (IX+#0F),#80    ; set fire Y position #2

; set up 2nd fireball

1101  DD362001  LD      (IX+#20),#01    ; set fire active
1105  DD3623EB  LD      (IX+#23),#EB    ; set fire X position
1109  DD362EEB  LD      (IX+#2E),#EB    ; set fire X position
110D  DD362560  LD      (IX+#25),#60    ; set fire Y position
1111  DD362F60  LD      (IX+#2F),#60    ; set fire Y position
1105  DD3623B3  LD      (IX+#23),#B3    ; set fire X adjusted position
1109  DD362EB3  LD      (IX+#2E),#B3    ; set fire X adjusted position
110D  DD3625A0  LD      (IX+#25),#A0    ; set fire Y adjusted position
1111  DD362FA0  LD      (IX+#2F),#A0    ; set fire Y adjusted position


1115  117069    LD      DE,#6970        ; destination #6970 (sprites used at top and bottom of elevators)
1118  212111    LD      HL,#1121        ; source data at table below
111B  011000    LD      BC,#0010        ; byte counter at #10
111E  EDB0      LDIR                    ; copy
1120  C9        RET                     ; return

; data used above for top and bottom of elevator shafts

1121  37 45 0F 60                       ; X = #37, color = #45, sprite = #F, Y = #60
1125  37 45 8F F7
1129  77 45 0F 60
112D  77 45 8F F7

; arrive here when rivets starts from #0FCC

1131  21F03D    LD      HL,#3DF0        ; load HL with start of table data
1134  110764    LD      DE,#6407        ; load DE with destination ?
1137  011C05    LD      BC,#051C        ; set counters

113A  CD2A12    CALL    #122A           ; copy fire location data to screen?

113D  21143E    LD      HL,#3E14        ; load HL with start of table data for hammer locations
1140  CDA611    CALL    #11A6           ; draw the hammers

1143  21543E    LD      HL,#3E54        ; load HL with start of bonus items for rivets
1146  110C6A    LD      DE,#6A0C        ; set destination sprite address
1149  010C00    LD      BC,#000C        ; set counter to #C bytes to copy
114C  EDB0      LDIR                    ; draw purse, umbrella, hat to screen

114E  218211    LD      HL,#1182        ; load HL with start of data table
1151  11A364    LD      DE,#64A3        ; load DE with destination ?
1154  011E02    LD      BC,#021E        ; set counters
1157  CDEC11    CALL    #11EC           ; copy

; draws black squares next to kong???

115A  217E11    LD      HL,#117E        ; load HL with start of data table
115D  11A764    LD      DE,#64A7        ; set destination sprites
```

```
1160  011C02    LD      BC,#021C        ; set counters B := 2, C := #1C
1163  CD2A12    CALL    #122A           ; copy

1166  DD21A064  LD      IX,#64A0        ; load IX with address of black square sprite start
116A  DD360001  LD      (IX+#00),#01    ; store 1 into #64A0 = turn on first sprite
116E  DD362001  LD      (IX+#20),#01    ; store 1 into #64C0 = turn on second sprite

1172  215069    LD      HL,#6950        ; load HL with ???
1175  0602      LD      B,#02           ; set counter to 2
1177  112000    LD      DE,#0020        ; set offset to #20
117A  CDD311    CALL    #11D3           ; draw items ???

117D  C9        RET                     ; return

; data used above for black space next to kong

117E  3F 0C 08 08                       ; sprite code #3F (invisible square), color = #0C (black), size = 8x8 ???
1182  73 50 8D 50                       ; 1st is at #73,#50 and the 2nd is at #8D,#50

; called from #102B and #1093

1186  21A211    LD      HL,#11A2        ; load HL with start of data table
1189  110765    LD      DE,#6507        ; load DE with destination
118C  010C0A    LD      BC,#0A0C        ; set counters
118F  CD2A12    CALL    #122A           ; copy

1192  DD210065  LD      IX,#6500        ; load IX with ???
1196  218069    LD      HL,#6980        ; load HL with sprite start (???)
1199  060A      LD      B,#0A           ; B := #A
119B  111000    LD      DE,#0010        ; load DE with offset
119E  CDD311    CALL    #11D3           ; copy

11A1  C9        RET                     ; return

; data table used above

11A2  3B 00 02 02

; called from 3 locations with HL preloaded with address of locations to draw to

11A6  118366    LD      DE,#6683        ; load DE with sprite destination address ???
11A9  010E02    LD      BC,#020E        ; B := 2 for the 2 hammers.  C := #E for ???
11AC  CDEC11    CALL    #11EC           ;

11AF  21083E    LD      HL,#3E08        ; set source
11B2  118766    LD      DE,#6687        ; set destination
11B5  010C02    LD      BC,#020C        ; set counters
11B8  CD2A12    CALL    #122A           ; copy table data from #3E08 into #6687 with counters #02 and #0C

11BB  DD218066  LD      IX,#6680        ; load IX with start of hammer array
11BF  DD360001  LD      (IX+#00),#01    ; set hammer 1 active
11C3  DD361001  LD      (IX+#10),#01    ; set hammer 2 active
11C7  21186A    LD      HL,#6A18        ; set destination for hammer sprites ?
11CA  0602      LD      B,#02           ; set counter to 2
11CC  111000    LD      DE,#0010        ; set offset to #10
11CF  CDD311    CALL    #11D3           ; draw hammers

11D2  C9        RET                     ; return

; subroutine uses HL, DE, IX
; B used for loop counter (how many times to loop before returning)
; DE used as an offset for the next set of items to copy
; used to draw hammers initially on each level that has them ?
;

11D3  DD7E03    LD      A,(IX+#03)      ; Load A with item's X position
11D6  77        LD      (HL),A          ; store into HL = sprite X position
11D7  2C        INC     L               ; next HL
11D8  DD7E07    LD      A,(IX+#07)      ; load A with item's sprite value
11DB  77        LD      (HL),A          ; store into sprite value
11DC  2C        INC     L               ; next HL
11DD  DD7E08    LD      A,(IX+#08)      ; load A with item color
11E0  77        LD      (HL),A          ; store into sprite color
11E1  2C        INC     L               ; next HL
11E2  DD7E05    LD      A,(IX+#05)      ; load A with Y position
11E5  77        LD      (HL),A          ; store into sprite Y position
11E6  2C        INC     L               ; next HL
11E7  DD19      ADD     IX,DE           ; add offset into IX for next set of data
11E9  10E8      DJNZ    #11D3           ; loop until B == 0

11EB  C9        RET                     ; return

; draw umbrella, etc to screen on rivets level?
; also used on elevators, called from #10C0

11EC  7E        LD      A,(HL)          ; load A with first table data
11ED  12        LD      (DE),A          ; store into (DE) = sprite ?
11EE  23        INC     HL              ; next table data
11EF  1C        INC     E
11F0  1C        INC     E               ; next sprite
11F1  7E        LD      A,(HL)          ; load next data
11F2  12        LD      (DE),A          ; store
11F3  23        INC     HL              ; next data
11F4  7B        LD      A,E             ; load A with E
11F5  81        ADD     A,C             ; add C (offset for next sprite);  EG #0E
```

```
11F6  5F        LD      E,A             ; store into E
11F7  10F3      DJNZ    #11EC           ; loop until done

11F9  C9        RET                     ; return

;
; called from #104C for conveyors
; called from #0FF2 for girders
; draw stuff in conveyors and girders
; HL is preloaded with #3DFA for conveyors and #3DF4 for girders = table data for intial fire location
; 3DF4:  27 70 01 E0 00 00      ; initial data for fires on girders ?
; 3DFA:  7F 40 01 78 02 00      ; initial data for conveyors to release a fire ?
;

11FA  DD21A066  LD      IX,#66A0        ; load IX with sprite memory array for fire above the barrel
11FE  11286A    LD      DE,#6A28        ; load DE with hardware sprite memory for same fire
1201  DD360001  LD      (IX+#00),#01    ; enable the sprite
1205  7E        LD      A,(HL)          ; load A with table data
1206  DD7703    LD      (IX+#03),A      ; store into sprite X position
1209  12        LD      (DE),A          ; store into sprite X position
120A  1C        INC     E               ; next DE
120B  23        INC     HL              ; next HL
120C  7E        LD      A,(HL)          ; load A with table data
120D  DD7707    LD      (IX+#07),A      ; store into sprite graphic
1210  12        LD      (DE),A          ; store into sprite graphic
1211  1C        INC     E               ; next DE
1212  23        INC     HL              ; next HL
1213  7E        LD      A,(HL)          ; load A with table data
1214  DD7708    LD      (IX+#08),A      ; store into sprite color
1217  12        LD      (DE),A          ; store into sprite color
1218  1C        INC     E               ; next DE
1219  23        INC     HL              ; next HL
121A  7E        LD      A,(HL)          ; load A with table data
121B  DD7705    LD      (IX+#05),A      ; store into sprite Y position
121E  12        LD      (DE),A          ; store into sprite Y position
121F  23        INC     HL              ; next HL
1220  7E        LD      A,(HL)          ; load A with table data
1221  DD7709    LD      (IX+#09),A      ; store into size (width?) ???
1224  23        INC     HL              ; next HL
1225  7E        LD      A,(HL)          ; load A with table data
1226  DD770A    LD      (IX+#0A),A      ; store into size? (height?) ??
1229  C9        RET                     ; return


; Subroutine from #10CC
; Copies Data from Table in HL into the Destination at DE in chunks of 4
; B is used for the second loop variable
; C is used to specify the difference between the tables, assumed to be 4 or 5 or 0 ?
; used for example to place the hammers ???

122A  E5        PUSH    HL              ; Save HL
122B  C5        PUSH    BC              ; Save BC
122C  0604      LD      B,#04           ; For B = 1 to 4

122E  7E        LD      A,(HL)          ; load A with the Contents of HL table data
122F  12        LD      (DE),A          ; store data into address DE
1230  23        INC     HL              ; next table data
1231  1C        INC     E               ; next destination
1232  10FA      DJNZ    #122E           ; Next B

1234  C1        POP     BC              ; Restore BC - For B = 1 to Initial B value
1235  E1        POP     HL              ; Restore HL
1236  7B        LD      A,E             ; A := E
1237  81        ADD     A,C             ; add C
1238  5F        LD      E,A             ; store result into E
1239  10EF      DJNZ    #122A           ; Loop again if not zero

123B  C9        RET                     ; Return

; set initial mario sprite position and draw remaining lives and level

123C  DF        RST     #18             ; count down WaitTimerMSB and only continue when 0
123D  3A2762    LD      A,(#6227)       ; load a with screen number
1240  fe03      CP      #03             ; is this the elevators?
1242  0116E0    LD      BC,#e016        ; B := #E0, C := #16.  used for X,Y coordinates
1245  cA4B12    JP      Z,#124B         ; if elevators skip next step

1248  013FF0    LD      BC,#F03F        ; else load alternate coordinates for elevators

124B  DD210062  LD      IX,#6200        ; set IX to mario sprite array
124F  214C69    LD      HL,#694C        ; load HL with address for mario sprite X value
1252  DD360001  LD      (IX+#00),#01    ; turn on sprite
1256  DD7103    LD      (IX+#03),C      ; store X position
1259  71        LD      (HL),C          ; store X position
125A  2C        INC     L               ; next
125B  DD360780  LD      (IX+#07),#80    ; store sprite graphic
125F  3680      LD      (HL),#80        ; store sprite graphic
1261  2C        INC     L               ; next
1262  DD360802  LD      (IX+#08),#02    ; store sprite color
1266  3602      LD      (HL),#02        ; store sprite color
1268  2C        INC     L               ; next
1269  DD7005    LD      (IX+#05),B      ; store Y position
126C  70        LD      (HL),B          ; store Y position
126D  DD360F01  LD      (IX+#0F),#01    ; turn this on (???)
1271  210A60    LD      HL,GameMode2        ; load HL with game mode2 address
```

```
1274  34        INC     (HL)            ; increase game mode2 = start game
1275  110106    LD      DE,#0601        ; set task #6, parameter 1 to draw lives-1 and level
1278  CD9F30    CALL    #309F           ; insert task
127B  C9        RET                     ; return

; jump here from #0701 when GameMode2 == #D
; mario died ?

127C  CDBD1D    CALL    #1DBD           ; check for bonus items and jumping scores, rivets
127F  3A9D63    LD      A,(#639D)       ; load A with this normally 0.  1 while mario dying, 2 when dead
1282  EF        RST     #28             ; jump based on A

1283  8B 12                             ; #128B         ; 0 normal
1285  AC 12                             ; #12AC         ; 1 mario dying
1287  DE 12                             ; #12DE         ; 2 mario dead
1289  00 00                             ; unused ?

128B  DF        RST     #18             ; count down WaitTimerMSB and only continue when 0
128C  214D69    LD      HL,#694D        ; load HL with mario sprite value
128F  3EF0      LD      A,#F0           ; A := #F0
1291  CB16      RL      (HL)            ; rotate left (HL)
1293  1F        RRA                     ; rotate right that carry bit into A
1294  77        LD      (HL),A          ; store result into mario sprite
1295  219D63    LD      HL,#639D        ; load HL with mario death indicator
1298  34        INC     (HL)            ; increase.  mario is now dying
1299  3E0D      LD      A,#0D           ; A := #D (13 decimal)
129B  329E63    LD      (#639E),A       ; store into counter for number of times to rotate mario (?)
129E  3E08      LD      A,#08           ; load A with 8 frames of delay
12A0  320960    LD      (WaitTimerMSB),A    ; store into timer for sound delay
12A3  CDBD30    CALL    #30BD           ; clear sprites ?
12A6  3E03      LD      A,#03           ; load A with duration of sound
12A8  328860    LD      (#6088),A       ; play death sound
12AB  C9        RET                     ; return

; arrive here when mario dies
; animates mario

12AC  DF        RST     #18             ; count down WaitTimerMSB and only continue when 0
12AD  3E08      LD      A,#08           ; load A with 8 frames of delay
12AF  320960    LD      (WaitTimerMSB),A    ; store into timer for sound delays
12B2  219E63    LD      HL,#639E        ; load counter
12B5  35        DEC     (HL)            ; decrease.  are we done ?
12B6  CACB12    JP      Z,#12CB         ; yes, skip ahead

12B9  214D69    LD      HL,#694D        ; load HL with mario sprite value
12BC  7E        LD      A,(HL)          ; get the value
12BD  1F        RRA                     ; roll right = div 2
12BE  3E02      LD      A,#02           ; load A with 2
12C0  1F        RRA                     ; roll right , A now has 1
12C1  47        LD      B,A             ; copy to B
12C2  AE        XOR     (HL)            ; toggle HL rightmost bit
12C3  77        LD      (HL),A          ; save new sprite value
12C4  2C        INC     L               ; next HL
12C5  78        LD      A,B             ; load A with B
12C6  E680      AND     #80             ; apply mask
12C8  AE        XOR     (HL)            ; toggle HL
12C9  77        LD      (HL),A          ; save new value
12CA  C9        RET                     ; return

; mario done rotating after death

12CB  214D69    LD      HL,#694D        ; load HL with mario sprite value
12CE  3EF4      LD      A,#F4           ; load A with #F4
12D0  CB16      RL      (HL)            ; rotate left HL (goes from F8 to F0)
12D2  1F        RRA                     ; roll right A.  A becomes FA
12D3  77        LD      (HL),A          ; store into sprite value (mario dead)
12D4  219D63    LD      HL,#639D        ; load HL with death indicator
12D7  34        INC     (HL)            ; increase.  mario now dead
12D8  3E80      LD      A,#80           ; load A with delay of 80
12DA  320960    LD      (WaitTimerMSB),A    ; store into sound delay counter
12DD  C9        RET                     ; return

; mario is completely dead

12DE  DF        RST     #18             ; count down WaitTimerMSB and only continue when 0
12DF  CDDB30    CALL    #30DB           ; clear mario and elevator sprites from screen
12E2  210A60    LD      HL,GameMode2        ; set HL to game mode2
12E5  3A0E60    LD      A,(PlayerTurnB)       ; load A with current player
12E8  A7        AND     A               ; is this player 1 ?
12E9  CAED12    JP      Z,#12ED         ; yes, skip next step

12EC  34        INC     (HL)            ; increase game mode

12ED  34        INC     (HL)            ; increase game mode
12EE  2B        DEC     HL              ; load HL with WaitTimerMSB
12EF  3601      LD      (HL),#01        ; store 1 into timer
12F1  C9        RET                     ; return

; jump here from #0701
; player 1 died
; clear sounds, decrease life, check for and handle game over

12F2  CD1C01    CALL    #011C           ; clear all sounds
12F5  AF        XOR     A               ; A := 0
12F6  322C62    LD      (#622C),A       ; store into game start flag
```

```
12F9  212862    LD      HL,#6228        ; load HL with address for number of lives remaining
12FC  35        DEC     (HL)            ; one less life
12FD  7E        LD      A,(HL)          ; load A with number of lives left
12FE  114060    LD      DE,P1NumLives       ; set destination address
1301  010800    LD      BC,#0008        ; set counter
1304  EDB0      LDIR                    ; copy (#6228) to (#6230) into (P1NumLives) to (P2NumLives).  copies data from player
area to storage area for player 1
1306  A7        AND     A               ; number of lives == 0 ?
1307  C23413    JP      NZ,#1334        ; no, skip ahead

; game over for this player [?]

130A  3E01      LD      A,#01           ; A := 1
130C  21B260    LD      HL,#60B2        ; load HL with player 1 score address
130F  CDCA13    CALL    #13CA           ; check for high score entry ???
1312  21D476    LD      HL,#76D4        ; load HL with screen VRAM address ???
1315  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
1318  A7        AND     A               ; 1 player game?
1319  2807      JR      Z,#1322         ; yes, skip next 3 steps

131B  110203    LD      DE,#0302        ; load task data for text #2 "PLAYER <I>"
131E  CD9F30    CALL    #309F           ; insert task to draw text
1321  2B        DEC     HL              ; HL := #76D3

1322  CD2618    CALL    #1826           ; clear an area of the screen
1325  110003    LD      DE,#0300        ; load task data for text #0 "GAME OVER"
1328  CD9F30    CALL    #309F           ; insert task to draw text
132B  210960    LD      HL,WaitTimerMSB ; load HL with timer
132E  36C0      LD      (HL),#C0        ; set timer to #C0
1330  23        INC     HL              ; HL := GameMode2
1331  3610      LD      (HL),#10        ; set game mode2 to #10
1333  C9        RET                     ; return

1334  0E08      LD      C,#08           ; C := 8
1336  3A0F60    LD      A,(TwoPlayerGame)       ; load A with number of players
1339  A7        AND     A               ; 1 player game?
133A  CA3F13    JP      Z,#133F         ; yes, skip next step

133D  0E17      LD      C,#17           ; C := #17

133F  79        LD      A,C             ; A := C
1340  320A60    LD      (GameMode2),A       ; store into game mode2
1343  C9        RET                     ; return

; arrive from #0701 when GameMode2 == #F
; clear sounds, clear game start flag, draw game over if needed, set game mode2 accordingly

1344  CD1C01    CALL    #011C           ; clear all sounds
1347  AF        XOR     A               ; A := 0
1348  322C62    LD      (#622C),A       ; store into game start flag
134B  212862    LD      HL,#6228        ; load HL with number of lives remaining
134E  35        DEC     (HL)            ; decrease
134F  7E        LD      A,(HL)          ; load A with the number of lives remaining
1350  114860    LD      DE,P2NumLives       ; load DE with destination address
1353  010800    LD      BC,#0008        ; set counter to 8
1356  EDB0      LDIR                    ; copy
1358  A7        AND     A               ; any lives left?
1359  C27F13    JP      NZ,#137F        ; yes, skip ahead

; game over

135C  3E03      LD      A,#03           ; A := 3
135E  21B560    LD      HL,#60B5        ; load HL with player 2 score address
1361  CDCA13    CALL    #13CA           ; check for high score entry ???
1364  110303    LD      DE,#0303        ; load task data for text #3 "PLAYER <II>"
1367  CD9F30    CALL    #309F           ; insert task to draw text
136A  110003    LD      DE,#0300        ; load task data for text #0 "GAME OVER"
136D  CD9F30    CALL    #309F           ; insert task to draw text
1370  21D376    LD      HL,#76D3        ; load HL with screen address ???
1373  CD2618    CALL    #1826           ; clear an area of the screen
1376  210960    LD      HL,WaitTimerMSB ; load HL with timer
1379  36C0      LD      (HL),#C0        ; set timer to #C0
137B  23        INC     HL              ; HL := GameMode2
137C  3611      LD      (HL),#11        ; set game mode2 to #11
137E  C9        RET                     ; return

137F  0E17      LD      C,#17           ; C := #17
1381  3A4060    LD      A,(P1NumLives)      ; load A with number of lives left for player 1
1384  A7        AND     A               ; player 1 has lives remaining?
1385  C28A13    JP      NZ,#138A        ; yes, skip next step

1388  0E08      LD      C,#08           ; C := 8

138A  79        LD      A,C             ; A := C
138B  320A60    LD      (GameMode2),A       ; store A into game mode2
138E  C9        RET                     ; return

; code to draw screen and random ladders for the rivets screen
; #0BC7 : draw girders screen, #0919 : draw pies screen, #135B : draw elevators screen, #1344 : draw rivets screen

1344  3A2762    LD      A,(#6227)       ; load A with screen #
1347  FE04      CP      #04             ; is this the rivets?
1349  C21909    JP      NZ,#0919        ; no, jump to additional code
134C  D5        PUSH    DE              ; save DE to the stack for later
134D  CD1A0C    CALL    #0C1A           ; create the random ladder definitions
```

```
1350  11006B    LD      DE,#6B00          ; load DE with pointer to output data structure containing ladder definitions
1353  CDA70D    CALL    #0DA7             ; draw the random ladders
1356  D1        POP     DE                ; restore DE from the stack
1357  CDA70D    CALL    #0DA7             ; draw the screen
135A  C9        RET                       ; return

; code to draw screen and random ladders for the elevators screen
; #0BC7 : draw girders screen, #0919 : draw pies screen, #135B : draw elevators screen, #1344 : draw rivets screen

135B  CDA70D    CALL    #0DA7             ; draw the screen
135E  DD21053C  LD      IX,#3C05          ; load IX with pointer to ladder data structure for elevators screen
1362  21006B    LD      HL,#6B00          ; load HL with pointer to output data structure to build ladder definitions
1365  CD210C    CALL    #0C21             ; create the random ladder definitions
1368  11006B    LD      DE,#6B00          ; load DE with pointer to output data structure containing ladder definitions
136B  CDA70D    CALL    #0DA7             ; draw the random ladders
136E  C9        RET                       ; jump to continue

; code to draw level and sub level to screen

136F  32C374    LD      (#74C3),A         ; draw level to screen (high byte)
1372  3E2C      LD      A,#2C             ; load A with #02
1374  328374    LD      (#7483),A         ; draw "-"
1377  3A2962    LD      A,(#6229)         ; load A with level number
137A  FE00      CP      #00               ; is the level number equal to 0?
137C  2804      JR      Z,#1382           ; yes, skip next lines
137E  3A2E62    LD      A,(#622E)         ; load A with sub level indicator
1381  3C        INC     A                 ; increment A
1382  326374    LD      (#7463),A         ; draw sub level to screen
1385  C9        RET                       ; return

; code to increase the sub level

1386  3608      LD      (HL),#08          ; set game mode2 to 8 (original code from #17B3)
1388  212E62    LD      HL,#622E          ; load HL with address of sub level indicator
138B  34        INC     (HL)              ; increase the sub level indicator
138C  C9        RET                       ; return
138D  00        NOP                       ; no operation
138E  00        NOP                       ; no operation

; arrive from #0701 when GameMode2 == #10
; when 2 player game has ended

138F  DF        RST     #18               ; count down timer and only continue here if zero, else RET
1390  0E17      LD      C,#17             ; C := #17
1392  3A4860    LD      A,(P2NumLives)        ; load A with number of lives for player 2

1395  34        INC     (HL)              ; increase timer ??? [EG HL = WaitTimerMSB]
1396  A7        AND     A                 ; player has lives remaining ?
1397  C29C13    JP      NZ,#139C          ; yes, skip next step

139A  0E14      LD      C,#14             ; else C := #14

139C  79        LD      A,C               ; A := C
139D  320A60    LD      (GameMode2),A         ; store into game mode2
13A0  C9        RET                       ; return


; arrive from #0701 when GameMode2 == #11

13A1  DF        RST     #18               ; count down timer and only continue here if zero, else RET
13A2  0E17      LD      C,#17             ; C := #17
13A4  3A4060    LD      A,(P1NumLives)        ; load A with number of lives remaining for player1
13A7  C39513    JP      #1395             ; jump back, rest of this sub is above

; arrive from #0701 when GameMode2 == 12
; flip screen if needed, reset game mode2 to zero, set player 2

13AA  3A2660    LD      A,(UprightCab)        ; load A with upright/cocktail
13AD  32827D    LD      (REG_FLIPSCREEN),A        ; store into hardware screen flip
13B0  AF        XOR     A                 ; A := 0
13B1  320A60    LD      (GameMode2),A         ; set game mode2 to 0
13B4  210101    LD      HL,#0101          ; HL := #101
13B7  220D60    LD      (PlayerTurnA),HL      ; store 1 into PlayerTurnA (set player2) and PlayerTurnB (set player2)
13BA  C9        RET                       ; return

; arrive from #0701 when GameMode2 == 13
; set player 1, reset game mode2 to zero, set screen flip to not flipped

13BB  AF        XOR     A                 ; A := 0
13BC  320D60    LD      (PlayerTurnA),A       ; set for player 1
13BF  320E60    LD      (PlayerTurnB),A       ; store into current player number 1
13C2  320A60    LD      (GameMode2),A         ; set game mode2 to 0
13C5  3C        INC     A                 ; A := 1
13C6  32827D    LD      (REG_FLIPSCREEN),A        ; store into screen flip for no flipping
13C9  C9        RET                       ; return

; code to reset the sub level

13A1  32206A    LD      (#6A20),A         ; store A into heart sprite X position (original code from #1939)
13A4  3E00      LD      A,#00             ; load A with #00
13A6  322E62    LD      (622E),A          ; reset the sub level indicator
13A9  C9        RET                       ; return

13AA  00000000000000000000000000000000  ; no operations - free space
13BA  00000000000000000000000000000000  ; no operations - free space
```

```
; causes the player's score to percolate up the high score list
; [but it is never read from ???]

; called from #1361, HL is preloaded with #60B5 = player 2 score address, A is preloaded with 3
; called from #130F, HL is preloaded with #60B2 = player 1 score address, A is preloaded with 1

; this sub copies player score into #61C7-#61C9
; then it breaks the score into component digits and stores them into #61B1 through #61B6
; then it sets #61B7 through #61C4 to #10 (???)
;

13CA  11C661    LD      DE,#61C6        ; load DE with address for ???
13CD  12        LD      (DE),A          ; store A into it
13CE  CF        RST     #8              ; continue if there are credits or the game is being played, else RET

13CF  13        INC     DE              ; DE := #61C7
13D0  010300    LD      BC,#0003        ; set counter to 3
13D3  EDB0      LDIR                    ; copy players score into this area
13D5  0603      LD      B,#03           ; for B = 1 to 3
13D7  21B161    LD      HL,#61B1        ; load HL with ???

13DA  1B        DEC     DE              ; count down DE.  first time it has #61C9 after the DEC
13DB  1A        LD      A,(DE)          ; load A with this
13DC  0F        RRCA
13DD  0F        RRCA
13DE  0F        RRCA
13DF  0F        RRCA                    ; rotate right 4 times.  this transposes the 4 low and 4 high bits of the byte
13E0  E60F      AND     #0F             ; mask bits, now between 0 and #F.  this will give the thousands of the score on the
2nd loop.
13E2  77        LD      (HL),A          ; store into (HL) ???
13E3  23        INC     HL              ; next
13E4  1A        LD      A,(DE)          ; load A with this
13E5  E60F      AND     #0F             ; mask bits.  this will give the hundreds of the score on the 2nd loop
13E7  77        LD      (HL),A          ; store into (HL)
13E8  23        INC     HL              ; next
13E9  10EF      DJNZ    #13DA           ; next B

; sets #61B7 through #61C4 to #10 (???)

13EB  060E      LD      B,#0E           ; for B = 1 to #E
13ED  3610      LD      (HL),#10        ; store #10 into memory at (HL)
13EF  23        INC     HL              ; next HL
13F0  10FB      DJNZ    #13ED           ; next B
13EB  3610      LD      (HL),#10        ; draw a space at this position
13ED  23        INC     HL              ; next position in high score row
13EE  C3C93C    JP      #3CC9           ; jump to additional code to display the sub-level in the high score list
13F1  00        NOP                     ; no operation

;
13F2  363F      LD      (HL),#3F        ; store #3F into #61C5 = end code ?

13F4  0605      LD      B,#05           ; for B = 1 to 5.  Do for each high score in top 5
13F6  21A561    LD      HL,#61A5        ; load HL with lowest high score address
13F9  11C761    LD      DE,#61C7        ; load DE with copy of player score

13FC  1A        LD      A,(DE)          ; load A with a digit of player's score
13FD  96        SUB     (HL)            ; subtract next lowest high score
13FE  23        INC     HL              ; next
13FF  13        INC     DE              ; next
1400  1A        LD      A,(DE)          ; load A with next digit of player's score
1401  9E        SBC     A,(HL)          ; subtract with carry next lowest high score
1402  23        INC     HL              ; next
1403  13        INC     DE              ; next
1404  1A        LD      A,(DE)          ; load A with next digit of player's score
1405  9E        SBC     A,(HL)          ; subtract with carry next lowest high score
1406  D8        RET     C               ; if player has not made this high score, return

; player has made a high score for entry in top 5

1407  C5        PUSH    BC              ; else save BC

1408  0619      LD      B,#19           ; for B = 1 to #19

; exchange the values in (HL) and (DE) for #19 bytes
; this causes the high score to percolate up the high score list

140A  4E        LD      C,(HL)          ; C := (HL)
140B  1A        LD      A,(DE)          ; A := (DE)
140C  77        LD      (HL),A          ; (HL) := A
140D  79        LD      A,C             ; A := C
140E  12        LD      (DE),A          ; (DE) := A
140F  2B        DEC     HL              ; next HL
1410  1B        DEC     DE              ; next DE
1411  10F7      DJNZ    #140A           ; Next B

1413  01F5FF    LD      BC,#FFF5        ; load BC with -#A
1416  09        ADD     HL,BC           ; add to HL.  HL now has #A less than before
1417  EB        EX      DE,HL           ; DE <> HL
1418  09        ADD     HL,BC           ; add to HL, now has #A less than before
1419  EB        EX      DE,HL           ; DE <> HL
141A  C1        POP     BC              ; restore BC
141B  10DF      DJNZ    #13FC           ; Next B

141D  C9        RET                     ; return
```

```
; jump here from #0701 when GameMode2 == #14 (game is over)
; draw credits on screen, clears screen and sprites, checks for high score, flips screen if necessary

141E  CD1606    CALL    #0616           ; draw credits on screen
1421  DF        RST     #18             ; count down timer and only continue here if zero, else RET

1422  CD7408    CALL    #0874           ; clears the screen and sprites
1425  3E00      LD      A,#00           ; A := 0
1427  320E60    LD      (PlayerTurnB),A       ; set player number 1
142A  320D60    LD      (PlayerTurnA),A       ; set player1
142D  211C61    LD      HL,#611C        ; load HL with high score entry indicator
1430  112200    LD      DE,#0022        ; offset to add is #22
1433  0605      LD      B,#05           ; for B = 1 to 5
1435  3E01      LD      A,#01           ; A := 1 = code for a new high score for player 1

1437  BE        CP      (HL)            ; compare (HL) to 1 .  equal ?
1438  CA5914    JP      Z,#1459         ; yes, jump to high score entry for player 1

143B  19        ADD     HL,DE           ; else next HL
143C  10F9      DJNZ    #1437           ; next B

143E  211C61    LD      HL,#611C        ; load HL with high score entry indicator
1441  0605      LD      B,#05           ; For B = 1 to 5
1443  3E03      LD      A,#03           ; A := 3 = code for a new high score for player 2

1445  BE        CP      (HL)            ; compare.  same?
1446  CA4F14    JP      Z,#144F         ; yes, skip ahead and being high score entry for pl2

1449  19        ADD     HL,DE           ; add offset for next
144A  10F9      DJNZ    #1445           ; Next B

144C  C37514    JP      #1475           ; skip ahead, no high score was achieved

; high score achieved ?

144F  3E01      LD      A,#01           ; A := 1
1451  320E60    LD      (PlayerTurnB),A       ; set player #2
1454  320D60    LD      (PlayerTurnA),A       ; set player2
1457  3E00      LD      A,#00           ; A := 0

1459  212660    LD      HL,UprightCab        ; load HL with address for upright/cocktail
145C  B6        OR      (HL)            ; mix with A
145D  32827D    LD      (REG_FLIPSCREEN),A       ; store A into screen flip setting
1460  3E00      LD      A,#00           ; A := 0
1462  320960    LD      (WaitTimerMSB),A    ; clear timer
1465  210A60    LD      HL,GameMode2        ; load HL with game mode2 address
1468  34        INC     (HL)            ; increase game mode2 to #15
1469  110D03    LD      DE,#030D        ; load task data for text #D "NAME REGISTRATION"
146C  060C      LD      B,#0C           ; set counter for #0C items (12 decimal)

146E  CD9F30    CALL    #309F           ; insert task to draw text
1471  13        INC     DE              ; next text set
1472  10FA      DJNZ    #146E           ; next B

1474  C9        RET                     ; return

; jump here from #144C

1475  3E01      LD      A,#01           ; A := 1
1477  32827D    LD      (REG_FLIPSCREEN),A       ; set screen flip setting
147A  320560    LD      (GameMode1),A   ; store into game mode1
147D  320760    LD      (NoCredits),A   ; set indicator for no credits
1480  3E00      LD      A,#00           ; A := 0
1482  320A60    LD      (GameMode2),A       ; reset game mode2 to 0.  game is now totally over.
1485  C9        RET                     ; return


; jump from #0701 when GameMode2 == #15
; game is over - high score entry


1486  CD1606    CALL    #0616           ; draw credits on screen
1489  210960    LD      HL,WaitTimerMSB ; load HL with timer
148C  7E        LD      A,(HL)          ; load A with timer value
148D  A7        AND     A               ; == 0 ?
148E  C2DC14    JP      NZ,#14DC        ; no, skip ahead

1491  32867D    LD      (REG_PALETTE_A),A       ; set palette bank selector
1494  32877D    LD      (REG_PALETTE_B),A       ; set palette bank selector
1497  3601      LD      (HL),#01        ; set timer to 1
1499  213060    LD      HL,HSCursorDelay      ; load HL with HSCursorDelay
149C  360A      LD      (HL),#0A
149E  23        INC     HL              ; HL := HSBlinkToggle
149F  3600      LD      (HL),#00
14A1  23        INC     HL              ; HL := HSBlinkTimer
14A2  3610      LD      (HL),#10
14A4  23        INC     HL              ; HL := HSRegiTime
14A5  361E      LD      (HL),#1E
14A7  23        INC     HL              ; HL := HSTimer
14A8  363E      LD      (HL),#3E        ; set outer loop timer
14AA  23        INC     HL              ; HL := HSCursorPos
14AB  3600      LD      (HL),#00        ; set high score digit selected
14AD  21E875    LD      HL,#75E8        ; load HL with screen position for first player initial
14B0  223660    LD      (HSInitialPos),HL      ; save into this indicator
```

```
14B3  211C61    LD      HL,#611C        ; load HL with address of high score indicator
14B6  3A0E60    LD      A,(PlayerTurnB)       ; load A with current player number
14B9  07        RLCA                    ; rotate left
14BA  3C        INC     A               ; increase
14BB  4F        LD      C,A             ; copy to C.  C now has 1 for player 1, 3 for player 2
14BC  112200    LD      DE,#0022        ; load DE with offset
14BF  0604      LD      B,#04           ; for B = 1 to 4

14C1  7E        LD      A,(HL)          ; load A with high score indicator
14C2  B9        CP      C               ; == current player number ?
14C3  CAC914    JP      Z,#14C9         ; yes, skip next 2 steps - this is the one

14C6  19        ADD     HL,DE           ; add offset for next HL
14C7  10F8      DJNZ    #14C1           ; Next B

14C9  223860    LD      (Unk6038),HL      ; store HL into Unk6038
14CC  11F3FF    LD      DE,#FFF3        ; load DE with offset of  #13
14CC  11F9FF    LD      DE,#FFF9        ; load DE with adapted offset of -8 (decimal) to move NAME in high score list
14CF  19        ADD     HL,DE           ; add offset
14D0  223A60    LD      (#603A),HL      ; store result into ???
14D3  0600      LD      B,#00           ; B := 0
14D5  3A3560    LD      A,(HSCursorPos)       ; load A with high score entry digit selected
14D8  4F        LD      C,A             ; copy to C
14D9  CDFA15    CALL    #15FA           ; ???

14DC  213460    LD      HL,HSTimer      ; load HL with outer loop timer
14DF  35        DEC     (HL)            ; count down timer.  at zero?
14E0  C2FC14    JP      NZ,#14FC        ; no, skip ahead

14E3  363E      LD      (HL),#3E        ; reset outer loop timer
14E5  2B        DEC     HL              ; HL := HSRegiTime
14E6  35        DEC     (HL)            ; decrease.  at zero?
14E7  CAC615    JP      Z,#15C6         ; yes, skip ahead to handle

14EA  7E        LD      A,(HL)          ; else load A with time remaining
14EB  06FF      LD      B,#FF           ; B := #FF.  used to count 10's

14ED  04        INC     B               ; increase B
14EE  D60A      SUB     #0A             ; subtract #0A (10 decimal).  gone under?
14F0  D2ED14    JP      NC,#14ED        ; no, loop again.  B will have number of 10's

14F3  C60A      ADD     A,#0A           ; add #0A to make between 0 and 9
14F5  325275    LD      (#7552),A       ; draw digit to screen
14F8  78        LD      A,B             ; A := B = 10's of time left
14F9  327275    LD      (#7572),A       ; draw digit to screen

14FC  213060    LD      HL,HSCursorDelay      ; load HL with HSCursorDelay
14FF  46        LD      B,(HL)          ; load B with the value
1500  360A      LD      (HL),#0A        ; store #A into it
1502  3A1060    LD      A,(InputState)       ; load A with input
1505  CB7F      BIT     7,A             ; is jump button pressed?
1507  C24615    JP      NZ,#1546        ; yes, skip ahead

150A  E603      AND     #03             ; mask bits.  check for a left or right direction pressed
150C  C21415    JP      NZ,#1514        ; if direction, skip next 3 steps

150F  3C        INC     A               ; else increase A
1510  77        LD      (HL),A          ; store into HSCursorDelay
1511  C38A15    JP      #158A           ; skip ahead

; left or right pressed while in high score entry

1514  05        DEC     B               ; decrease B.  at zero?
1515  CA1D15    JP      Z,#151D         ; yes, skip next 3 steps

1518  78        LD      A,B             ; A := B
1519  77        LD      (HL),A          ; store into ???
151A  C38A15    JP      #158A           ; skip ahead

151D  CB4F      BIT     1,A             ; is direction == left ?
151F  C23915    JP      NZ,#1539        ; yes, skip ahead

1522  3A3560    LD      A,(HSCursorPos)       ; load A with high score entry digit selected
1525  3C        INC     A               ; increase
1526  FE1E      CP      #1E             ; == #1E ?  (have we gone past END ?)
1528  C22D15    JP      NZ,#152D        ; no, skip next step

152B  3E00      LD      A,#00           ; A := 0 [why this way and not XOR A ?] - reset this counter to "A" in the table

152D  323560    LD      (HSCursorPos),A       ; store into high score entry digit selected
1530  4F        LD      C,A             ; C := A
1531  0600      LD      B,#00           ; B := 0
1533  CDFA15    CALL    #15FA           ; ???
1536  C38A15    JP      #158A           ; skip ahead

1539  3A3560    LD      A,(HSCursorPos)       ; load A with high score entry digit selected
153C  D601      SUB     #01             ; decrease [why written this way?  DEC A is standard...]
153E  F22D15    JP      P,#152D         ; if sign positive, loop again

1541  3E1D      LD      A,#1D           ; A := #1D
1543  C32D15    JP      #152D           ; jump back

; jump pressed in high score entry

1546  3A3560    LD      A,(HSCursorPos)       ; load A with high score entry digit selected
```

```
1549  FE1C      CP      #1C             ; == #1C ? = code for backspace ?
154B  CA6D15    JP      Z,#156D         ; yes, skip ahead to handle

154E  FE1D      CP      #1D             ; == #1D ? = code for END
1550  CAC615    JP      Z,#15C6         ; yes, skip ahead to hanlde

1553  2A3660    LD      HL,(HSInitialPos)      ; else load HL with VRAM address of the initial being entered
1556  018875    LD      BC,#7588        ; load BC with screen address
1559  A7        AND     A               ; clear carry flag
155A  ED42      SBC     HL,BC           ; subtract.  equal?
155C  CA8A15    JP      Z,#158A         ; yes, skip ahead

155F  09        ADD     HL,BC           ; else add it back
1560  C611      ADD     A,#11           ; add ascii offset of #11 to A
1562  77        LD      (HL),A          ; write letter to screen
1563  01E0FF    LD      BC,#FFE0        ; load BC with offset for next column
1566  09        ADD     HL,BC           ; set HL to next column

1567  223660    LD      (HSInitialPos),HL      ; store HL back into VRAM address of the initial being entered
156A  C38A15    JP      #158A           ; skip ahead

; backspace selected in high score entry

156D  2A3660    LD      HL,(HSInitialPos)      ; else load HL with VRAM address of the initial being entered
1570  012000    LD      BC,#0020        ; load offset of #20
1573  09        ADD     HL,BC           ; add offset
1574  A7        AND     A               ; clear carry flag
1575  010876    LD      BC,#7608        ; load BC with screen address
1578  ED42      SBC     HL,BC           ; subtract.  equal?
157A  C28615    JP      NZ,#1586        ; no, skip ahead

157D  21E875    LD      HL,#75E8        ; else load HL with other screen address

1580  3E10      LD      A,#10           ; A := #10 = blank code
1582  77        LD      (HL),A          ; clear the screen at this position
1583  C36715    JP      #1567           ; jump back

1586  09        ADD     HL,BC           ; restore HL back to what it was
1587  C38015    JP      #1580           ; jump back

; jump here from #156A and #155C and #1536 and #151A and #1511

158A  213260    LD      HL,HSBlinkTimer ; load HL with HSBlinkTimer
158D  35        DEC     (HL)            ; decrease.  at zero ?
158E  C2F915    JP      NZ,#15F9        ; no, jump to RET. [RET NZ would be faster and more compact]

; Blink the high score in high score table
1591  3A3160    LD      A,(HSBlinkToggle)
1594  A7        AND     A               ; Is HSBlinkToggle zero?
1595  C2B815    JP      NZ,#15B8        ; no, skip ahead

1598  3E01      LD      A,#01           ; A := 1
159A  323160    LD      (HSBlinkToggle),A   ; store into HSBlinkToggle
159D  11BF01    LD      DE,#01BF

15A0  FD2A3860  LD      IY,(Unk6038)      ; load IY with Unk6038
15A4  FD6E04    LD      L,(IY+#04)
15A7  FD6605    LD      H,(IY+#05)
15AA  E5        PUSH    HL
15AB  DDE1      POP     IX              ; load IX with HL
15AD  CD7C05    CALL    #057C           ; ???
15B0  3E10      LD      A,#10           ; A := #10
15B2  323260    LD      (HSBlinkTimer),A    ; store into HSBlinkTimer
15B5  C3F915    JP      #15F9           ; jump to RET [RET would be faster and more compact]

15B8  AF        XOR     A               ; A := 0
15B9  323160    LD      (HSBlinkToggle),A       ; store into HSBlinkToggle
15BC  ED5B3860  LD      DE,(Unk6038)
15C0  13        INC     DE
15C1  13        INC     DE
15C2  13        INC     DE
15C3  C3A015    JP      #15A0           ; jump back

; arrive here from #14E7
; high score entry complete ???

15C6  ED5B3860  LD      DE,(Unk6038)      ; load DE with address of high score entry indicator
15CA  AF        XOR     A               ; A := 0
15CB  12        LD      (DE),A          ; store.  this clears the high score indicator
15CC  210960    LD      HL,WaitTimerMSB ; load HL with timer
15CF  3680      LD      (HL),#80        ; set time to #80
15D1  23        INC     HL              ; HL := GameMode2
15D2  35        DEC     (HL)            ; decrease game mode2
~~15D3  060C      LD      B,#0C           ; for B = 1 to #C (12 decimal)~~
15D3  0605      LD      B,#05           ; load B with adapted offset of 5 (decimal) to move NAME in high score list
15D5  21E875    LD      HL,#75E8        ; load HL with screen vram address
15D8  FD2A3A60  LD      IY,(#603A)      ; load IY with ???
15DC  11E0FF    LD      DE,#FFE0        ; load DE with offset of -#20

15DF  7E        LD      A,(HL)          ; load A with
15E0  FD7700    LD      (IY+#00),A      ; store
15E3  FD23      INC     IY              ; next
15E5  19        ADD     HL,DE           ; add offset
15E6  10F7      DJNZ    #15DF           ; next B
```

```
15E8  0605      LD      B,#05           ; For B = 1 to 5
15EA  111403    LD      DE,#0314        ; load task data for text #14 - start of high score table

15ED  CD9F30    CALL    #309F           ; insert task to draw text
15F0  13        INC     DE              ; next high score
15F1  10FA      DJNZ    #15ED           ; next B

15F3  111A03    LD      DE,#031A        ; load task data for text #1A - "YOUR NAME WAS REGISTERED"
15F6  CD9F30    CALL    #309F           ; insert task to draw text
15F9  C9        RET                     ; return

; sets the sprite to the square selector for intials entry
; called from #14D9 and #1533

15FA  D5        PUSH    DE              ; save DE
15FB  E5        PUSH    HL              ; save HL
15FC  CB21      SLA     C               ;
15FE  210F36    LD      HL,#360F        ; start of table data
1601  09        ADD     HL,BC
1602  EB        EX      DE,HL
1603  217469    LD      HL,#6974
1606  1A        LD      A,(DE)          ; load A with table data
1607  13        INC     DE              ; next table entry
1608  77        LD      (HL),A          ; store
1609  23        INC     HL              ; next location
160A  3672      LD      (HL),#72
160C  23        INC     HL
160D  360C      LD      (HL),#0C
160F  23        INC     HL
1610  1A        LD      A,(DE)
1611  77        LD      (HL),A
1612  E1        POP     HL              ; restore HL
1613  D1        POP     DE              ; restore DE
1614  C9        RET                     ; return

; arrive when GameMode2 == #16 (level completed).  called from #0701

1615  CDBD30    CALL    #30BD           ; clear sprites
1618  3A2762    LD      A,(#6227)       ; load a with screen number
161B  0F        RRCA                    ; roll right with carry.  is this the rivets or the conveyors?
161C  d22f16    JP      NC,#162f        ; yes, skip ahead to #162F

                                        ; handle for girders or elevators, they are same here

161F  3A8863    LD      A,(#6388)       ; load A with this counter usually zero, counts from 1 to 5 when the level is
complete
1622  EF        RST     #28             ; jump based on A

1623  54 16                             ; #1654         ; 0
1625  70 16                             ; #1670         ; 1
1627  8A 16                             ; #168A         ; 2
1629  32 17                             ; #1732         ; 3
162B  57 17                             ; #1757         ; 4
162D  8E 17                             ; #178E         ; 5

162F  0F        RRCA                    ; roll right again.  is this the rivets ?
1630  D24116    JP      NC,#1641        ; yes, skip ahead

; else the conveyors

1633  3A8863    LD      A,(#6388)       ; load A with this usually zero, counts from 1 to 5 when the level is complete
1636  EF        RST     #28             ; jump based on A

1637  A3 16                             ; #16A3         ; 0
1639  BB 16                             ; #16BB         ; 1
163B  32 17                             ; #1732         ; 2
163D  57 17                             ; #1757         ; 3
163F  8E 17                             ; #178E         ; 4

; rivets

1641  CDBD1D    CALL    #1DBD           ; check for bonus items and jumping scores, rivets
1644  3A8863    LD      A,(#6388)       ; load A with usually zero, counts from 1 to 5 when the level is complete

1647  EF        RST     #28             ; jump based on A

1648  B6 17                             ; #17B6         ; 0
164A  69 30                             ; #3069         ; 1
164C  39 18                             ; #1839         ; 2
164E  6F 18                             ; #186F         ; 3
1650  80 18                             ; #1880         ; 4
1652  C6 18                             ; #18C6         ; 5

; jump here from #1622 when girders or elevators is finished.  step 1 of 6

1654  CD0817    CALL    #1708           ; clear all sounds, draw heart sprite, redraw girl sprite, clear "help", play end of
level sound
1657  215C38    LD      HL,#385C        ; load HL with start of kong graphic table data
165A  CD4E00    CALL    #004E           ; update kong's sprites
165D  3E20      LD      A,#20           ; A := #20
165F  320960    LD      (WaitTimerMSB),A    ; set timer to #20

1662  218863    LD      HL,#6388        ; load HL with end of level counter
1665  34        INC     (HL)            ; increase counter
1666  3E01      LD      A,#01           ; A := 1 = code for girders
```

```
1668  F7        RST     #30             ; if girders, continue below.  else RET

1669  210B69    LD      HL,#690B        ; load HL with start of kong sprite
166C  0EFC      LD      C,#FC           ; set movement for -4 pixels
166E  FF        RST     #38             ; move kong
166F  C9        RET                     ; return

; jump here from #1622 when girders or elevators is finished.  step 2 of 6

1670  DF        RST     #18             ; count down timer and only continue here if zero, else RET
1671  213239    LD      HL,#3932        ; load HL with start of kong's sprites table data
1674  CD4E00    CALL    #004E           ; update kong's sprites
1677  3E20      LD      A,#20           ; A := #20
1679  320960    LD      (WaitTimerMSB),A    ; set timer to #20
167C  218863    LD      HL,#6388        ; load HL with end of level counter
167F  34        INC     (HL)            ; increase counter
1680  3E04      LD      A,#04           ; A := 4 = 100 code for elevators
1682  F7        RST     #30             ; only continue here if elevators, else RET

1683  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
1686  0E04      LD      C,#04           ; set to move by 4
1688  FF        RST     #38             ; move kong by +4
1689  C9        RET                     ; return

; jump here from #1622 when girders or elevators is finished.  step 3 of 6

168A  DF        RST     #18             ; count down timer and only continue here if zero, else RET
168B  218C38    LD      HL,#388C        ; load HL with start of table data for kong
168E  CD4E00    CALL    #004E           ; update kong's sprites
1691  3E66      LD      A,#66           ; A := #66
1693  320C69    LD      (#690C),A       ; store into kong's right arm sprite
1696  AF        XOR     A               ; A := 0
1697  322469    LD      (#6924),A       ; clear the other side of kongs arm
169A  322C69    LD      (#692C),A       ; clear the girl sprite that kong is carrying
169D  32AF62    LD      (#62AF),A       ; clear the kong climbing counter
16A0  C36216    JP      #1662           ; jump back

; jump here from #1622 when conveyors is finished.  step 1 of 5

16A3  CD0817    CALL    #1708           ; clear all sounds, draw heart sprite, redraw girl sprite, clear "help", play end of
level sound
16A6  3A1069    LD      A,(#6910)       ; load A with kong's X position
16A9  D63B      SUB     #3B             ; subtract #3B
16AB  215C38    LD      HL,#385C        ; load HL with kong graphic table data
16AE  CD4E00    CALL    #004E           ; update kong's sprites to default kong graphic
16B1  210869    LD      HL,#6908        ; load HL with start of Kong sprite
16B4  4F        LD      C,A             ; load C with offset computed above to move kong back where he was
16B5  FF        RST     #38             ; move Kong
16B6  218863    LD      HL,#6388        ; load HL with end of level counter
16B9  34        INC     (HL)            ; increase counter
16BA  C9        RET                     ; return

; jump here from #1622 when conveyors is finished.  step 2 of 5

16BB  AF        XOR     A               ; A := 0
16BC  32A062    LD      (#62A0),A       ; clear top conveyor counter
16BF  3AA363    LD      A,(#63A3)       ; load A with direction vector for top conveyor
16C2  4F        LD      C,A             ; copy to C
16C3  3A1069    LD      A,(#6910)       ; load A with kong's X position
16C6  FE5A      CP      #5A             ; < #5A ?
16C8  D2E116    JP      NC,#16E1        ; yes, skip ahead

16CB  CB79      BIT     7,C
16CD  CAD516    JP      Z,#16D5         ; yes, skip next 2 steps

16D0  3E01      LD      A,#01           ; A := 1
16D2  32A062    LD      (#62A0),A       ; store into top conveyor counter

16D5  CD0226    CALL    #2602           ; ???
16D8  3AA363    LD      A,(#63A3)       ; load A with direction vector for top conveyor
16DB  4F        LD      C,A             ; C := 1
16DC  210869    LD      HL,#6908        ; load HL with start of Kong sprite
16DF  FF        RST     #38             ; move kong
16E0  C9        RET                     ; return

16E1  FE5D      CP      #5D             ; < #5D ?
16E3  DAEE16    JP      C,#16EE         ; no, skip ahead

16E6  CB79      BIT     7,C             ; is bit 7 of C zero?
16E8  CAD016    JP      Z,#16D0         ; yes, jump back

16EB  C3D516    JP      #16D5           ; jump back

16EE  218C38    LD      HL,#388C        ; load HL with start of table data for kong
16F1  CD4E00    CALL    #004E           ; update kong's sprites
16F4  3E66      LD      A,#66           ; A := #66
16F6  320C69    LD      (#690C),A       ; store into kong's right arm sprite for climbing
16F9  AF        XOR     A               ; A := 0
16FA  322469    LD      (#6924),A       ; clear kong's arm sprite
16FD  322C69    LD      (#692C),A       ; clear girl under kong's arm
1700  32AF62    LD      (#62AF),A       ; clear kong climbing counter
1703  218863    LD      HL,#6388        ; load HL with end of level counter
1706  34        INC     (HL)            ; increase counter
1707  C9        RET                     ; return
```

```
; called from #1654 and #16A3
; clears all sounds, draws heart sprite, redraws girl sprite, clear "help", play end of level sound

1708  CD1C01    CALL    #011C           ; clear all sounds
170B  21206A    LD      HL,#6A20        ; load HL with heart sprite
170E  3680      LD      (HL),#80        ; set heart sprite X position
1710  23        INC     HL              ; next
1711  3676      LD      (HL),#76        ; set heart sprite
1713  23        INC     HL              ; next
1714  3609      LD      (HL),#09        ; set heart sprite color
1716  23        INC     HL              ; next
1717  3620      LD      (HL),#20        ; set heart sprite Y position
1719  210569    LD      HL,#6905        ; load HL with girl's sprite
171C  3613      LD      (HL),#13        ; set girl's sprite
171E  21C475    LD      HL,#75C4        ; load HL with VRAM screen address
1721  112000    LD      DE,#0020        ; DE := #20
1724  3E10      LD      A,#10           ; A := #10
1726  CD1405    CALL    #0514           ; clear "help" that the girl yells
1729  218A60    LD      HL,#608A        ; load sound address
172C  3607      LD      (HL),#07        ; play sound for end of level
172E  23        INC     HL              ; HL now has sound duration
172F  3603      LD      (HL),#03        ; set duration to 3
1731  C9        RET                     ; return

; jump here from #1622 when girders or elevators is finished.  step 4 of 6
; jump here from #1622 when conveyors is finished.  step 3 of 5

1732  CD6F30    CALL    #306F           ; animate kong climbing up the ladder with girl under arm
1735  3A1369    LD      A,(#6913)       ; load A with kong sprite Y position
1738  FE2C      CP      #2C             ; < #2C ? (level of the girl)
173A  D0        RET     NC              ; yes, return

; else kong has grabbed the girl on the way out

173B  AF        XOR     A               ; A := #00
173C  320069    LD      (#6900),A       ; clear girl's head sprite
173F  320469    LD      (#6904),A       ; clear girl's body sprite
1742  320C69    LD      (#690C),A       ; clear kong's top right sprite
1745  3E6B      LD      A,#6B           ; A := #6B = code for sprite with kong's arm out
1747  322469    LD      (#6924),A       ; store into kong's right arm sprite for carrying girl
174A  3D        DEC     A               ; A := #6A = code for sprite with girl being carried
174B  322C69    LD      (#692C),A       ; store into girl being carried sprite
174E  21216A    LD      HL,#6A21        ; load HL with heart sprite
1751  34        INC     (HL)            ; change heart to broken
1752  218863    LD      HL,#6388        ; load HL with end of level counter
1755  34        INC     (HL)            ; increase counter
1756  C9        RET                     ; return

; jump here from #1622 when girders or elevators is finished.  step 5 of 6
; jump here from #1622 when conveyors is finished.  step 4 of 5

1757  CD6F30    CALL    #306F           ; animate kong climbing up the ladder with girl under arm
175A  CD6C17    CALL    #176C           ; ???
175D  23        INC     HL
175E  13        INC     DE
175F  CD8317    CALL    #1783           ; ???
1762  3E40      LD      A,#40           ; A := #40
1764  320960    LD      (WaitTimerMSB),A    ; set timer to #40
1767  218863    LD      HL,#6388        ; load HL with end of level counter
176A  34        INC     (HL)            ; increase counter
176B  C9        RET                     ; return

; called from #175A, above

176C  110300    LD      DE,#0003        ; load DE with offset to subtract
176F  212F69    LD      HL,#692F        ; load HL with girl under kong's arm Y position.  counting down, it will go through
all of kong's body
1772  060A      LD      B,#0A           ; for B = 1 to #0A

1774  A7        AND     A               ; clear carry flag
1775  7E        LD      A,(HL)          ; load A with Y position
1776  ED52      SBC     HL,DE           ; next offset
1778  FE19      CP      #19             ; girl still on screen?
177A  D27F17    JP      NC,#177F        ; yes, skip next step

177D  3600      LD      (HL),#00        ; set Y position to 0 = clear from screen ?

177F  2B        DEC     HL              ; previous data
1780  10F2      DJNZ    #1774           ; Next B

1782  C9        RET                     ; return

; called from #175F

1783  060A      LD      B,#0A           ; for B = 1 to #A

1785  7E        LD      A,(HL)          ; load A with ???
1786  A7        AND     A               ; == 0 ?
1787  C22600    JP      NZ,#0026        ; no, jump to #0026.  This will effectively RET twice

178A  19        ADD     HL,DE           ; else add offset for next memory
178B  10F8      DJNZ    #1785           ; next B

178D  C9        RET                     ; return
```

```
; jump here from #1622 when girders or elevators is finished.  step 6 of 6
; jump here from #1622 when conveyors is finished.  step 5 of 5

178E  DF        RST     #18             ; count down timer and only continue here if zero, else RET
178F  2A2A62    LD      HL,(#622A)      ; load HL with address for this screen/level
1792  23        INC     HL              ; next screen
1793  7E        LD      A,(HL)          ; load A with the screen for next
1794  FE7F      CP      #7F             ; at end ?
1796  C29D17    JP      NZ,#179D        ; no, skip next 2 steps

1799  21733A    LD      HL,#3A73        ; load HL with table for screens/levels for level 5+
179C  7E        LD      A,(HL)          ; load A with the screen

179D  222A62    LD      (#622A),HL      ; store screen address lookup for next time
17A0  322762    LD      (#6227),A       ; store A into screen number
17A3  110005    LD      DE,#0500        ; load task #5, parameter 0 ; adds bonus to player's score
17A6  CD9F30    CALL    #309F           ; insert task
17A9  AF        XOR     A               ; A := 0
17AA  328863    LD      (#6388),A       ; clear end of level counter
17AD  210960    LD      HL,WaitTimerMSB ; load HL with timer addr.
17B0  3630      LD      (HL),#30        ; set timer to #30
17B2  23        INC     HL              ; HL := GameMode2
17B3  3608      LD      (HL),#08        ; set game mode2 to 8
17B5  C9        RET                     ; return
17B3  C38613    JP      #1386           ; call additional code to increase the sub level

17B6  00        NOP

; arrive when rivets is cleared

17B7  CD1C01    CALL    #011C           ; clear all sounds
17BA  218A60    LD      HL,#608A        ; load HL with sound address
17BD  360E      LD      (HL),#0E        ; play sound for rivets falling and kong beating chest
17BF  23        INC     HL              ; HL := #608B = sound duration
17C0  3603      LD      (HL),#03        ; set duration to 3
17C2  3E10      LD      A,#10           ; A := #10 = code for clear space
17C4  112000    LD      DE,#0020        ; DE := #20
17C7  212376    LD      HL,#7623        ; load HL with video RAM location
17CA  CD1405    CALL    #0514           ; clear "help" on left side of girl
17CD  218375    LD      HL,#7583        ; load HL with video RAM location
17D0  CD1405    CALL    #0514           ; clear "help of right side of girl
17D3  21DA76    LD      HL,#76DA        ; load HL with center area of video ram
17D6  CD2618    CALL    #1826           ; clear screen area
17D9  11473A    LD      DE,#3A47        ; load DE with start of table data
17DC  CDA70D    CALL    #0DA7           ; draw the screen
17DF  21D576    LD      HL,#76D5        ; load HL with center area of video ram
17E2  CD2618    CALL    #1826           ; clear screen area
17E5  114D3A    LD      DE,#3A4D        ; load DE with start of table data
17E8  CDA70D    CALL    #0DA7           ; draw the screen
17EB  21D076    LD      HL,#76D0        ; load HL with center area of video ram
17EE  CD2618    CALL    #1826           ; clear screen area
17F1  11533A    LD      DE,#3A53        ; load DE with start of table data
17F4  CDA70D    CALL    #0DA7           ; draw the screen
17F7  21CB76    LD      HL,#76CB        ; load HL with center area of video ram
17FA  CD2618    CALL    #1826           ; clear screen area
17FD  11593A    LD      DE,#3A59        ; load DE with start of table data
1800  CDA70D    CALL    #0DA7           ; draw the screen
1803  215C38    LD      HL,#385C        ; load HL with start of kong graphic table data
1806  CD4E00    CALL    #004E           ; update kong's sprites
1809  210869    LD      HL,#6908        ; load HL with start of kong sprites
180C  0E44      LD      C,#44           ; load offset of #44
180E  FF        RST     #38             ; move kong
180F  210569    LD      HL,#6905        ; load HL with girl's sprite
1812  3613      LD      (HL),#13        ; set girl's sprite
1814  3E20      LD      A,#20           ; A := #20
1816  320960    LD      (WaitTimerMSB),A    ; set timer to #20
1819  3E80      LD      A,#80           ; A := #80
181B  329063    LD      (#6390),A       ; store into timer ???
181E  218863    LD      HL,#6388        ; load HL with end of level counter
1821  34        INC     (HL)            ; increase counter
1822  22C063    LD      (#63C0),HL      ; store into ???
1825  C9        RET                     ; return

; called from several places with HL preloaded with a video RAM address
; used to clear sections of the rivets screen when it is completed

1826  11DBFF    LD      DE,#FFDB        ; load DE with offset for each column
1829  0E0E      LD      C,#0E           ; for C = 1 to #0E
182B  3E10      LD      A,#10           ; A := #10 (clear space on screen)

182D  0605      LD      B,#05           ; for B = 1 to 5

182F  77        LD      (HL),A          ; store A into (HL) - clears the screen element
1830  23        INC     HL              ; next HL
1831  10FC      DJNZ    #182F           ; next B

1833  19        ADD     HL,DE           ; add offset to HL
1834  0D        DEC     C               ; next C
1835  C22D18    JP      NZ,#182D        ; loop until done

1838  C9        RET                     ; return

; arrive from #1647 when #6388 == 2

1839  219063    LD      HL,#6390        ; load HL with timer ???
```

```
183C  34        INC     (HL)            ; increase.  at zero?
183D  CA5918    JP      Z,#1859         ; yes, skip ahead

1840  7E        LD      A,(HL)          ; load A with the timer value
1841  E607      AND     #07             ; mask bits, now between 0 and 7.  zero?
1843  C0        RET     NZ              ; no, return

; kong is beating his chest after rivets have been cleared

1844  11CF39    LD      DE,#39CF        ; load DE with start of table data
1847  CB5E      BIT     3,(HL)          ; test bit 3.  True?
1849  2003      JR      NZ,#184E        ; Yes, skip next step

184B  11F739    LD      DE,#39F7        ; else load DE with other table start

184E  EB        EX      DE,HL           ; DE <> HL
184F  CD4E00    CALL    #004E           ; update kong's sprites
1852  210869    LD      HL,#6908        ; load HL with start of Kong sprite
1855  0E44      LD      C,#44           ; C := #44
1857  FF        RST     #38             ; move kong
1858  C9        RET                     ; return

1859  215C38    LD      HL,#385C        ; load HL with start of kong graphic table data
185C  CD4E00    CALL    #004E           ; update kong's sprites
185F  210869    LD      HL,#6908        ; load HL with start of Kong sprite
1862  0E44      LD      C,#44           ; C := #44
1864  FF        RST     #38             ; move kong
1865  3E20      LD      A,#20           ; A := #20
1867  320960    LD      (WaitTimerMSB),A    ; store into timer
186A  218863    LD      HL,#6388        ; load HL with end of level counter
186D  34        INC     (HL)            ; increase counter
186E  C9        RET                     ; return

; rivets has been cleared and kong is falling upside down
; arrive from #1647

186F  DF        RST     #18             ; count down timer and only continue here if zero, else RET

1870  211F3A    LD      HL,#3A1F        ; start of table data for kong upside down
1873  CD4E00    CALL    #004E           ; update kong's sprites
1876  3E03      LD      A,#03           ; A := 3
1878  328460    LD      (#6084),A       ; play falling sound
187B  218863    LD      HL,#6388        ; load HL with end of level counter
187E  34        INC     (HL)            ; increase
187F  C9        RET                     ; return

; arrive from #1647 when #6388 == 4

1880  210B69    LD      HL,#690B        ; load HL with kong start sprite
1883  0E01      LD      C,#01           ; load C with 1 pixel to move
1885  FF        RST     #38             ; move kong
1886  3A1B69    LD      A,(#691B)       ; load A with ???
1889  FED0      CP      #D0             ; == #D0 ?
188B  C0        RET     NZ              ; no, return

188C  3E20      LD      A,#20           ; A := #20
188E  321969    LD      (#6919),A       ; store into kong's face sprite - kong is now bigmouthed with crazy eyes
1891  21246A    LD      HL,#6A24        ; load HL with sprite address used for kong's aching head lines
1894  367F      LD      (HL),#7F        ; set sprite X value
1896  2C        INC     L               ; next
1897  3639      LD      (HL),#39        ; set sprite color
1899  2C        INC     L               ; next
189A  3601      LD      (HL),#01        ; set sprite value
189C  2C        INC     L               ; next
189D  36D8      LD      (HL),#D8        ; set sprite Y value
189F  21C676    LD      HL,#76C6        ; load HL with start of screen location to clear
18A2  CD2618    CALL    #1826           ; clear the top part of rivets
18A5  115F3A    LD      DE,#3A5F        ; load DE with table data for sections to clear after rivets done
18A8  CDA70D    CALL    #0DA7           ; draw the top girder where mario and girl meet

18AB  110400    LD      DE,#0004        ; load counters
18AE  012802    LD      BC,#0228        ; load counters
18B1  210369    LD      HL,#6903        ; set sprite girl table data Y position
18B4  CD3D00    CALL    #003D           ; move the girl down

18B7  3E00      LD      A,#00           ; A := 0 [why written this way?]
18B9  32AF62    LD      (#62AF),A       ; store into kong climbing counter
18BC  3E03      LD      A,#03           ; set boom sound duration
18BE  328260    LD      (#6082),A       ; play boom sound
18C1  218863    LD      HL,#6388        ; load HL with end of level counter
18C4  34        INC     (HL)            ; increase counter
18C5  C9        RET                     ; return

; arrive from #1647 when level is complete, last of 5 steps

18C6  21AF62    LD      HL,#62AF        ; load HL with kong climbing counter address
18C9  35        DEC     (HL)            ; decrease.  zero?
18CA  CA3D19    JP      Z,#193D         ; yes, skip ahead, handle next level routine

18CD  7E        LD      A,(HL)          ; load A with kong climbing counter
18CE  E607      AND     #07             ; mask bits, now between 0 and 7.  zero?
18D0  C0        RET     NZ              ; no , return

18D1  21256A    LD      HL,#6A25        ; load HL with ???
18D4  7E        LD      A,(HL)          ; get value
```

```
18D5  EE80      XOR     #80             ; toggle bit 7
18D7  77        LD      (HL),A          ; store result

18D8  211969    LD      HL,#6919        ; load HL with ???
18DB  46        LD      B,(HL)          ; load B with this value
18DC  CBA8      RES     5,B             ; clear bit 5 of B
18DE  AF        XOR     A               ; A := 0
18DF  CD0930    CALL    #3009           ; ???
18E2  F620      OR      #20             ; turn on bit 5
18E4  77        LD      (HL),A          ; store result

18E5  21AF62    LD      HL,#62AF        ; load HL with kong climbing counter
18E8  7E        LD      A,(HL)          ; get value
18E9  FEE0      CP      #E0             ; == #E0 ?
18EB  C21019    JP      NZ,#1910        ; no, skip ahead

18EE  3E50      LD      A,#50           ; A := #50
18F0  324F69    LD      (#694F),A       ; store into mario sprite Y value
18F3  3E00      LD      A,#00           ; A := 0
18F5  324D69    LD      (#694D),A       ; store into mario sprite value
18F8  3E9F      LD      A,#9F           ; A := #9F
18FA  324C69    LD      (#694C),A       ; set mario sprite X value at #9F
18FD  3A0362    LD      A,(#6203)       ; load A with mario X position
1900  FE80      CP      #80             ; < 80 ?
1902  D20F19    JP      NC,#190F        ; yes, skip next 4 steps

1905  3E80      LD      A,#80           ; A := #80
1907  324D69    LD      (#694D),A       ; store into mario sprite value
190A  3E5F      LD      A,#5F           ; A := #5F
190C  324C69    LD      (#694C),A       ; store into mario sprite X value

190F  7E        LD      A,(HL)          ; load A with ???

1910  FEC0      CP      #C0             ; == #C0 ?
1912  C0        RET     NZ              ; no, return

1913  218A60    LD      HL,#608A        ; load HL with sound address
1916  360C      LD      (HL),#0C        ; play sound for rivets cleared
1918  3A2962    LD      A,(#6229)       ; load A with level #
191B  0F        RRCA                    ; roll a right .  is this an odd level ?
191C  3802      JR      c,#1920         ; Yes, skip next step

191E  3605      LD      (HL),#05        ; else play sound for even numbered rivets

1920  23        INC     HL              ; HL := #608B = sound duration
1921  3603      LD      (HL),#03        ; set duration to 3
1923  21236A    LD      HL,#6A23        ; load HL with heart sprite
1926  3640      LD      (HL),#40        ; set heart sprite Y position
1928  2B        DEC     HL              ; decrement HL
1929  3609      LD      (HL),#09        ; set heart sprite color
192B  2B        DEC     HL              ; decrement HL
192C  3676      LD      (HL),#76        ; set heart sprite
192E  2B        DEC     HL              ; decrement HL
192f  368f      LD      (HL),#8f        ; set heart sprite X position
1931  3A0362    LD      A,(#6203)       ; load A with mario X position
1934  fe80      CP      #80             ; is mario on the left side of the screen?
1936  d0        RET     nc              ; yes, return

1937  3E6f      LD      A,#6f           ; else A := #6F
1939  32206A    LD      (#6A20),A       ; store A into heart sprite X position
1939  C3A113    JP      #13A1           ; call additional code to reset the sub level
193C  c9        RET                     ; return from sub

; kong has climbed off the screen at end of level

193D  2A2A62    LD      HL,(#622A)      ; load HL with contents of #622A.  this is a pointer to the levels/screens data
1940  23        INC     HL              ; increase HL.  = next level
1941  7E        LD      A,(HL)          ; load A with contents of HL = the screen we are going to play next
1942  fe7f      CP      #7f             ; is this the end code ?
1944  c24B19    JP      NZ,#194B        ; no, skip next 2 steps

1947  21733A    LD      HL,#3A73        ; yes, load HL with #3A73 = start of table data for screens/levels for level 5+
194A  7E        LD      A,(HL)          ; load A with screen number from table

194B  222A62    LD      (#622A),HL      ; store
194E  322762    LD      (#6227),A       ; store A into screen number
1951  212962    LD      HL,#6229        ; load HL with level number address
1954  34        INC     (HL)            ; increase #6229 by one
1955  110005    LD      DE,#0500        ; load task #5, parameter 0 ; adds bonus to player's score
1958  CD9F30    CALL    #309F           ; insert task
195B  AF        XOR     A               ; A := 0
195C  322E62    LD      (#622E),A       ; store into number of goofys to draw
195F  328863    LD      (#6388),A       ; store into end of level counter
1962  210960    LD      HL,WaitTimerMSB ; load HL with timer
1965  36E0      LD      (HL),#e0        ; set timer to #E0
1967  23        INC     HL              ; increase HL to GameMode2
1968  3608      LD      (HL),#08        ; set game mode2 to 8
196A  C9        RET                     ; return

; arrive from jump table at #0701 when GameMode2 == #17

196B  CD5208    CALL    #0852           ; clear screen and all sprites
196E  3A0E60    LD      A,(PlayerTurnB)       ; load A with current player number.  0 = player 1, 1 = player 2
1971  C612      ADD     A,#12           ; add #12
1973  320A60    LD      (GameMode2),A       ; store into game mode2, now had #12 for player 1 or #13 for player 2
```

```
1976  C9        RET                     ; return
196B  00000000000000000000000000        ; no operations - free space


; main routine

1977  CDEE21    CALL    #21EE           ; used during attract mode only.  sets virtual input.

; arrive here from #0701 when playing

197A  CDBD1D    CALL    #1DBD           ; check for bonus items and jumping scores, rivets
197D  CD8C1E    CALL    #1E8C           ; do stuff for items hit with hammer
1980  CDC31A    CALL    #1AC3           ; check for jumping
1983  CD721F    CALL    #1F72           ; roll barrels
1986  CD8F2C    CALL    #2C8F           ; roll barrels ?
1989  CD032C    CALL    #2C03           ; do barrel deployment ?
198C  CDED30    CALL    #30ED           ; update fires if needed
198F  CD042E    CALL    #2E04           ; update bouncers if on elevators
1992  CDEA24    CALL    #24EA           ; do stuff for pie factory
1995  CDDB2D    CALL    #2DDB           ; deploy fireball/firefoxes for conveyors and rivets
1998  CDD42E    CALL    #2ED4           ; do stuff for hammer
199B  CD0722    CALL    #2207           ; do stuff for conveyors
199E  CD331A    CALL    #1A33           ; check for and handle running over rivets
19A1  CD852A    CALL    #2A85           ; check for mario falling
19A4  CD461F    CALL    #1F46           ; handle mario falling
19A7  CDFA26    CALL    #26FA           ; do stuff for elevators
19AA  CDF225    CALL    #25F2           ; handle conveyor directions, adjust Mario's speed based on conveyor directions
19AD  CDDA19    CALL    #19DA           ; check for mario picking up bonus item
19B0  CDFB03    CALL    #03FB           ; check for kong beating chest and animate girl and her screams
19B3  CD0828    CALL    #2808           ; check for collisions with hostile sprites [set to NOPS to make mario invincible to
enemy sprites]
19B6  CD1D28    CALL    #281D           ; do stuff for hammers
19B9  CD571E    CALL    #1E57           ; check for end of level
19BC  CD071A    CALL    #1A07           ; handle when the bonus timer has run out
19BF  CDCB2F    CALL    #2FCB           ; for non-girder levels, checks for bonus timer changes. if the bonus counts down,
sets a possible new fire to be released,
                                        ; sets a bouncer to be deployed, updates the bonus timer onscreen, and checks for
bonus time running out
19C2  00        NOP
19C3  00        NOP
19C4  00        NOP                     ; no operations.  [a deleted call ?]

19C5  3A0062    LD      A,(#6200)       ; load A with 0 if mario is dead, 1 if he is alive
19C8  A7        AND     A               ; is mario alive?
19C9  C0        RET     NZ              ; yes, return to #00D2

; mario died

19CA  CD1C01    CALL    #011C           ; no, mario died.  clear all sounds
19CD  218260    LD      HL,#6082        ; load HL with boom sound address
19D0  3603      LD      (HL),#03        ; play boom sound for 3 units
19D2  210A60    LD      HL,GameMode2        ; load HL with game mode2
19D5  34        INC     (HL)            ; increase
19D6  2B        DEC     HL              ; HL := WaitTimerMSB (timer used for sound effects)
19D7  3640      LD      (HL),#40        ; set timer to wait 40 units
19D9  C9        RET                     ; return to #00D2

; called from #19AD as part of the main routine
; checks for bonus items being picked up

19DA  3A0362    LD      A,(#6203)       ; load A with Mario's X position
19DD  0603      LD      B,#03           ; for B = 1 to 3
19DF  210C6A    LD      HL,#6A0C        ; load HL with X position of first bonus

19E2  BE        CP      (HL)            ; are they equal?
19E3  CAED19    JP      Z,#19ED         ; yes, then test the Y position too

19E6  2C        INC     L
19E7  2C        INC     L
19E8  2C        INC     L
19E9  2C        INC     L               ; increase 4 times to point to next bonus item position
19EA  10F6      DJNZ    #19E2           ; Loop 3 times, check for the 3 items

19EC  C9        RET                     ; return

19ED  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
19F0  2C        INC     L
19F1  2C        INC     L
19F2  2C        INC     L               ; get HL to point to Y position of bonus item
19F3  BE        CP      (HL)            ; are they equal?
19F4  C0        RET     NZ              ; no, return from this test

19F5  2D        DEC     L               ; yes, decrement L 2 times to check if this item has already been picked up
19F6  2D        DEC     L
19F7  CB5E      BIT     3,(HL)          ; test bit 3 of HL, tells whether picked up already or not.  Item not already picked
up?
19F9  C0        RET     NZ              ; Item picked up already, then return

; bonus item has been picked up

19FA  2D        DEC     L               ; decrease L.  HL now has the starting address of the sprite that was picked up
19FB  224363    LD      (#6343),HL      ; store into this temp memory.  read from at #1E18
19FE  AF        XOR     A               ; A := 0
19FF  324263    LD      (#6342),A       ; store into ???.  read from at #1DD6
1A02  3C        INC     A               ; A := 1
```

```
1A03  324063    LD      (#6340),A       ; store into #6340 - usually 0, changes when mario picks up bonus item. jumps over
item turns to 1 quickly, then 2 until bonus disappears
1A06  C9        RET                     ; return

; called from main routine at #19BC

1A07  3A8663    LD      A,(#6386)       ; load A with the location which tells if the timer has run out yet.
1A0A  EF        RST     #28             ; jump based on A

1A0B  1E 1A                             ; #1A1E if zero return immediately, bonus timer has not run out
1A0D  15 1A                             ; #1A15
1A0F  1F 1A                             ; #1A1F
1A11  2A 1A                             ; #1A2A
1A13  00 00                             ; unused

; arrive from #1A0A

1A15  AF        XOR     A               ; A := 0
1A16  328763    LD      (#6387),A       ; clear timer which counts down when the timer runs out
1A19  3E02      LD      A,#02           ; A := 2
1A1B  328663    LD      (#6386),A       ; store into the location which tells if the timer has run out yet.
1A1E  C9        RET                     ; return

; arrive from #1A0A

1A1F  218763    LD      HL,#6387        ; load HL with timer address
1A22  35        DEC     (HL)            ; decreases the timer which counts down after time has run out. time out?
1A23  C0        RET     NZ              ; no, return

1A24  3E03      LD      A,#03           ; A := 3
1A26  328663    LD      (#6386),A       ; store 3 into #6386 - time is up for mario!
1A29  C9        RET                     ; return

; we arrive here when the timer runs out

1A2A  3A1662    LD      A,(#6216)       ; load A with jump indicator
1A2D  A7        AND     A               ; is mario jumping ?
1A2E  C0        RET     NZ              ; yes, return, mario never dies while jumping

1A2F  E1        POP     HL              ; no, pop HL to return to higher subroutine
1A30  C3D219    JP      #19D2           ; jump to mario died and return

; called from main routine
; check for running over rivets ?

1A33  3E08      LD      A,#08           ; A := 8 = 1000 binary = code for rivets
1A35  F7        RST     #30             ; continue here only on rivets, else RET

1A36  3A0362    LD      A,(#6203)       ; load A with mario's X position
1A39  FE4B      CP      #4B             ; == #4B = the column the left rivets are on ?
1A3B  CA4B1A    JP      Z,#1A4B         ; yes, skip ahead and set the indicator

1A3E  FEB3      CP      #B3             ; == #B3 = the column the right rivets are on ?
1A40  CA4B1A    JP      Z,#1A4B         ; yes, skip ahead and set the indicator

1A43  3A9162    LD      A,(#6291)       ; else load A with rivet column indicator
1A46  3D        DEC     A               ; is mario possibly traversing a column?
1A47  CA511A    JP      Z,#1A51         ; yes, skip ahead
1A4A  C9        RET                     ; else return

1A4B  3E01      LD      A,#01           ; A := 1
1A4D  329162    LD      (#6291),A       ; store into column indicator
1A50  C9        RET                     ; return

1A51  329162    LD      (#6291),A       ; clear the column indicator
1A54  47        LD      B,A             ; B := 0
1A55  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
1A58  3D        DEC     A               ; decrement
1A59  FED0      CP      #D0             ; compare with #D0.  is mario too low to go over a rivet?
1A5B  D0        RET     NC              ; yes, return

1A5C  07        RLCA                    ; rotate left = mult by 2
1A5D  D2621A    JP      NC,#1A62        ; no carry, skip next step

1A60  CBD0      SET     2,B             ; else B := 4

1A62  07        RLCA                    ;
1A63  07        RLCA                    ; rotate left twice = mult by 4
1A64  D2691A    JP      NC,#1A69        ; no carry, skip next step

1A67  CBC8      SET     1,B             ; B := B + 2
1A69  E607      AND     #07             ; mask bits in A, now between 0 and 7
1A6B  FE06      CP      #06             ; == 6 ?
1A6D  C2721A    JP      NZ,#1A72        ; no, skip next step

1A70  CBC8      SET     1,B             ; else set this bit
1A72  3A0362    LD      A,(#6203)       ; load A with mario's X position
1A75  07        RLCA                    ; rotate left
1A76  D27B1A    JP      NC,#1A7B        ; no carry, skip next step

1A79  CBC0      SET     0,B             ; B := B + 1
1A7B  219262    LD      HL,#6292        ; load HL with start of array of rivets
1A7E  78        LD      A,B             ; A := B
1A7F  85        ADD     A,L             ; add #92
1A80  6F        LD      L,A             ; copy to L
```

```
1A81  7E        LD      A,(HL)          ; get the status of the rivet mario is crossing
1A82  A7        AND     A               ; has this rivet already been traversed?
1A83  C8        RET     Z               ; yes, return

; a rivet has been traversed

1A84  3600      LD      (HL),#00        ; set this rivet as cleared
1A86  219062    LD      HL,#6290        ; load HL with address of number of rivets remaining
1A89  35        DEC     (HL)            ; decrease number of rivets
1A8A  78        LD      A,B             ; A := B
1A8B  010500    LD      BC,#0005        ; load BC with offset of 5
1A8E  1F        RRA                     ; rotate right.  carry?  (is this rivet on right side?)
1A8F  DABD1A    JP      C,#1ABD         ; yes, skip ahead and load HL with #012B and return to #1A95

1A92  21CB02    LD      HL,#02CB        ; else load HL with master offset for rivets

1A95  A7        AND     A               ; A == 0 ?
1A96  CA9E1A    JP      Z,#1A9E         ; yes, skip next 3 steps

1A99  09        ADD     HL,BC           ; add offset to HL
1A9A  3D        DEC     A               ; decrease A.  zero?
1A9B  C2991A    JP      NZ,#1A99        ; no, loop again

1A9E  010074    LD      BC,#7400        ; start of video RAM is #7400
1AA1  09        ADD     HL,BC           ; add offset computed based on which rivet is cleared
1AA2  3E10      LD      A,#10           ; A := #10 = clear space
1AA4  77        LD      (HL),A          ; erase the rivet
1AA5  2D        DEC     L               ; next video memory
1AA6  77        LD      (HL),A          ; erase the top of the rivet
1AA7  2C        INC     L               ;
1AA8  2C        INC     L               ; next video memory
1AA9  77        LD      (HL),A          ; erase underneath the rivet [ not needed , there is nothing there to erase ???]
1AAA  3E01      LD      A,#01           ; A := 1
1AAC  324063    LD      (#6340),A       ; store into #6340 - usually 0, changes when mario picks up bonus item. jumps over
item turns to 1 quickly, then 2 until bonus disappears
1AAF  324263    LD      (#6342),A       ; store into scoring indicator
1AB2  322562    LD      (#6225),A       ; store into bonus sound indicator
1AB5  3A1662    LD      A,(#6216)       ; load A with jump indicator
1AB8  A7        AND     A               ; is mario jumping ?
1AB9  CC951D    CALL    Z,#1D95         ; no, play the bonus sound

1ABC  C9        RET                     ; else return

; arrive from #1A8F above

1ABD  212B01    LD      HL,#012B        ; load HL with alternate master offset for rivets
1AC0  C3951A    JP      #1A95           ; jump back to program and resume

; check for jumping and other movements
; called from main routine at #1980

1AC3  3A1662    LD      A,(#6216)       ; load A with jump indicator
1AC6  3D        DEC     A               ; is mario already jumping?
1AC7  CAB21B    JP      Z,#1BB2         ; yes, jump ahead

1ACA  3A1E62    LD      A,(#621E)       ; else load A with jump coming down indicator
1ACD  A7        AND     A               ; is the jump almost done ?
1ACE  C2551B    JP      NZ,#1B55        ; yes, skip way ahead

1AD1  3A1762    LD      A,(#6217)       ; load A with hammer check
1AD4  3D        DEC     A               ; is hammer active?
1AD5  CAE61A    JP      Z,#1AE6         ; yes, skip ahead

1AD8  3A1562    LD      A,(#6215)       ; else load A with ladder check
1ADB  3D        DEC     A               ; is mario on a ladder?
1ADC  CA381B    JP      Z,#1B38         ; yes, skip ahead

1ADF  3A1060    LD      A,(InputState)      ; load A with input
1AE2  17        RLA                     ; is player pressing jump ?
1AE3  DA6E1B    JP      C,#1B6E         ; yes, begin jump subroutine

1AE6  CD1F24    CALL    #241F           ; else call this other sub which loads DE with something depending on mario's
position.  ladder check?

1AE9  3A1060    LD      A,(InputState)      ; load A with input
1AEC  1D        DEC     E               ; E == 1 ?
1AED  CAF51A    JP      Z,#1AF5         ; yes, jump ahead

1AF0  CB47      BIT     0,A             ; test bit 0 of input.  is player pressing right ?
1AF2  C28F1C    JP      NZ,#1C8F        ; yes, skip ahead

1AF5  15        DEC     D               ; else is D == 1 ?
1AF6  CAFE1A    JP      Z,#1AFE         ; yes, skip ahead

1AF9  CB4F      BIT     1,A             ; is player pressing left ?
1AFB  C2AB1C    JP      NZ,#1CAB        ; yes, skip ahead

1AFE  3A1762    LD      A,(#6217)       ; else load A with hammer check
1B01  3D        DEC     A               ; is the hammer active?
1B02  C8        RET     Z               ; yes, return

1B03  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
1B06  C608      ADD     A,#08           ; Add 8
1B08  57        LD      D,A             ; copy into D
1B09  3A0362    LD      A,(#6203)       ; load A with Mario's X position
```

```
1B0C  F603      OR      #03             ; turn on left 2 bits (0 and 1)
1B0E  CB97      RES     2,A             ; turn off bit 2
1B10  011500    LD      BC,#0015        ; load BC with #15 = number of ladders to check
1B13  CD6E23    CALL    #236E           ; check for ladders nearby if none, RET to higher sub.  else A := 0 if at bottom of
ladder, A := 1 if at top.  C has the ladder number/type?

; mario is near a ladder

1B16  F5        PUSH    AF              ; save AF for later
1B17  210762    LD      HL,#6207        ; load HL with movement indicator
1B1A  7E        LD      A,(HL)          ; load movement
1B1B  E680      AND     #80             ; mask bits
1B1D  F606      OR      #06             ; mask bits
1B1F  77        LD      (HL),A          ; store movement
1B20  211A62    LD      HL,#621A        ; load HL with ladder type address
1B23  3E04      LD      A,#04           ; A := 4
1B25  B9        CP      C               ; compare.  is the ladder broken?
1B26  3601      LD      (HL),#01        ; store 1 into ladder type = broken ladder by default
1B28  D22C1B    JP      NC,#1B2C        ; if ladder broken, skip next step

1B2B  35        DEC     (HL)            ; set indicator to unbroken ladder

1B2C  F1        POP     AF              ; restore AF
1B2D  A7        AND     A               ; A == 0 ?  is mario at bottom of ladder?
1B2E  CA4E1B    JP      Z,#1B4E         ; yes, skip ahead

; else mario at top of ladder

1B31  7E        LD      A,(HL)          ; load A with broken ladder indicator
1B32  A7        AND     A               ; is this ladder broken?
1B33  C0        RET     NZ              ; yes, return.  we can't go down broken ladders

; top of unbroken ladder

1B34  2C        INC     L               ; next HL := #621B
1B35  72        LD      (HL),D          ; store D
1B36  2C        INC     L               ; next HL := #621C
1B37  70        LD      (HL),B          ; store B

; if mario is on a ladder
; jump here from #1ADC

1B38  3A1060    LD      A,(InputState)      ; load A with input
1B3B  CB5F      BIT     3,A             ; is joystick pushed down ?
1B3D  C2F21C    JP      NZ,#1CF2        ; yes, skip ahead to handle

1B40  3A1562    LD      A,(#6215)       ; load A with ladder status
1B43  A7        AND     A               ; is mario on a ladder?
1B44  C8        RET     Z               ; no, return

1B45  3A1060    LD      A,(InputState)      ; load A with input
1B48  CB57      BIT     2,A             ; is joystick pushed up ?
1B4A  C2031D    JP      NZ,#1D03        ; yes, skip ahead to handle

1B4D  C9        RET                     ; else return

; mario is next to bottom of ladder

1B4E  2C        INC     L               ; next HL := #621B
1B4F  70        LD      (HL),B          ; store B
1B50  2C        INC     L               ; next HL := #621C
1B51  72        LD      (HL),D          ; store D
1B52  C3451B    JP      #1B45           ; loop back

1B55  211E62    LD      HL,#621E        ; load HL with jump coming down indicator
1B58  35        DEC     (HL)            ; decrease.  is it zero ?
1B59  C0        RET     NZ              ; no, return

; arrive here when jump is complete

1B5A  3A1862    LD      A,(#6218)       ; load A with hammer grabbing indicator
1B5D  321762    LD      (#6217),A       ; store into hammer indicator
1B60  210762    LD      HL,#6207        ; load HL with movement indicator address
1B63  7E        LD      A,(HL)          ; load A with movement indicator
1B64  E680      AND     #80             ; mask bits.  we only care about bit 7, which we leave as is.  all other bits are now
zero
1B66  77        LD      (HL),A          ; store into movement indicator.  mario is no longer jumping
1B67  AF        XOR     A               ; A := 0
1B68  320262    LD      (#6202),A       ; set mario animation state to 0
1B6B  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; jump initiated.  arrive from #1AE3 when jump pressed and jump not already underway etc.

1B6E  3E01      LD      A,#01           ; A := 1
1B70  321662    LD      (#6216),A       ; set jump indicator
1B73  211062    LD      HL,#6210        ; load HL with mario's jump direction address
1B76  3A1060    LD      A,(InputState)      ; load A with copy of input
1B79  018000    LD      BC,#0080        ; B:= 0, C := #80 = codes for jumping right
1B7C  1F        RRA                     ; rotate input right.  is joystick moved right ?
1B7D  DA8A1B    JP      C,#1B8A         ; yes, skip ahead

; jumping left or straight up

1B80  0180FF    LD      BC,#FF80        ; B := #FF, C := #80 = codes for jumping left
1B83  1F        RRA                     ; rotate right again.  jumping to the left ?
```

```
1B84  DA8A1B    JP      C,#1B8A         ; yes, skip next step

; else jumping straight up

1B87  010000    LD      BC,#0000        ; B := 0, C := 0 = codes for jumping straight up

1B8A  AF        XOR     A               ; A := 0
1B8B  70        LD      (HL),B          ; store B into #6210 = jump direction (0 = right, #FF = left, 0 = up)
1B8C  2C        INC     L               ; HL := #6211
1B8D  71        LD      (HL),C          ; store C into jump direction indicator (#80 for left or right, 0 for up)
1B8E  2C        INC     L               ; HL := #6212
1B8F  3601      LD      (HL),#01        ; store 1 into this indicator ???
1B91  2C        INC     L               ; HL := #6213
1B92  3648      LD      (HL),#48
1B94  2C        INC     L               ; HL := #6214 (jump counter)
1B95  77        LD      (HL),A          ; clear jump counter
1B96  320462    LD      (#6204),A
1B99  320662    LD      (#6206),A
1B9C  3A0762    LD      A,(#6207)       ; load movement indicator
1B9F  E680      AND     #80             ; clear right 4 bits and leftmost bit
1BA1  F60E      OR      #0E             ; set right bits to E = 1110
1BA3  320762    LD      (#6207),A       ; set jumping bits to indicate a jump in progress
1BA6  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
1BA9  320E62    LD      (#620E),A       ; save mario's Y position when jump
1BAC  218160    LD      HL,#6081        ; load HL with sound buffer address for jumping
1BAF  3603      LD      (HL),#03        ; load sound buffer jumping sound for 3 units (3 frames?)
1BB1  C9        RET                     ; return to main routine (#1983)

; arrive here when mario is already jumping from #1AC7

1BB2  DD210062  LD      IX,#6200        ; load IX with start of array for mario
1BB6  3A0362    LD      A,(#6203)       ; load A with mario's X position
1BB9  DD770B    LD      (IX+#0B),A      ; store into +B
1BBC  3A0562    LD      A,(#6205)       ; load A with mario's Y position
1BBF  DD770C    LD      (IX+#0C),A      ; store into +C = #620C = jump height
1BC2  CD9C23    CALL    #239C           ; handle jump stuff ?
1BC5  CD1F24    CALL    #241F           ; loads DE with something depending on mario's position
1BC8  15        DEC     D               ; D == 1 ?
1BC9  C2F21B    JP      NZ,#1BF2        ; no, skip ahead

; bounce mario off left side wall ?

1BCC  DD361000  LD      (IX+#10),#00    ; clear jump direction
1BD0  DD361180  LD      (IX+#11),#80    ; set +11 indicator to #80 (???)
1BD4  DDCB07FE  SET     7,(IX+#07)      ; set bit 7 of +7 = sprite used = make mario face the other way

1BD8  3A2062    LD      A,(#6220)       ; load A with falling too far indicator
1BDB  3D        DEC     A               ; == 1 ? (falling too far?)
1BDC  CAEC1B    JP      Z,#1BEC         ; yes, skip ahead

1BDF  CD0724    CALL    #2407           ; ???
1BE2  DD7412    LD      (IX+#12),H
1BE5  DD7513    LD      (IX+#13),L
1BE8  DD361400  LD      (IX+#14),#00    ; clear the +14 indicator (???)

1BEC  CD9C23    CALL    #239C           ; ???
1BEF  C3051C    JP      #1C05           ; skip ahead

1BF2  1D        DEC     E               ; decrease E.  at zero ?
1BF3  C2051C    JP      NZ,#1C05        ; no, skip ahead

; bounce mario off right side wall ?

1BF6  DD3610FF  LD      (IX+#10),#FF    ; set jump direction to left
1BFA  DD361180  LD      (IX+#11),#80    ; set +11 indicator to #80
1BFE  DDCB07BE  RES     7,(IX+#07)      ; reset bit 7 of +7 = sprite used = makes mario face the other way
1C02  C3D81B    JP      #1BD8           ; jump back to program

1C05  CD1C2B    CALL    #2B1C           ; do stuff for jumping, load A with landing indicator ?
1C08  3D        DEC     A               ; decrease A.  mario landing ?
1C09  CA3A1C    JP      Z,#1C3A         ; yes, skip ahead to handle

1C0C  3A1F62    LD      A,(#621F)       ; else load A with #621F = 1 when mario is at apex or on way down after jump, 0
otherwise.
1C0F  3D        DEC     A               ; decrease A.  at zero ?  is mario at apex or on way down ?
1C10  CA761C    JP      Z,#1C76         ; yes, skip ahead

1C13  3A1462    LD      A,(#6214)       ; load A with jump counter
1C16  D614      SUB     #14             ; == #14 ? (apex of jump)
1C18  C2331C    JP      NZ,#1C33        ; no, skip ahead

; mario at apex of jump ?

1C1B  3E01      LD      A,#01           ; A := 1
1C1D  321F62    LD      (#621F),A       ; store into #621F = 1 when mario is at apex or on way down after jump, 0 otherwise.
1C20  CD5328    CALL    #2853           ; check for items under mario
1C23  A7        AND     A               ; was an item jumped?
1C24  CAA61D    JP      Z,#1DA6         ; no, jump ahead to update mario sprite and RET

; an item was jumped

1C27  324263    LD      (#6342),A       ; yes, barrel has been jumped, set for later use
1C2A  3E01      LD      A,#01           ; A := 1
1C2C  324063    LD      (#6340),A       ; store into #6340 - usually 0, changes when mario picks up bonus item. jumps over
item turns to 1 quickly, then 2 until bonus disappears
```

```
1C2F  322562    LD      (#6225),A       ; store into bonus sound indicator

1C32  00        NOP                     ; No operation [what was here ???]

; can arrive from #1C18

1C33  3C        INC     A               ; increase A.  Will turn to zero 1 pixel before apex of jump
1C34  CC5429    CALL    Z,#2954         ; if zero, call this sub to check for hammer grab

1C37  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; arrive here when mario lands.  B is preloaded with a parameter

1C3A  05        DEC     B               ; B == 1 ?
1C3B  CA4F1C    JP      Z,#1C4F         ; if so, skip ahead

1C3E  3C        INC     A               ; increase A
1C3F  321F62    LD      (#621F),A       ; store into #621F = 1 when mario is at apex or on way down after jump, 0 otherwise.
1C42  AF        XOR     A               ; A := 0
1C43  211062    LD      HL,#6210        ; load HL with jump direction
1C46  0605      LD      B,#05           ; for B := 1 to 5

1C48  77        LD      (HL),A          ; clear this memory (jump direction, etc)
1C49  2C        INC     L               ; next HL
1C4A  10FC      DJNZ    #1C48           ; next B

1C4C  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; jump almost complete ...

1C4F  321662    LD      (#6216),A       ; store A into jump indicator
1C52  3A2062    LD      A,(#6220)       ; load A with falling too far indicator
1C55  EE01      XOR     #01             ; toggle rightmost bit [ change to LD A, #01 to enable infinite falling without
death]
1C57  320062    LD      (#6200),A       ; store into mario life indicator.  if mario fell too far, he will die.
1C5A  210762    LD      HL,#6207        ; load HL with address of movement indicator
1C5D  7E        LD      A,(HL)          ; load A with movement indicator
1C5E  E680      AND     #80             ; maks bits, leave bit 7 as is.  all other bits are zeroed.
1C60  F60F      OR      #0F             ; turn on all 4 low bits
1C62  77        LD      (HL),A          ; store result into movement indicator
1C63  3E04      LD      A,#04           ; A := 4
1C65  321E62    LD      (#621E),A       ; store into jump coming down indicator
1C68  AF        XOR     A               ; A := 0
1C69  321F62    LD      (#621F),A       ; store into #621F = 1 when mario is at apex or on way down after jump, 0 otherwise.
1C6C  3A2562    LD      A,(#6225)       ; load A with bonus sound indicator
1C6F  3D        DEC     A               ; was a bonus awarded?
1C70  CC951D    CALL    Z,#1D95         ; yes, call this sub to play bonus sound

1C73  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; mario is on way down from jump or falling

1C76  3A0562    LD      A,(#6205)       ; load A with mario's Y position
1C79  210E62    LD      HL,#620E        ; load HL with mario original Y position ?
1C7C  D60F      SUB     #0F             ; subtract #F
1C7E  BE        CP      (HL)            ; compare.  is mario falling too far ?
1C7F  DAA61D    JP      C,#1DA6         ; no, jump ahead to update mario sprite and RET

; mario falling too far on a jump

1C82  3E01      LD      A,#01           ; A := 1
1C84  322062    LD      (#6220),A       ; store into falling too far indicator
1C87  218460    LD      HL,#6084        ; load HL with address for falling sound
1C8A  3603      LD      (HL),#03        ; play falling sound for 3 units
1C8C  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; arrive here when joystick is being pressed right

1C8F  0601      LD      B,#01           ; B := 1 = movement to right
1C91  3A0F62    LD      A,(#620F)       ; load A with movement indicator
1C94  A7        AND     A               ; time to move mario ?
1C95  C2D21C    JP      NZ,#1CD2        ; yes, jump ahead

1C98  3A0262    LD      A,(#6202)       ; varies from 0, 2, 4, 1 when mario is walking left or right
1C9B  47        LD      B,A             ; copy into B. this is used in sub at #3009 called below
1C9C  3E05      LD      A,#05           ; A := 5
1C9E  CD0930    CALL    #3009           ; ??? change A depending on where mario is?
1CA1  320262    LD      (#6202),A       ; put back
1CA4  E603      AND     #03             ; mask bits, now between 0 and 3
1CA6  F680      OR      #80             ; turn on bit 7
1CA8  C3C21C    JP      #1CC2           ; skip ahead

; arrive here when joystick is being pressed left

1CAB  06FF      LD      B,#FF           ; B := #FF = -1 (movement to left)
1CAD  3A0F62    LD      A,(#620F)       ; load A with movement indicator
1CB0  A7        AND     A               ; time to move mario?
1CB1  C2D21C    JP      NZ,#1CD2        ; yes, skip ahead and move mario

1CB4  3A0262    LD      A,(#6202)       ; varies from 0, 2, 4, 1 when mario is walking left or right
1CB7  47        LD      B,A             ; copy to B.  this is used in sub at #3009 called below
1CB8  3E01      LD      A,#01           ; A := 1
1CBA  CD0930    CALL    #3009           ; ??? change A depending on where mario is?
1CBD  320262    LD      (#6202),A       ; put back
1CC0  E603      AND     #03             ; mask bits. now between 0 and 3
```

```
1CC2  210762    LD      HL,#6207        ; load HL with mario movement indicator/sprite value
1CC5  77        LD      (HL),A          ; store A into this
1CC6  1F        RRA                     ; rotate right.  is A odd?
1CC7  DC8F1D    CALL    C,#1D8F         ; yes , skip ahead to start walking sound and RET

1CCA  3E02      LD      A,#02           ; A := 2
1CCC  320F62    LD      (#620F),A       ; store into movement indicator (reset)
1CCF  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

1CD2  210362    LD      HL,#6203        ; load HL with mario X position address
1CD5  7E        LD      A,(HL)          ; load A with mario X position
1CD6  80        ADD     A,B             ; add movement (either 1 or #FF)
1CD7  77        LD      (HL),A          ; store new result
1Cd8  3A2762    LD      A,(#6227)       ; load A with screen number
1Cdb  3D        DEC     a               ; are we on the girders?
1Cdc  c2Eb1C    JP      NZ,#1Ceb        ; no, skip ahead

1Cdf  66        LD      h,(HL)          ; else load H with mario X position
1Ce0  3A0562    LD      A,(#6205)       ; load A with mario Y position
1Ce3  6f        LD      l,A             ; copy to L.  HL now has mario X,Y
1Ce4  cd3323    CALL    #2333           ; check for movement up/down a girder, might also change Y position ?
1Ce7  7D        LD      A,l             ; load A with new Y position
1Ce8  320562    LD      (#6205),A       ; store into Y position

1CEB  210F62    LD      HL,#620F        ; load HL with address of movement indicator
1CEE  35        DEC     (HL)            ; decrease movement indicator
1CEF  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; mario moving down on a ladder
; jump here from #1B3D

1CF2  3A0F62    LD      A,(#620F)       ; load A with movmement indicator (from 3 to 0)
1CF5  A7        AND     A               ; == 0 ?
1CF6  C28A1D    JP      NZ,#1D8A        ; no, skip ahead, decrease indicator and return

; ok for mario to move

1CF9  3E03      LD      A,#03           ; A := 3
1CFB  320F62    LD      (#620F),A       ; reset movement indicator to 3
1CFE  3E02      LD      A,#02           ; A := 2 pixels to move down
1D00  C3111D    JP      #1D11           ; skip ahead

; mario moving up on a ladder
; jump here from #1B4A

1D03  3A0F62    LD      A,(#620F)       ; load A with movement indicator (from 4 to 0)
1D06  A7        AND     A               ; time to move mario ?
1D07  C2761D    JP      NZ,#1D76        ; no, skip ahead

1D0A  3E04      LD      A,#04           ; A := 4
1D0C  320F62    LD      (#620F),A       ; reset movement indicator to 4 (slower movement going up)
1D0F  3EFE      LD      A,#FE           ; A := #FE = -2 pixels movement

1D11  210562    LD      HL,#6205        ; load HL with mario Y position address
1D14  86        ADD     A,(HL)          ; add A to Y position
1D15  77        LD      (HL),A          ; store result into Y position
1D16  47        LD      B,A             ; copy to B
1D17  3A2262    LD      A,(#6222)       ; load A with ladder toggle
1D1A  EE01      XOR     #01             ; toggle the bit
1D1C  322262    LD      (#6222),A       ; store.  is it zero?
1D1F  C2511D    JP      NZ,#1D51        ; no, skip ahead

1D22  78        LD      A,B             ; A := B =  mario Y position
1D23  C608      ADD     A,#08           ; add 8 [offset for mario's actual position ???]
1D25  211C62    LD      HL,#621C        ; load HL with Y value of top of ladder
1D28  BE        CP      (HL)            ; is mario at top of ladder ?
1D29  CA671D    JP      Z,#1D67         ; yes, skip ahead to handle

1D2C  2D        DEC     L               ; HL := #621B = Y value of bottom of ladder
1D2D  96        SUB     (HL)            ; is mario at bottom of ladder ?
1D2E  CA671D    JP      Z,#1D67         ; yes, skip ahead to handle

1D31  0605      LD      B,#05           ; B := 5
1D33  D608      SUB     #08             ; subtract 8.  zero?
1D35  CA3F1D    JP      Z,#1D3F         ; yes, skip next 4 steps

1D38  05        DEC     B               ; B := 4
1D39  D604      SUB     #04             ; subtract 4.  zero?
1D3B  CA3F1D    JP      Z,#1D3F         ; yes, skip next step

1D3E  05        DEC     B               ; B := 3

1D3F  3E80      LD      A,#80           ; A := #80
1D41  210762    LD      HL,#6207        ; load HL with address of mario movement indicator/sprite value
1D44  A6        AND     (HL)            ; mask bits with movement
1D45  EE80      XOR     #80             ; toggle bit 7
1D47  B0        OR      B               ; turn on bits based on ladder position
1D48  77        LD      (HL),A          ; store into mario movement indicator/sprite value

1D49  3E01      LD      A,#01           ; A := 1
1D4B  321562    LD      (#6215),A       ; store into ladder status.  mario is on a ladder now
1D4E  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

1D51  2D        DEC     L
```

```
1D52  2D        DEC     L               ; HL := #6203
1D53  7E        LD      A,(HL)          ; load A with mario sprite value
1D54  F603      OR      #03             ; turn on bits 0 and 1
1D56  CB97      RES     2,A             ; clear bit 2
1D58  77        LD      (HL),A          ; store into mario sprite
1D59  3A2462    LD      A,(#6224)       ; load A with sound alternator
1D5C  EE01      XOR     #01             ; toggle bit 0
1D5E  322462    LD      (#6224),A       ; store result
1D61  CC8F1D    CALL    Z,#1D8F         ; if zero, play walking sound for moving on ladder

1D64  C3491D    JP      #1D49           ; jump back

; arrive from #1D29 when mario at top or bottom of ladder

1D67  3E06      LD      A,#06           ; A := 6
1D69  320762    LD      (#6207),A       ; store into mario movement indicator/sprite value
1D6C  AF        XOR     A               ; A := 0
1D6D  321962    LD      (#6219),A       ; clear this status indicator
1D70  321562    LD      (#6215),A       ; clear ladder status.  mario no longer on ladder
1D73  C3A61D    JP      #1DA6           ; jump ahead to update mario sprite and RET

; jump here from #1D07 when going up a ladder but not actually moving

1D76  3A1A62    LD      A,(#621A)       ; load A with this indicator.  set when mario is on moving ladder or broken ladder
1D79  A7        AND     A               ; is mario boarding or on a retracting or broken ladder?
1D7A  CA8A1D    JP      Z,#1D8A         ; no, skip ahead

; mario on or moving onto a rectracting or broken ladder

1D7D  321962    LD      (#6219),A       ; store 1 into status indicator
1D80  3A1C62    LD      A,(#621C)       ; load A with Y value of top of ladder
1D83  D613      SUB     #13             ; subtract #13
1D85  210562    LD      HL,#6205        ; load HL with mario Y position address
1D88  BE        CP      (HL)            ; is mario at or above the top of ladder ?
1D89  D0        RET     NC              ; yes, return without changing movement

1D8A  210F62    LD      HL,#620F        ; else load HL with address of movement indicator
1D8D  35        DEC     (HL)            ; decrease
1D8E  C9        RET                     ; return

; mario is walking

1D8F  3E03      LD      A,#03           ; load sound duration of 3 for walking
1D91  328060    LD      (#6080),A       ; store into walking sound buffer
1D94  C9        RET                     ; return

; arrive here when walking over a rivet, not jumping.  from #1AB9, or from #1C70

1D95  322562    LD      (#6225),A       ; store A into bonus sound indicator.  A is zero so this clears the indicator
1D98  3A2762    LD      A,(#6227)       ; load A with screen number
1D9B  3D        DEC     A               ; is this the girders?
1D9C  C8        RET     Z               ; yes , then return, we don't play this sound for the girders

; play bonus sound

1D9D  218A60    LD      HL,#608A        ; else load HL with sound address
1DA0  360D      LD      (HL),#0D        ; play bonus sound
1DA2  2C        INC     L               ; HL := #608B = sound duration
1DA3  3603      LD      (HL),#03        ; set sound duration to 3
1DA5  C9        RET                     ; return

; update mario sprite

1DA6  214C69    LD      HL,#694C        ; load HL with mario sprite X position
1DA9  3A0362    LD      A,(#6203)       ; load A with mario's X position
1DAC  77        LD      (HL),A          ; store into hardware sprite mario X position
1DAD  3A0762    LD      A,(#6207)       ; load A with movement indicator
1DB0  2C        INC     L               ; HL := #694D = hardware mario sprite
1DB1  77        LD      (HL),A          ; store into hardware mario sprite value
1DB2  3A0862    LD      A,(#6208)       ; load A with mario color
1DB5  2C        INC     L               ; HL := #694E = hardware mario sprite color
1DB6  77        LD      (HL),A          ; store into mario sprite color
1DB7  3A0562    LD      A,(#6205)       ; load A with mario Y position
1DBA  2C        INC     L               ; HL := #694F = mario sprite Y position
1DBB  77        LD      (HL),A          ; store into mario sprite Y position
1DBC  C9        RET                     ; return


; called from main routine at #197A
; also called from other areas


1DBD  3A4063    LD      A,(#6340)       ; load A with #6340 - usually 0, changes when mario picks up bonus item. jumps over
item turns to 1 quickly, then 2 until bonus disappears
1DC0  EF        RST     #28             ; jump based on A

1DC1  49 1E                             ; #1E49 = no item.  returns immediately
1DC3  C9 1D                             ; #1DC9 = item just picked up
1DC5  4A 1E                             ; #1E4A = bonus appears
1DC7  00 00                             ; unused

; an item was just picked up / jumped over / hit with hammer

1DC9  3E40      LD      A,#40           ; A := #40
1DCB  324163    LD      (#6341),A       ; store into timer
```

```
1DCE  3E02      LD      A,#02           ; A := 2
1DD0  324063    LD      (#6340),A       ; store into #6340 - usually 0, changes when mario picks up bonus item. jumps over
item turns to 1 quickly, then 2 until bonus disappears
1DD3  3A4263    LD      A,(#6342)       ; load A with scoring indicator
1DD6  1F        RRA                     ; roll right.  is this a jumped item?
1DD7  DA703E    JP      C,#3E70         ; yes, award points for jumping items [ patch ? orig code had JP C,#1E25 ??? ]

1DDA  1F        RRA                     ; else roll right
1DDB  DA001E    JP      C,#1E00         ; award for hitting regular barrel with hammer

1DDE  1F        RRA                     ; roll right.  hit blue barrel with hammer?
1DDF  DAF51D    JP      C,#1Df5         ; yes, skip ahead to handle

; else it was a bonus item pickup

1DE2  218560    LD      HL,#6085        ; else load HL with bonus sound address
1DE5  3603      LD      (HL),#03        ; play bonus sound for 3 duration
1DE7  3A2962    LD      A,(#6229)       ; load A with level #
1DEA  3D        DEC     A               ; decrease A.  is this level 1 ?
1DEB  cA001E    JP      Z,#1E00         ; yes, jump ahead for 300 pts

1DEE  3D        DEC     A               ; else is this level 2 ?
1DEF  CA081E    JP      Z,#1E08         ; yes, award 500 pts

1DF2  C3101E    JP      #1E10           ; else award 800 pts

; blue barrel hit with hammer

1DF5  3A1860    LD      A,(RngTimer1)       ; load timer, a psuedo random number
1DF8  1F        RRA                     ; roll right = 50% chance of 500 points
1DF9  DA081E    JP      C,#1E08         ; award 500 points

1DFC  1F        RRA                     ; roll right again, gives overall 25% chance of 800 points
1DFD  DA101E    JP      C,#1E10         ; award 800 points

; else award 300 points

1E00  067D      LD      B,#7D           ; set sprite for 300 points
1E02  110300    LD      DE,#0003        ; set points at 300
1E05  c3151E    JP      #1E15           ; award points

; award 500 pts

1E08  067E      LD      B,#7E           ; set sprite for 500 points
1E0A  110500    LD      DE,#0005        ; set points at 500
1E0D  C3151E    JP      #1E15           ; award points

; award 800 pts

1E10  067f      LD      B,#7f           ; set sprite for 800 points
1E12  110800    LD      DE,#0008        ; set points at 800

1E15  cd9f30    CALL    #309f           ; insert task to add score

; arrive here when bonus item picked up or smashed with hammer

1E18  2A4363    LD      HL,(#6343)      ; load HL with contents of #6343 , this gives the address of the sprite location
1E1B  7E        LD      A,(HL)          ; load A with the X position of the sprite in question
1E1C  3600      LD      (HL),#00        ; clear the sprite from the screen
1E1E  2C        INC     l               ; increase L 3 times
1E1F  2C        INC     l               ;
1E20  2C        INC     l               ;
1E21  4E        LD      c,(HL)          ; load C with the Y position of the item
1E22  c3361E    JP      #1E36           ; jump ahead

1E25  110100    LD      DE,#0001        ; load task for scoring, 100 pts [ never arrive at this line ??? possibly orig code
came from #1DD7 ]

1E28  CD9F30    CALL    #309F           ; insert task to add score
1E2B  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
1E2E  C614      ADD     A,#14           ; add #14
1E30  4F        LD      C,A             ; store into C
1E31  3A0362    LD      A,(#6203)       ; load A with mario's X position

1E34  00        NOP
1E35  00        NOP                     ; [ what used to be here?  was it LD B,#7B to set sprite for 100 pts? ]

; draw the bonus score on the screen

1E36  21306A    LD      HL,#6A30        ; load HL with scoring sprite start
1E39  77        LD      (HL),A          ; store X position
1E3A  2C        INC     L               ; next location
1E3B  70        LD      (HL),B          ; store sprite graphic
1E3C  2C        INC     L               ; next
1E3D  3607      LD      (HL),#07        ; store color code 7
1E3F  2C        INC     L               ; next
1E40  71        LD      (HL),C          ; store Y position
1E41  3E05      LD      A,#05           ; A := 5 = binary 0101
1E43  F7        RST     #30             ; only allow continue on girders and elevators, others do RET here [no bonus sound
for killing firefox with hammer]
1E44  218560    LD      HL,#6085        ; load HL with bonus sound address
1E47  3603      LD      (HL),#03        ; play bonus sound for 3 duration
1E49  C9        RET                     ; return

; arrive here from #1DC0 when bonus appears
```

```
1E4A  214163    LD      HL,#6341        ; load HL with timer
1E4D  35        DEC     (HL)            ; has it run out yet ?
1E4E  C0        RET     NZ              ; no, return

1E4F  AF        XOR     A               ; else A := 0
1E50  32306A    LD      (#6A30),A       ; clear this
1E53  324063    LD      (#6340),A       ; clear this
1E56  C9        RET                     ; return

; called from main routine at #19B9
; checks for end of level ?

1E57  3A2762    LD      A,(#6227)       ; load a with screen number
1E5A  cb57      BIT     2,A             ; are we on the rivets?
1E5C  c2801E    JP      NZ,#1E80        ; yes, skip ahead to handle

1E5f  1F        rra                     ; else rotate right with carry
1E60  3A0562    LD      A,(#6205)       ; load A with y position of mario
1E63  dA7A1E    JP      c,#1E7A         ; skip ahead on girders and elevators

1E66  fe51      CP      #51             ; else on the conveyors.  is mario high enough to end level?
1E68  d0        RET     nc              ; no, return

1E69  3A0362    LD      A,(#6203)       ; else load A with mario's X position
1E6C  17        RLA                     ; on left or right side of screen?

1E6D  3E00      LD      A,#00           ; load A with #00.  sprite for facing left
1E6F  DA741E    JP      C,#1E74         ; if on left side, skip next step

1E72  3E80      LD      A,#80           ; else load A with sprite facing right
1E74  324D69    LD      (#694D),A       ; set mario sprite
1E77  C3851E    JP      #1E85           ; jump ahead

; check for end of level on girders and elevators

1E7A  FE31      CP      #31             ; are we on top level (rescued girl?)
1E7C  D0        RET     NC              ; no, return

1E7D  C36D1E    JP      #1E6D           ; level has been fished.  jump to end of level routine.

; arrive here when on rivets

1E80  3A9062    LD      A,(#6290)       ; load A with number of rivets left
1E83  A7        AND     A               ; all done with rivets ?
1E84  C0        RET     NZ              ; no, return

1E85  3E16      LD      A,#16           ; else A := #16
1E87  320A60    LD      (GameMode2),A       ; store into game mode2
1E8A  E1        POP     HL              ; pop stack to get higher address
1E8B  C9        RET                     ; return to a higher level [returns to #00D2]

; called from main routine at #197D
; handles items hit with hammer

1E8C  3A5063    LD      A,(#6350)       ; load A with hammer hit item indicator
1E8F  A7        AND     A               ; is an item being smashed ?
1E90  C8        RET     Z               ; no, return

1E91  CD961E    CALL    #1E96           ; else call sub below
1E94  E1        POP     HL              ; then return to a higher sub
1E95  C9        RET                     ; returns to #00D2

1E96  3A4563    LD      A,(#6345)       ; load A with this

; #6345 - usually 0.  changes to 1, then 2 when items are hit with the hammer

1E99  EF        RST     #28             ; jump based on A

1E9A  A0 1E                     0       ; #1EA0
1E9C  09 1F                     1       ; #1F09
1E9E  23 1F                     2       ; #1F23

; arrive right when an item is hit

1EA0  3A5263    LD      A,(#6352)       ; load A with ???
1EA3  FE65      CP      #65             ; == #65 ?
1EA5  21B869    LD      HL,#69B8        ; load HL with sprites for pies
1EA8  CAB41E    JP      Z,#1EB4         ; yes, skip next 3 steps

1EAB  21D069    LD      HL,#69D0        ; load HL with start of fire sprites ???
1EAE  DAB41E    JP      C,#1EB4         ; if carry, then skip next step

1EB1  218069    LD      HL,#6980        ; HL is X position of a barrel

1EB4  DD2A5163  LD      IX,(#6351)      ; load IX with start of item array for the item hit
1EB8  1600      LD      D,#00           ; D := 0
1EBA  3A5363    LD      A,(#6353)       ; load A with the offset for each item in the array
1EBD  5F        LD      E,A             ; copy to E.  DE now has the offset
1EBE  010400    LD      BC,#0004        ; BC := 4
1EC1  3A5463    LD      A,(#6354)       ; load A with the index of the item hit
1EC4  A7        AND     A               ; == 0 ?
1EC5  CACF1E    JP      Z,#1ECF         ; yes, skip ahead, we use the default HL and IX

1EC8  09        ADD     HL,BC           ; add offset
```

```
1EC9  DD19      ADD     IX,DE           ; add offset
1ECB  3D        DEC     A               ; decrease counter.  done ?
1ECC  C2C81E    JP      NZ,#1EC8        ; no, loop again

1ECF  DD360000  LD      (IX+#00),#00    ; set this sprite as no longer active
1ED3  DD7E15    LD      A,(IX+#15)      ; load A with +15 (0 = normal barrel,  1 = blue barrel, see next comments)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; It turns out that IX+15 is used by firefoxes and fireballs as a counter for their animation
; This value can be 0, 1, or 2 and is updated every frame
;
; For pies, this value is 0, #7C or #CC, because it grabs the +5 slot of the next pie when one is hit
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

1ED6  A7        AND     A               ; ==0 ? is this a regular barrel?  (sometimes fires and pies fall here too)
1ED7  3E02      LD      A,#02           ; A := 2, used for 300 pts
1ED9  CADE1E    JP      Z,#1EDE         ; yes, skip next step

1EDC  3E04      LD      A,#04           ; else A := 4, used for random points (blue barrel, sometimes fire, sometimes pie)

1EDE  324263    LD      (#6342),A       ; store A into scoring indicator
1EE1  012C6A    LD      BC,#6A2C        ; load BC with scoring sprite address
1EE4  7E        LD      A,(HL)          ; load A with sprite value ?
1EE5  3600      LD      (HL),#00        ; clear the sprite that was hit
1EE7  02        LD      (BC),A          ; store sprite value into the scoring sprite
1EE8  0C        INC     C               ; next
1EE9  2C        INC     L               ; next
1EEA  3E60      LD      A,#60           ; A := #60 = sprite for large bluewhite circle
1EEC  02        LD      (BC),A          ; store into sprite graphic
1EED  0C        INC     C               ; next
1EEE  2C        INC     L               ; next
1EEF  3E0C      LD      A,#0C           ; A := #0C = color code
1EF1  02        LD      (BC),A          ; store into sprite color
1EF2  0C        INC     C               ; next
1EF3  2C        INC     L               ; next
1EF4  7E        LD      A,(HL)          ; load A with Y value for sprite hit
1EF5  02        LD      (BC),A          ; store into Y value for scoring sprite
1EF6  214563    LD      HL,#6345        ; load HL with item hit phase counter address

; #6345 - usually 0.  changes to 1, then 2 when items are hit with the hammer
; item has been hit by hammer

1EF9  34        INC     (HL)            ; increase the item hit phase counter
1EFA  2C        INC     L               ; HL := #6346 = a timer used for hammering items?
1EFB  3606      LD      (HL),#06        ; set timer to 6
1EFD  2C        INC     L               ; HL := #6347 = counter for number of times to change between circle and small circle
1EFE  3605      LD      (HL),#05        ; set to 5
1F00  218A60    LD      HL,#608A        ; load HL with sound buffer address
1F03  3606      LD      (HL),#06        ; play sound for hammering object
1F05  2C        INC     L               ; HL := 608B = sound duration
1F06  3603      LD      (HL),#03        ; set duration to 3
1F08  C9        RET                     ; return

; item has been hit by hammer , phase 2 of 3

1F09  214663    LD      HL,#6346        ; load HL with timer
1F0C  35        DEC     (HL)            ; count down.  zero ?
1F0D  C0        RET     NZ              ; no, return

1F0E  3606      LD      (HL),#06        ; else reset counter to 6
1F10  2C        INC     L               ; HL := #6347 = counter for this function
1F11  35        DEC     (HL)            ; decrease counter.  zero?
1F12  CA1D1F    JP      Z,#1F1D         ; yes, skip ahead

1F15  212D6A    LD      HL,#6A2D        ; else load HL with scoring sprite graphic
1F18  7E        LD      A,(HL)          ; get value
1F19  EE01      XOR     #01             ; toggle bit 0 = change sprite to small circle or back again
1F1B  77        LD      (HL),A          ; store
1F1C  C9        RET                     ; return

1F1D  3604      LD      (HL),#04        ; store 4 into #6347 = timer?
1F1F  2D        DEC     L               ;
1F20  2D        DEC     L               ; HL := #6345
1F21  34        INC     (HL)            ; increase item hit phase counter
1F22  C9        RET                     ; return

; arrive from jump at #1E99 when an item is hit with hammer (last step of 3)

1F23  214663    LD      HL,#6346        ; load HL with timer?
1F26  35        DEC     (HL)            ; count down.  zero ?
1F27  C0        RET     NZ              ; no, return

1F28  360C      LD      (HL),#0C        ; reset counter to #C
1F2A  2C        INC     L               ; HL := #6347 = counter
1F2B  35        DEC     (HL)            ; decrease counter.  zero?
1F2C  CA341F    JP      Z,#1F34         ; yes, skip ahead

1F2F  212D6A    LD      HL,#6A2D        ; no, load HL with sprite graphic
1F32  34        INC     (HL)            ; increase
1F33  C9        RET                     ; return

1F34  2D        DEC     L
1F35  2D        DEC     L               ; HL := 6345
```

```
1F36  AF        XOR     A               ; A := 0
1F37  77        LD      (HL),A          ; store into HL.  reset the item being hit with hammer
1F38  325063    LD      (#6350),A       ; store into item hit indicator
1F3B  3C        INC     A               ; A := 11:18 AM 6/15/2009
1F3C  324063    LD      (#6340),A       ; store into bonus indicator
1F3F  212C6A    LD      HL,#6A2C        ; load HL with location of item hit
1F42  224363    LD      (#6343),HL      ; store into #6343 for use later
1F45  C9        RET                     ; return

; called from main routine at #19A4

1F46  3A2162    LD      A,(#6221)       ; load A with falling indicator.  also set when mario lands from jumping off elevator
1F49  A7        AND     A               ; is mario falling?
1F4A  C8        RET     Z               ; no, return

; mario is falling

1F4B  AF        XOR     A               ; A := 0
1F4C  320462    LD      (#6204),A
1F4F  320662    LD      (#6206),A
1F52  322162    LD      (#6221),A       ; clear mario falling indicator
1F55  321062    LD      (#6210),A       ; clear jump direction
1F58  321162    LD      (#6211),A
1F5B  321262    LD      (#6212),A       ; clear this indicator (???)
1F5E  321362    LD      (#6213),A
1F61  321462    LD      (#6214),A       ; clear jump counter
1F64  3C        INC     A               ; A := 1
1F65  321662    LD      (#6216),A       ; set jump indicator
1F68  321F62    LD      (#621F),A       ; set #621F = 1 when mario is at apex or on way down after jump, 0 otherwise.
1F6B  3A0562    LD      A,(#6205)       ; load A with ???
1F6E  320E62    LD      (#620E),A       ; store into ???
1F71  C9        RET                     ; return

; called from main routine at #1983
; used to roll barrels

1F72  3A2762    LD      A,(#6227)       ; load a with screen number
1F75  3D        DEC     a               ; is this the girders ?
1F76  c0        RET     NZ              ; no, return

; yes, we are on girders
; this subroutine checks the barrels, if any are rolling it does something, otherwise returns

1F77  DD210067  LD      IX,#6700        ; load IX with start of barrel array
1F7B  218069    LD      HL,#6980        ; load HL with start of sprites used for barrels
1F7E  112000    LD      DE,#0020        ; load DE with offset of #20.  used for checking next barrel
1F81  060A      LD      B,#0A           ; for B = 1 to #0A ( do for each barrel)

1F83  DD7E00    LD      A,(IX+#00)      ; Load A with Barrel indicator (0 = no barrel, 2 = being deployed, 1=rolling)
1F86  3D        DEC     A               ; Is this barrel rolling ?
1F87  CA931F    JP      Z,#1F93         ; Yes, jump ahead

1F8A  2C        INC     L               ; otherwise increase L by 4
1F8B  2C        INC     L
1F8C  2C        INC     L
1F8D  2C        INC     L
1F8E  DD19      ADD     IX,DE           ; Add offset to check for next barrel
1F90  10F1      DJNZ    #1F83           ; Next B

1F92  C9        RET                     ; return

1F93  DD7E01    LD      A,(IX+#01)      ; Load Crazy Barrel indicator
1F96  3D        DEC     A               ; is this a crazy barrel?
1F97  CAEC20    JP      Z,#20EC         ; Yes, jump ahead

1F9A  DD7E02    LD      A,(IX+#02)      ; no load A with next indicator - determines the direction of the barrel
1F9D  1F        RRA                     ; Is this barrel going down a ladder?
1F9E  DAAC1F    JP      C,#1FAC         ; Yes, jump away to ladder sub.

1FA1  1F        RRA                     ; Is this barrel moving right?
1FA2  DAE51F    JP      C,#1FE5         ; yes, jump away to move right sub.

1FA5  1F        RRA                     ; is this barrel moving left?
1FA6  DAEF1F    JP      C,#1FEF         ; yes, jump to moving left sub

1FA9  C35320    JP      #2053           ; else jump ahead

; arrived here because the barrel is going down a ladder from #1F9E

1FAC  D9        EXX                     ; exchange HL, DE, and BC with their clones
1FAD  DD3405    INC     (IX+#05)        ; increase the barrels Y position ( move it down)
1FB0  DD7E17    LD      A,(IX+#17)      ; load A with the bottom Y location of the ladder we are on

; #6717 = bottom position of next ladder it is going down or the ladder it just passed.
; ladders bottoms are at :  70, 6A, 93, 8D, 8B, B3, B0, AC, D1, CD, F3, EE

1FB3  DDBE05    CP      (IX+#05)        ; check against item's Y position.  are we at the bottom of this ladder?
1FB6  C2CE1F    JP      NZ,#1FCE        ; no, jump ahead

; barrel reached bottom of ladder

1FB9  DD7E15    LD      A,(IX+#15)      ; load A with Barrel #15 indicator, zero = normal barrel,  1 = blue barrel
1FBC  07        RLCA                    ; roll left twice (multiply by 4)
1FBD  07        RLCA
1FBE  C615      ADD     A,#15           ; add #15
```

```
1FC0  DD7707    LD      (IX+#07),A      ; store into +7 indicator = sprite used

; #6707 - right 2 bits are 01 when rolling, 10 when being deployed.  bit 7 toggles as it rolls

1FC3  DD7E02    LD      A,(IX+#02)      ; load A with direction of barrel
1FC6  EE07      XOR     #07             ; XOR right 3 bits - reverses direction ?
1FC8  DD7702    LD      (IX+#02),A      ; store back in direction
1FCB  C3BA21    JP      #21BA           ; jump ahead

; we arrived here because we are not at the bottom of the ladder
; animates barrel as it rolls down ladder?

1FCE  DD7E0F    LD      A,(IX+#0F)      ; load A with barrel #0F counter (from 4 to 1)
1FD1  3D        DEC     A               ; decrement, has it reached 0?
1FD2  C2DF1F    JP      NZ,#1FDF        ; No, jump ahead, store into counter and continue on

; else animate the barrel

1FD5  DD7E07    LD      A,(IX+#07)      ; yes, Load A with #07 indicator = sprite used
1FD8  EE01      XOR     #01             ; toggle bit 1
1FDA  DD7707    LD      (IX+#07),A      ; store back in #07 indicator = toggle sprite
1FDD  3E04      LD      A,#04           ; A := 4

1FDF  DD770F    LD      (IX+#0F),A      ; store A into barrel #0F counter (from 4 to 1)
1FE2  C3BA21    JP      #21BA           ; jump ahead

; we arrived here because the barrel is moving to the right

1FE5  D9        EXX                     ; exchange HL, DE, and BC with their clones
1FE6  010001    LD      BC,#0100        ; BC := #0100
1FE9  DD3403    INC     (IX+#03)        ; Increase Barrel's X posiition
1FEC  C3F61F    JP      #1FF6           ; jump ahead

; we arrived here because the barrel is moving to the left

1FEF  D9        EXX                     ; exchange HL, DE, and BC with their clones
1FF0  0104FF    LD      BC,#FF04        ; load BC with #FF04
1FF3  DD3503    DEC     (IX+#03)        ; decrease barrel's X position

; we are here becuase the barrel is moving either left or right

1FF6  DD6603    LD      H,(IX+#03)      ; load H with barrel's X position
1FF9  DD6E05    LD      L,(IX+#05)      ; load L with barrel's Y position
1FFC  7C        LD      A,H             ; load A with barrel's X position
1FFD  E607      AND     #07             ; mask left 5 bits to zero.  result is between 0 and 7
1FFF  FE03      CP      #03             ; compare with #03
2001  CA5F21    JP      Z,#215F         ; equal to #03, jump ahead to check for ladders ?

2004  2D        DEC     L               ; otherwise decrease L 3 times
2005  2D        DEC     L
2006  2D        DEC     L
2007  CD3323    CALL    #2333           ; check for barrel going down a slanted girder ?
200A  2C        INC     L               ; increase L back to what it was
200B  2C        INC     L
200C  2C        INC     L
200D  7D        LD      A,L             ; Load A with Barrel's Y position
200E  DD7705    LD      (IX+#05),A      ; store back into barrel's y position
2011  CDDE23    CALL    #23DE           ;
2014  CDB424    CALL    #24B4           ;
2017  DD7E03    LD      A,(IX+#03)      ; Load A with Barrels' X position
201A  FE1C      CP      #1C             ; have we arrived at left edge of girder?
201C  DA2F20    JP      C,#202F         ; yes, jump ahead to handle

201F  FEE4      CP      #E4             ; else , have we arrived at right edge of girder?
2021  DABA21    JP      C,#21BA         ; no, jump way ahead - we're done, store values and try next barrel

; right edge of girder

2024  AF        XOR     A               ; A := 0
2025  DD7710    LD      (IX+#10),A      ; clear #10 barrel index to 0
2028  DD361160  LD      (IX+#11),#60    ; store #60 into barrel +#11  , indicates a roll over the right edge
202C  C33820    JP      #2038           ; skip next 3 steps

; arrive here when barrel at left edge of girder

202F  AF        XOR     A               ; A := 0
2030  DD3610FF  LD      (IX+#10),#FF    ; Set Barrel #10 index with #FF
2034  DD3611A0  LD      (IX+#11),#A0    ; set barrel #11 index with #A0 - indicates a roll over left edge

2038  DD3612FF  LD      (IX+#12),#FF    ;
203C  DD3613F0  LD      (IX+#13),#F0    ;
2040  DD7714    LD      (IX+#14),A      ;
2043  DD770E    LD      (IX+#0E),A      ; clear the barrel's edge indicator
2046  DD7704    LD      (IX+#04),A      ; clear ???
2049  DD7706    LD      (IX+#06),A      ;
204C  DD360208  LD      (IX+#02),#08    ; load barrel properties with various numbers to indicate edge roll?
2050  C3BA21    JP      #21BA           ; jump way ahead - we're done, store values and try next barrel

; jump from #1FA9
; we arrive here because the barrel isn't going left, right, or down a ladder
; could be crazy barrel or barrel going over edge

2053  D9        EXX                     ; Exchange DE, HL, BC with counterparts
2054  CD9C23    CALL    #239C           ; update barrel position ?
2057  CD2F2A    CALL    #2A2F           ; ???  set A to zero or 1 depending on ???
```

```
205A  A7        AND     A               ; iS A == 0 ?
205B  C28320    JP      NZ,#2083        ; no, jump ahead

205E  DD7E03    LD      A,(IX+#03)      ; load A with barrel X position
2061  C608      ADD     A,#08           ; Add #08
2063  FE10      CP      #10             ; compare with #10
2065  DA7920    JP      C,#2079         ; If carry, jump ahead, clear barrel, (rolled off screen?)

2068  CDB424    CALL    #24B4           ; check for barrel running into oil can?
206B  DD7E10    LD      A,(IX+#10)      ; load A with +10 = rolling over edge / direction indicator
206E  E601      AND     #01             ; mask all bits but 1.  result is 0 or 1
2070  07        RLCA                    ; rotate left
2071  07        RLCA                    ; rotate left again.  result is 0 or 4
2072  4F        LD      C,A             ; copy into C
2073  CDDE23    CALL    #23DE           ; ???
2076  C3BA21    JP      #21BA           ; skip ahead

2079  AF        XOR     A               ; A := 0
207A  DD7700    LD      (IX+#00),A      ; clear barrel active indicator
207D  DD7703    LD      (IX+#03),A      ; clear barrel X position
2080  C3BA21    JP      #21BA           ; done, store values and try next barrel

; barrel has landed on a new girder after going over edge, or has just done so and is bouncing

2083  DD340E    INC     (IX+#0E)        ; increase +E (???)
2086  DD7E0E    LD      A,(IX+#0E)      ; load A with this value
2089  3D        DEC     A               ; decrease.  zero? (did this barrel just land???)
208A  CAA220    JP      Z,#20A2         ; yes, skip ahead

208D  3D        DEC     A               ; else decrease again.  zero?
208E  CAC320    JP      Z,#20C3         ; yes, skip ahead

; barrel has finsished its edge maneuever

2091  DD7E10    LD      A,(IX+#10)      ; else load A with +10 = rolling over edge/direction indicator
2094  3D        DEC     A               ; decrease.  was this value a 1 ?  (barrel moving right)
2095  3E04      LD      A,#04           ; A := 4 = rolling left code
2097  C29C20    JP      NZ,#209C        ; no, skip next step

209A  3E02      LD      A,#02           ; else A := 2

209C  DD7702    LD      (IX+#02),A      ; store into motion indicator.  02 = rolling right, 08 = rolling down, 04 = rolling
left, bit 1 set when rolling down ladder
209F  C3BA21    JP      #21BA           ; jump ahead

; barrel has landed on a new girder after going over edge

20A2  DD7E15    LD      A,(IX+#15)      ; load A with Barrel #15 indicator, zero = normal barrel,  1 = blue barrel
20A5  A7        AND     A               ; is this a blue barrel?
20A6  C2B520    JP      NZ,#20B5        ; yes, skip ahead, blue barrels always continue all the way down

; normal barrel traversed edge

20A9  210562    LD      HL,#6205        ; load HL with mario's Y position address
20AC  DD7E05    LD      A,(IX+#05)      ; load A with +5 = barrel's Y position
20AF  D616      SUB     #16             ; subtract #16
20B1  BE        CP      (HL)            ; compare to mario Y position.  is the barrel below mario?
20B2  D2C320    JP      NC,#20C3        ; yes, skip next 5 steps

20B5  DD7E10    LD      A,(IX+#10)      ; load A with +10 = rolling over edge/direction indicator
20B8  A7        AND     A               ; A == 0 ? is this barrel is rolling right?
20B9  C2E120    JP      NZ,#20E1        ; no, skip ahead and set alternate values, continue at #20C3

20BC  DD7711    LD      (IX+#11),A      ; else set +11 (???) to zero
20BF  DD3610FF  LD      (IX+#10),#FF    ; set +10 = rolling over edge indicator to #FF for rolling left

; barrel has just finished bouncing after going around ledge

20C3  CD0724    CALL    #2407           ; ???
20C6  CB3C      SRL     H
20C8  CB1D      RR      L
20CA  CB3C      SRL     H
20CC  CB1D      RR      L
20CE  DD7412    LD      (IX+#12),H      ; store H into +#12 (???)
20D1  DD7513    LD      (IX+#13),L      ; store L into +#13 (???)
20D4  AF        XOR     A               ; A := 0
20D5  DD7714    LD      (IX+#14),A      ; clear +#14 (???)
20D8  DD7704    LD      (IX+#04),A      ; clear +#4 (???)
20DB  DD7706    LD      (IX+#06),A      ; clear +#6 (???)
20DE  C3BA21    JP      #21BA           ; skip ahead

20E1  DD361001  LD      (IX+#10),#01    ; set +10 = rolling over edge indicator to 1 for rolling right
20E5  DD361100  LD      (IX+#11),#00    ; set +11 = ??? to 0
20E9  C3C320    JP      #20C3           ; jump back

; we arrived here because its a crazy barrel from #1F97
; this is called for every pixel the barrel moves

20EC  D9        EXX                     ; exchange BC, DE, and HL with their alternates
20ED  CD9C23    CALL    #239C           ; update Barrel's variables ?. H now has +5 and L has +6
20F0  7C        LD      A,H             ; Load A with H = +5 = Y position
20F1  D61A      SUB     #1A             ; Subtract #1A (26 decimal)
20F3  DD4619    LD      B,(IX+#19)      ; load B with Barrel status #19 (?)
20F6  B8        CP      B               ; compare A with B
20F7  DA0421    JP      C,#2104         ; jump on carry ahead
```

```
20FA  CD2F2A    CALL    #2A2F           ; else call this sub (???)
20FD  A7        AND     A               ; is A == 0 ?
20FE  C21821    JP      NZ,#2118        ; No, jump ahead

2101  CDB424    CALL    #24B4           ; else call this sub (???)

2104  DD7E03    LD      A,(IX+#03)      ; load A with barrel X position
2107  C608      ADD     A,#08           ; add 8
2109  FE10      CP      #10             ; result < #10 ?
210B  D2CE1F    JP      NC,#1FCE        ; No, jump back and ???

210E  AF        XOR     A               ; yes, A := 0
210F  DD7700    LD      (IX+#00),A      ; set barrel status indicator #0 to 0 (barrel is gone)
2112  DD7703    LD      (IX+#03),A      ; set barrel x position to 0
2115  C3BA21    JP      #21BA           ; write to sprites and check next barrel

2118  DD7E05    LD      A,(IX+#05)      ; load A with barrel's Y position
211B  FEE0      CP      #E0             ; < #E0 ? - are we at bottom of screen?
211D  DA4621    JP      C,#2146         ; no, jump ahead

; else this crazy barrel is no longer crazy

2120  DD7E07    LD      A,(IX+#07)      ; else Load A with +7 = sprite used
2123  E6FC      AND     #FC             ; clear right 2 bits
2125  F601      OR      #01             ; turn on bit 0
2127  DD7707    LD      (IX+#07),A      ; store result
212A  AF        XOR     A               ; A := 0
212B  DD7701    LD      (IX+#01),A      ; barrel is no longer crazy
212E  DD7702    LD      (IX+#02),A      ;
2131  DD3610FF  LD      (IX+#10),#FF    ; set velocity to -1 (move left)
2135  DD7711    LD      (IX+#11),A      ;
2138  DD7712    LD      (IX+#12),A      ;
213B  DD3613B0  LD      (IX+#13),#B0    ;
213F  DD360E01  LD      (IX+#0E),#01    ;
2143  C35321    JP      #2153           ; jump ahead

; arrive here when crazy barrel hits a girder from #211D

2146  CD0724    CALL    #2407           ; load HL based on +14 status. also uses +11 and +12
2149  CDCB22    CALL    #22CB           ; do stuff for crazy barrels ?
214C  DD7E05    LD      A,(IX+#05)      ; load A with barrel Y position
214F  DD7719    LD      (IX+#19),A      ; store in barrel #19 status.  used for crazy barrels?
2152  AF        XOR     A               ; A := 0

2153  DD7714    LD      (IX+#14),A      ; clear +#14 (???)
2156  DD7704    LD      (IX+#04),A      ; clear +#4 (???)
2159  DD7706    LD      (IX+#06),A      ; store 0 in these barrel indicators
215C  C3BA21    JP      #21BA           ; jump ahead - we're done, store values and try next barrel

; arrive here every 8 pixels moved by barrel from #2001
; L has barrels Y pos
; H has barrels X pos

215F  7D        LD      A,L             ; load A with barrels Y position

2160  C605      ADD     A,#05           ; add 5
2162  57        LD      D,A             ; store into D
2163  7C        LD      A,H             ; load A with barrels X position
2164  011500    LD      BC,#0015        ; load BC with #15 to check for all ladders
2167  CD6D21    CALL    #216D           ; check for going down ladder
216A  C3BA21    JP      #21BA           ; skip ahead

; called from #2167

216D  CD6E23    CALL    #236E           ; check for ladder.  if no ladders, RET to higher sub.  if at top of ladder, A := 1
2170  3D        DEC     A               ; is there a ladder to go down?
2171  C0        RET     NZ              ; no, return

2172  78        LD      A,B             ; yes, load A with B which has the value of the ladder from the check ??
2173  D605      SUB     #05             ; subtract 5
2175  DD7717    LD      (IX+#17),A      ; store into +17 to indicate which ladder we might be going down ???
2178  3A4863    LD      A,(#6348)       ; get status of the oil can fire
217B  A7        AND     A               ; is the fire lit ?
217C  CAB221    JP      Z,#21B2         ; no, always take ladders before oil is lit

217F  3A0562    LD      A,(#6205)       ; else load A with mario's Y position + 5
2182  D604      SUB     #04             ; subtract 4
2184  BA        CP      D               ; is the barrel already below mario  ?
2185  D8        RET     C               ; yes, return without taking ladder

2186  3A8063    LD      A,(#6380)       ; else load A with difficulty from 1 to 5.  usually the level but increases during
play
2189  1F        RRA                     ; roll right (div 2) .  now can be 0, 1, or 2
218A  3C        INC     A               ; increment.  result is now 1, 2, or 3 based on skill level
218B  47        LD      B,A             ; store into B
218C  3A1860    LD      A,(RngTimer1)       ; load A with random timer ?
218F  4F        LD      C,A             ; store into C for later use ?
2190  E603      AND     #03             ; mask bits.   result now random number between 0 and 3
2192  B8        CP      B               ; compare with value computed above based on skill
2193  D0        RET     NC              ; return if greater.  on highest skill this works 75% of time, only returns on 3

2194  211060    LD      HL,InputState        ; load HL with player input.

; InputState - copy of RawInput, except when jump is pressed, bit 7 is set momentarily
```

```
; RawInput - right sets bit 0, left sets bit 1, up sets bit 2, down sets bit 3, jump sets bit 4

2197  3A0362    LD      A,(#6203)       ; load A with mario's x position
219A  BB        CP      E               ; compare with barrel's x position
219B  CAB221    JP      Z,#21B2         ; if equal, then go down ladder

219E  D2A921    JP      NC,#21A9        ; if barrel is to right of mario, then check for moving to left

21A1  CB46      BIT     0,(HL)          ; else is mario trying to move right ?
21A3  CAAE21    JP      Z,#21AE         ; no, skip ahead and return without going down ladder

21A6  C3B221    JP      #21B2           ; yes, make barrel go down ladder

21A9  CB4E      BIT     1,(HL)          ; is mario trying to move left ?
21AB  C2B221    JP      NZ,#21B2        ; yes, make barrel go down ladder

21AE  79        LD      A,C             ; else load A with random timer computed above
21AF  E618      AND     #18             ; mask with #18.    25% chance of being zero?
21B1  C0        RET     NZ              ; else return without going down ladder.  If zero then go down the ladder anyway

21B2  DD3407    INC     (IX+#07)        ; increase Barrel's deployment/animation status
21B5  DDCB02C6  SET     0,(IX+#02)      ; set barrel to go down the ladder
21B9  C9        RET                     ; return

; we arrive here because the barrel is rolling left or right or turning a corner or a crazy barrel
; stores position values, sprite value and colors into sprite values
; arrive from several locations, eg #20DE

21BA  D9        EXX                     ; swap DE, HL, and BC with counterparts
21BB  DD7E03    LD      A,(IX+#03)      ; load A with Barrels X position
21BE  77        LD      (HL),A          ; store into sprite X position
21BF  2C        INC     L               ; HL := HL + 1
21C0  DD7E07    LD      A,(IX+#07)      ; load A with Barrels deployment/animation status
21C3  77        LD      (HL),A          ; store into sprite value
21C4  2C        INC     L               ; HL := HL + 1
21C5  DD7E08    LD      A,(IX+#08)      ; load A with Barrel's color
21C8  77        LD      (HL),A          ; Store into sprite color
21C9  2C        INC     L               ; HL := HL + 1
21CA  DD7E05    LD      A,(IX+#05)      ; Load A with Barrel's Y position
21CD  77        LD      (HL),A          ; store into sprite Y position
21CE  C38D1F    JP      #1F8D           ; jump back and check for next barrel

; data used in sub below for attract mode movement
; first byte is movement, second is duration

21D1  80 FE     ; jump
21D3  01 C0     ; run right
21D5  04 50     ; up = climb ladder
21D7  02 10     ; run left
21D9  82 60     ; jump left
21DB  02 10     ; run left
21DD  82 CA     ; jump left
21DF  01 10     ; run right
21E1  81 FF     ; jump right (gets hammer)
21E3  02 38     ; run left
21E5  01 80     ; run right - mario dies falling over right edge
21E7  02 FF     ; run left
21E9  04 80     ; up
21EB  04 60     ; up
21ED  80        ; ?

; called during attract mode only from #1977

21EE  11D121    LD      DE,#21D1        ; load DE with start of table data
21F1  21CC63    LD      HL,#63CC        ; load HL with state of attract mode
21F4  7E        LD      A,(HL)          ; load A with state
21F5  07        RLCA                    ; rotate left (x2)
21F6  83        ADD     A,E             ; add to E to get the movement
21F7  5F        LD      E,A             ; put back
21F8  1A        LD      A,(DE)          ; load A with data from table
21F9  321060    LD      (InputState),A      ; store into copy of input
21FC  2C        INC     L               ; HL := #63CD (timer)
21FD  7E        LD      A,(HL)          ; load timer
21FE  35        DEC     (HL)            ; decrement
21FF  A7        AND     A               ; == #00 ?
2200  C0        RET     NZ              ; no, return

2201  1C        INC     E               ; else next movement
2202  1A        LD      A,(DE)          ; load A with timer from table
2203  77        LD      (HL),A          ; store into timer
2204  2D        DEC     L               ; HL := #63CC (state)
2205  34        INC     (HL)            ; increase state
2206  C9        RET                     ; return

; arrive here from main routine at #199B

2207  3E02      LD      A,#02           ; load A with 2 = 0010 binary
2209  F7        RST     #30             ; only continues here on conveyors, else returns from subroutine

220A  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
220D  1F        RRA                     ; time to do this ?
220E  218062    LD      HL,#6280        ; load HL with left side rectractable ladder
2211  7E        LD      A,(HL)          ; load A with ladder status
2212  DA1922    JP      C,#2219         ; if clock is odd, skip next 2 steps
```

```
2215  218862    LD      HL,#6288        ; load HL with right side retractable ladder
2218  7E        LD      A,(HL)          ; load A with ladder status

2219  E5        PUSH    HL              ; save HL
221A  EF        RST     #28             ; jump based on A

221B  27 22                             ; #2227         A = 0   ladder is all the way up
221D  59 22                             ; #2259         A = 1   ladder is moving down
221F  99 22                             ; #2299         A = 2   ladder is all the way down
2221  A2 22                             ; #22A2         A = 3   ladder is moving up
2223  00 00 00 00                       ; unused

; ladder is all the way up

2227  E1        POP     HL              ; restore HL - it has the ladder address
2228  2C        INC     L               ; HL := #6289 or #6281 - timer for movement ???
2229  35        DEC     (HL)            ; decrement.  at zero ?
222A  C23A22    JP      NZ,#223A        ; no, skip ahead and check to disable moving ladder indicator

222D  2D        DEC     L               ; put HL back where it was
222E  34        INC     (HL)            ; increase ladder status.  now it is moving down
222F  2C        INC     L               ;
2230  2C        INC     L               ; HL := #628A or #6282
2231  CD4322    CALL    #2243           ; only continue below if mario is on the ladder

2234  3E01      LD      A,#01           ; A := 1
2236  321A62    LD      (#621A),A       ; store into moving ladder indicator
2239  C9        RET                     ; return

223A  2C        INC     L               ; HL := #628A or #6282
223B  CD4322    CALL    #2243           ; only continue below if mario is on the ladder, else RET

223E  AF        XOR     A               ; A := 0
223F  321A62    LD      (#621A),A       ; store into moving ladder indicator
2242  C9        RET                     ; return

; called from #2231 above with HL = #628A
; called from #223B above with HL = #628A
; called from #2276 below

2243  3A0562    LD      A,(#6205)       ; load mario's Y position
2246  FE7A      CP      #7A             ; is mario on the top pie tray level or above?
2248  D25722    JP      NC,#2257        ; no, skip ahead and return to higher sub

224B  3A1662    LD      A,(#6216)       ; yes, check for a jump in progress ?
224E  A7        AND     A               ; is mario jumping ?
224F  C25722    JP      NZ,#2257        ; yes, jump ahead and return to higher sub

2252  3A0362    LD      A,(#6203)       ; else load A with mario's X position
2255  BE        CP      (HL)            ; is mario on the ladder? (or exactly lined up on it)
2256  C8        RET     Z               ; yes, return

2257  E1        POP     HL              ; adjust stack pointer
2258  C9        RET                     ; return to higher subroutine

; arrive from #221A when ladder is moving down

2259  E1        POP     HL              ; restore HL = ladder status
225A  2C        INC     L
225B  2C        INC     L
225C  2C        INC     L
225D  2C        INC     L               ; HL now has the ladder's ???
225E  35        DEC     (HL)            ; decrease.  at zero?
225F  C0        RET     NZ              ; no, return

2260  3E04      LD      A,#04           ; A := 4
2262  77        LD      (HL),A          ; store into the ladder's ???
2263  2D        DEC     L               ; HL now has the ladder's ???
2264  34        INC     (HL)            ; increase
2265  CDBD22    CALL    #22BD           ; ???
2268  3E78      LD      A,#78           ; A := #78
226A  BE        CP      (HL)            ; == (HL) ?
226B  C27522    JP      NZ,#2275        ; no, skip ahead

226E  2D        DEC     L
226F  2D        DEC     L
2270  2D        DEC     L
2271  34        INC     (HL)
2272  2C        INC     L
2273  2C        INC     L
2274  2C        INC     L

2275  2D        DEC     L               ; HL now has ???
2276  CD4322    CALL    #2243           ; only continue below if mario is on the ladder, else RET

; ladder is moving down and mario is on it

2279  3A0562    LD      A,(#6205)       ; load A with mario Y position
227C  FE68      CP      #68             ; is mario already at the low point of the ladder ?
227E  D28A22    JP      NC,#228A        ; yes, skip ahead

2281  210562    LD      HL,#6205        ; else load HL with Mario's Y position
2284  34        INC     (HL)            ; increase (move mario down one pixel)
2285  CDC03F    CALL    #3FC0           ; sets mario sprite to on ladder with left hand up and HL to #694F (mario's sprite Y
position) [this line seems like a patch ??? orig could be  LD HL,#694F ]
```

```
2288  34        INC     (HL)            ; increase sprite (move mario down one pixel in the hardware .  immediate update)
2289  C9        RET                     ; return

228A  1F        RRA                     ; rotate right A.  is A odd ?
228B  DA8122    JP      C,#2281         ; yes, loop back

228E  1F        RRA                     ; else rotate right A again.  is the 2-bit set ?
228F  3E01      LD      A,#01           ; A := 1
2291  DA9522    JP      C,#2295         ; yes, skip next step

2294  AF        XOR     A               ; A := 0
2295  322262    LD      (#6222),A       ; store into ladder toggle
2298  C9        RET                     ; return

; arrive from #221A when ladder is all the way down

2299  E1        POP     HL              ; restore HL
229A  3A1860    LD      A,(RngTimer1)       ; load A with random timer
229D  E63C      AND     #3C             ; mask bits.  result zero?
229F  C0        RET     NZ              ; no, return

22A0  34        INC     (HL)            ; else increase (HL) - the ladder is now moving up
22A1  C9        RET                     ; return

; arrive from jump at #221A
; a rectractable ladder is moving up
; HL popped from stack is either 6280 for left ladder or 6288 for right ladder

22A2  E1        POP     HL              ; restore HL
22A3  2C        INC     L
22A4  2C        INC     L
22A5  2C        INC     L
22A6  2C        INC     L               ; HL := HL + 4
22A7  35        DEC     (HL)            ; decrease (HL).  zero?
22A8  C0        RET     NZ              ; no, return

22A9  3602      LD      (HL),#02        ; else set (HL) to 2
22AB  2D        DEC     L               ;
22AC  35        DEC     (HL)            ; decrease ladder Y value - makes ladder move up
22AD  CDBD22    CALL    #22BD           ; update the sprite
22B0  3E68      LD      A,#68           ; A := #68
22B2  BE        CP      (HL)            ; reached top of ladder movement?
22B3  C0        RET     NZ              ; no, return

; ladder has moved all the way up

22B4  AF        XOR     A               ; A := 0
22B5  0680      LD      B,#80           ; B := #80
22B7  2D        DEC     L
22B8  2D        DEC     L
22B9  70        LD      (HL),B          ;
22BA  2D        DEC     L               ; set HL to ladder status
22BB  77        LD      (HL),A          ; set ladder status to 0 == all the way up
22BC  C9        RET                     ; return

; called from #22AD above and from #2265
; HL is preloaded with ladder Y position

22BD  7E        LD      A,(HL)          ; load A with ladder Y value
22BE  CB5D      BIT     3,L             ; test bit 3 of L
22C0  114B69    LD      DE,#694B        ; load DE with ladder sprite Y value
22C3  C2C922    JP      NZ,#22C9        ; if other ladder, skip next step

22C6  114769    LD      DE,#6947        ; load DE with other ladder sprite Y value
22C9  12        LD      (DE),A          ; update the sprite Y value
22CA  C9        RET                     ; return

; arrive here when crazy barrel is onscreen
; called when barrel deployed or hits a girder on the way down
; called from #2149

22CB  3A4863    LD      A,(#6348)       ; load A with oil can status
22CE  A7        AND     A               ; is the oil can lit ?
22CF  CAE122    JP      Z,#22E1         ; no , jump ahead

22D2  3A8063    LD      A,(#6380)       ; else load A with difficulty
22D5  3D        DEC     A               ; decrement.  will be between 0 and 4
22D6  EF        RST     #28             ; jump based on A

22D7  F6 22                             ; #22F6
22D9  F6 22                             ; #22F6
22DB  03 23                             ; #2303
22DD  03 23                             ; #2303
22DF  1A 23                             ; #231A

; arrive here when oil can is not yet lit
; used for initial crazy barrel

22E1  3A2962    LD      A,(#6229)       ; load A with level #
22E4  47        LD      B,A             ; store into B
22E5  05        DEC     B               ; decrement B
22E6  3E01      LD      A,#01           ; load A with 1
22E8  CAF922    JP      Z,#22F9         ; if level was 1, then jump ahead

22EB  05        DEC     B               ; decrement B again
```

```
22EC  3EB1      LD      A,#B1           ; load A with #B1 - for use with level 2 inital crazy barrel
22EE  CAF922    JP      Z,#22F9         ; if level 2, then jump ahead

22F1  3EE9      LD      A,#E9           ; else load A with #E9 - for level 3 and up inital crazy barrel
22F3  C3F922    JP      #22F9           ; jump ahead and store

; check for use with crazy barrels when difficulty is 1 or 2

22F6  3A1860    LD      A,(RngTimer1)       ; load A with random timer value

22F9  DD7711    LD      (IX+#11),A      ; store into +11
22FC  E601      AND     #01             ; mask bits, makes into #00 or #01
22FE  3D        DEC     A               ; decrement, now either #00 or #FF
22FF  DD7710    LD      (IX+#10),A      ; store into +10
2302  C9        RET                     ; return

; check for use with crazy barrels when difficulty is 3 or 4

2303  3A1860    LD      A,(RngTimer1)       ; load A with random timer value
2306  DD7711    LD      (IX+#11),A      ; store into +11
2309  3A0362    LD      A,(#6203)       ; load A with mario's X position
230C  DDBE03    CP      (IX+#03)        ; compare barrel's X position
230F  3E01      LD      A,#01           ; load A with 1
2311  D21623    JP      NC,#2316        ; if greater then skip ahead

2314  3D        DEC     A               ; else decrement twice
2315  3D        DEC     A               ; makes A := #FF

2316  DD7710    LD      (IX+#10),A      ; store into +10
2319  C9        RET                     ; return

; check for use with crazy barrels when difficulty is 5

231A  3A0362    LD      A,(#6203)       ; load A with mario's X position
231D  DD9603    SUB     (IX+#03)        ; subtract the barrel's X position
2320  0EFF      LD      C,#FF           ; load C with #FF
2322  DA2623    JP      C,#2326         ; if barrel is to left of mario, then jump ahead

2325  0C        INC     C               ; else increase C to 0

2326  07        RLCA                    ; rotate left A (doubles A)
2327  CB11      RL      C               ; rotate left C
2329  07        RLCA                    ; rotate left A (doubles A)
232A  CB11      RL      C               ; rotate left C
232C  DD7110    LD      (IX+#10),C      ; store C into +10
232F  DD7711    LD      (IX+#11),A      ; store A into +11
2332  C9        RET

; called from #2007 when barrels are rolling
; called from #     when mario is moving left or right on girders
; HL is preloaded with mario X,Y position
; B is preloaded with direction

2333  3E0F      LD      A,#0F           ; load A with binary 00001111
2335  A4        AND     H               ; and with H.  A now has between 0 and F
2336  05        DEC     B               ; Count down B.  is the direction == 1 ?
2337  CA4223    JP      Z,#2342         ; yes, then skip ahead 4 steps

233A  FE0F      CP      #0F             ; else check is A still = #0F ?
233C  D8        RET     C               ; return if Carry ( A < 0F ) most of time it wont?

233D  06FF      LD      B,#FF           ; else B := #FF
233F  C34723    JP      #2347           ; skip next 3 steps

2342  FE01      CP      #01             ; A > 1 ?
2344  D0        RET     NC              ; yes, return

2345  0601      LD      B,#01           ; B := 1

2347  3EF0      LD      A,#F0           ; A := #F0
2349  BD        CP      L               ; is A == L ?
234A  CA6023    JP      Z,#2360         ; Yes, skip ahead

234D  3E4C      LD      A,#4C           ; A := #4C
234F  BD        CP      L               ; == L ?
2350  CA6623    JP      Z,#2366         ; yes, skip ahead

2353  7D        LD      A,L
2354  CB6F      BIT     5,A
2356  CA5C23    JP      Z,#235C

2359  90        SUB     B
235A  6F        LD      L,A
235B  C9        RET                     ; return

235C  80        ADD     A,B             ; A := A + B
235D  C35A23    JP      #235A           ; loop back

2360  CB7C      BIT     7,H
2362  C25923    JP      NZ,#2359
2365  C9        RET                     ; return

2366  7C        LD      A,H             ; A := H
2367  FE98      CP      #98             ; < #98 ?
2369  D8        RET     C               ; no, return
```

```
236A  7D        LD      A,L             ; A := L
236B  C35C23    JP      #235C           ; loop back

; called from #1B13 when jumping ?
; called from #216D when checking for barrel to go down a ladder?
; A has X position of barrel ?
; BC starts with #15
; called when firefoxs are moving to check for ladders
; if no ladder is nearby , it RETs to a higher subroutine

236E  210063    LD      HL,#6300        ; load HL with start of table data that has positions of ladders
2371  EDB1      CPIR                    ; check for ladders ???

CPIR - The contents of the memory location addressed by the HL register pair is
compared with the contents of the Accumulator. In case of a true compare, a
condition bit is set. HL is incremented and the Byte Counter (register pair
BC) is decremented. If decrementing causes BC to go to zero or if A = (HL),
the instruction is terminated. If BC is not zero and A ? (HL), the program
counter is decremented by two and the instruction is repeated. Interrupts are
recognized and two refresh cycles are executed after each data transfer.
If BC is set to zero before instruction execution, the instruction loops
through 64 Kbytes if no match is found.



2373  C29A23    JP      NZ,#239A        ; if no match, return to higher sub, no ladder nearby

2376  E5        PUSH    HL              ; else a ladder may be near. save HL
2377  C5        PUSH    BC              ; save BC
2378  011400    LD      BC,#0014        ; load BC with #14 for offset
237B  09        ADD     HL,BC           ; add #14 to HL.  Now HL has the ladder's other value ?
237C  0C        INC     C               ; C := #15
237D  5F        LD      E,A             ; save A into E
237E  7A        LD      A,D             ; load A with D = barrels position ?
237F  BE        CP      (HL)            ; compare with ladder's position
2380  CA8F23    JP      Z,#238F         ; if equal then jump ahead

2383  09        ADD     HL,BC           ; else add #15 into HL
2384  BE        CP      (HL)            ; compare position
2385  CA9523    JP      Z,#2395         ; if equal then skip ahead

2388  57        LD      D,A             ; else load D with A
2389  7B        LD      A,E             ; load A with E
238A  C1        POP     BC              ; restore BC
238B  E1        POP     HL              ; restore HL
238C  C37123    JP      #2371           ; check for next ladder?

; arrive here when a barrel is above a ladder

238F  09        ADD     HL,BC           ; add #15 into HL
2390  3E01      LD      A,#01           ; load A with 1 = signal that we are at top of ladder
2392  C39823    JP      #2398           ; jump ahead

2395  AF        XOR     A               ; else A: = 0 = signal that we are at bottom of ladder
2396  ED42      SBC     HL,BC           ; subtract BC from HL.  restore HL to original value

2398  C1        POP     BC              ; restore BC
2399  46        LD      B,(HL)          ; load B with value in HL

239A  E1        POP     HL              ; restore HL
239B  C9        RET                     ; return

; called from #20ED for crazy barrel movement.  for this, BC, DE,and HL have their alternates
; subroutine called from #2054.  used when barrels are rolling.  only called when rolling around edges or mario jumping???
; IX has the start value of barrel sprite.  EG 6700
; IX can have 6200 for mario from #1BC2

239C  DD7E04    LD      A,(IX+#04)      ; load modified Y position, used for crazy barrels hitting girders ???
239F  DD8611    ADD     A,(IX+#11)      ; add +11 = vertical speed?
23A2  DD7704    LD      (IX+#04),A      ; update position ?

23A5  DD7E03    LD      A,(IX+#03)      ; load object's X position
23A8  DD8E10    ADC     A,(IX+#10)      ; add +10 = rolling over edge/direction indicator.  note this is add with carry
23AB  DD7703    LD      (IX+#03),A      ; store into X position

23AE  DD7E06    LD      A,(IX+#06)      ; load A with +6 == ??
23B1  DD9613    SUB     (IX+#13)        ; subtract +13 == ??
23B4  6F        LD      L,A             ; store into L
23B5  DD7E05    LD      A,(IX+#05)      ; load A with barrel Y position
23B8  DD9E12    SBC     A,(IX+#12)      ; subtract vertical speed????
23BB  67        LD      H,A             ; store into H
23BC  DD7E14    LD      A,(IX+#14)      ; load +14 = mirror of modified Y position?.  used for jump counter when mario jumps
23BF  A7        AND     A               ; clear flags
23C0  17        RLA                     ; rotate left (mult by 2)
23C1  3C        INC     A               ; add 1
23C2  0600      LD      B,#00           ; B := 0
23C4  CB10      RL      B               ;
23C6  CB27      SLA     A
23C8  CB10      RL      B
23CA  CB27      SLA     A
23CC  CB10      RL      B
23CE  CB27      SLA     A
23D0  CB10      RL      B
23D2  4F        LD      C,A             ; copy answer (A) to C. BC now has ???
```

```
23D3  09        ADD     HL,BC           ; add to HL
23D4  DD7405    LD      (IX+#05),H      ; update Y position
23D7  DD7506    LD      (IX+#06),L      ; update +6
23DA  DD3414    INC     (IX+#14)        ; increase +14.  used for 6214 for mario as a jump counter
23DD  C9        RET                     ; return

; called from subs that are moving a barrell left or right
; IX is memory base of the barrel in question (e.g. #6700)
; called from #2073 with C either 0 or 4
; C is preloaded with mask ?


23DE  DD7E0F    LD      A,(IX+#0F)      ; Load A with +#F property of barrel (counts from 4 to 1 over and over)
23E1  3D        DEC     A               ; decrease by one.  did counter go to zero?
23E2  C20324    JP      NZ,#2403        ; if not, jump ahead, store new timer value and return

23E5  AF        XOR     A               ; A := 0
23E6  DDCB0726  SLA     (IX+#07)        ; shift left the barrel sprite status, push bit 7 into carry flag
23EA  17        RLA                     ; rotate in carry flag into A
23EB  DDCB0826  SLA     (IX+#08)        ; shift left the other barrel color, push bit 7 into carry flag
23EF  17        RLA                     ; rotate in carry flag into A
23F0  47        LD      B,A             ; copy result into B
23F1  3E03      LD      A,#03           ; A := 3
23F3  B1        OR      C               ; bitwise OR with C
23F4  CD0930    CALL    #3009           ; ???
23F7  1F        RRA                     ;
23F8  DDCB081E  RR      (IX+#08)        ; rotate right the barrel's color
23FC  1F        RRA                     ;
23FD  DDCB071E  RR      (IX+#07)        ; Roll these values back
2401  3E04      LD      A,#04           ; A := 4

2403  DD770F    LD      (IX+#0F),A      ; store A into timer
2406  C9        RET                     ; return

;
; called from #1BDF and #20C3 and #2146
;

2407  DD7E14    LD      A,(IX+#14)      ; load A with Barrel +14 status
240A  07        RLCA
240B  07        RLCA
240C  07        RLCA
240D  07        RLCA                    ; rotate left 4 times
240E  4F        LD      C,A             ; save to C for use next 2 steps
240F  E60F      AND     #0F             ; mask with #0F.  now between #00 and #0F
2411  67        LD      H,A             ; store into H
2412  79        LD      A,C             ; restore A to value saved above
2413  E6F0      AND     #F0             ; mask with #F0
2415  6F        LD      L,A             ; store into L
2416  DD4E13    LD      C,(IX+#13)      ; load C with +13
2419  DD4612    LD      B,(IX+#12)      ; load B with +12
241C  ED42      SBC     HL,BC           ; HL := HL - BC
241E  C9        RET                     ; return

; arrive here when jump not pressed ?
; sets DE based on mario's position
; called from #1AE6
; called from #1BC5
; called from #2B09

241F  110001    LD      DE,#0100        ; DE:= #0100
2422  3A0362    LD      A,(#6203)       ; load A with Mario's X position
2425  FE16      CP      #16             ; is this greater than #16 ?
2427  D8        RET     C               ; yes, return

2428  15        DEC     D               ; no,
2429  1C        INC     E               ; DE := #0001
242A  FEEA      CP      #EA             ; is Mario's position > #EA ?
242C  D0        RET     NC              ; yes, return

242D  1D        DEC     E               ; no, DE:= #0000
242E  3A2762    LD      A,(#6227)       ; load A with screen number (01, 10, 11 or 100)
2431  0F        RRCA                    ; rotate right with carry.  is this the girders or elevators?
2432  d0        RET     nc              ; no, return

2433  3A0562    LD      A,(#6205)       ; otherwise load A with mario's Y position
2436  fe58      CP      #58             ; is this > #58 ?
2438  d0        RET     nc              ; Yes, return

2439  3A0362    LD      A,(#6203)       ; else load A with mario's X position
243C  FE6C      CP      #6C             ; is this > #6C ?
243E  D0        RET     NC              ; Yes, return

243F  14        INC     D               ; else DE := #0100
2440  C9        RET                     ; and return

; called from #0D62

; checksum ???

       ; 3F00:  5C 76 49 4A 01 09 08 01 3F 7D 77 1E 19 1E 24 15  .(C)1981...NINTE
       ; 3F10:  1E 14 1F 10 1F 16 10 11 1D 15 22 19 13 11 10 19  NDO.OF.AMERICA.I

; called from #0D62
; 1.  runs checksum on the NINTENDO, breaks if not correct
```

```
; 2.

2441  210C3F    LD      HL,#3F0C        ; load HL with ROM area that has NINTENDO written
2441  21103F    LD      HL,#3F10        ; load HL with adapted ROM area that has INTEND written used in copy protection
2444  3E5E      LD      A,#5E           ; A := #5E = constant so the checksum comes to zero
2446  0606      LD      B,#06           ; for B = 1 to 6
2448  86        ADD     A,(HL)          ; add this letter
2449  23        INC     HL              ; next letter
244A  10FC      DJNZ    #2448           ; loop until done
244C  FD211063  LD      IY,#6310        ;
2450  A7        AND     A               ; A == 0 ? checksum OK ?
2451  CA5624    JP      Z,#2456         ; yes, skip next step
2454  FD23      INC     IY              ; running this step will break the game ?  loops at #2371 forever
2456  3A2762    LD      A,(#6227)       ; load A with screen number
2459  3D        DEC     A               ; is this the girders?
245A  21E43A    LD      HL,#3AE4        ; load HL with start of table data for girders
245D  CA7124    JP      Z,#2471         ; if girders, skip ahead
245D  CAB80B    JP      Z,#0BB8         ; if girders, jump to additional code to activate the ladders
2460  3D        DEC     A               ; else is this the conveyors?
2461  215D3B    ld      HL,#3B5D        ; load HL with start of table data for conveyors
2461  21006B    LD      HL,#6B00        ; load HL with start of table with build ladder definitions
2464  CA7124    JP      Z,#2471         ; if conveyors, skip ahead
2467  3D        DEC     A               ; else is this the elevators?
2468  21E53B    LD      HL,#3BE5        ; load HL with start of table data for elevators
2468  21006B    LD      HL,#6B00        ; load HL with start of table with build ladder definitions
246B  CA7124    JP      Z,#2471         ; if elevators, skip ahead
246E  218B3C    LD      HL,#3C8B        ; otherwise we're on rivets.  load HL with table data for rivets
246E  21006B    LD      HL,#6B00        ; load HL with start of table with build ladder definitions
2471  DD210063  LD      IX,#6300        ; #6300 is used for ladder positions?
2475  110500    LD      DE,#0005        ; DE := 5 = offset
2478  7E        LD      A,(HL)          ; load A with the next item of data
2479  A7        AND     A               ; is this item == 0 ?
247A  CA8824    JP      Z,#2488         ; yes, jump ahead
247D  3D        DEC     A               ; no, decrease, was this item == 1 ?
247E  CA9E24    JP      Z,#249E         ; yes, jump down instead
2481  FEA9      CP      #A9             ; was the item == #AA ?
2483  C8        RET     Z               ; yes, return, we are done with this.  AA is at the end of each table
2484  19        ADD     HL,DE           ; if neither then add offset for next HL
2485  C37824    JP      #2478           ; loop again

; data element was #01

2488  23        INC     HL              ; next HL
2489  7E        LD      A,(HL)          ; load A with table data (EG #3B12)
248A  DD7700    LD      (IX+#00),A      ; store into index
248D  23        INC     HL              ; next HL
248E  7E        LD      A,(HL)          ; load A with table data
248F  DD7715    LD      (IX+#15),A      ; store into index +#15
2492  23        INC     HL              ;
2493  23        INC     HL              ; next HL, next HL
2494  7E        LD      A,(HL)          ; load A with table data
2495  DD772A    LD      (IX+#2A),A      ; store into index +#2A
2498  DD23      INC     IX              ; next location
249A  23        INC     HL              ; next table data
249B  C37824    JP      #2478           ; jump back

; data element was #02
; this sub is same as one above but uses IY instead of IX

249E  23        INC     HL
249F  7E        LD      A,(HL)
24A0  FD7700    LD      (IY+#00),A
24A3  23        INC     HL
24A4  7E        LD      A,(HL)
24A5  FD7715    LD      (IY+#15),A
24A8  23        INC     HL
24A9  23        INC     HL
24AA  7E        LD      A,(HL)
24AB  FD772A    LD      (IY+#2A),A
24AE  FD23      INC     IY
24B0  23        INC     HL
24B1  C37824    JP      #2478           ; jump back

; called this sub from barrel roll from #2068
; check for barrel collision with the oil can ????

24B4  DD7E05    LD      A,(IX+#05)      ; load A with Barrel Y position
24B7  FEE8      CP      #E8             ; Is it near the bottom or lower?
24B9  D8        RET     C               ; if so, return

24BA  DD7E03    LD      A,(IX+#03)      ; else load A with Barrel X position
24BD  FE2A      CP      #2A             ; is X position < #2A ? (rolling oever edge on left side of screen)
24BF  D0        RET     NC              ; no, return

24C0  FE20      CP      #20             ; is it past the edge of girder?
24C2  D8        RET     C               ; no, return

24C3  DD7E15    LD      A,(IX+#15)      ; load A with Barrel #15 indicator, zero = normal barrel,  1 = blue barrel
24C6  A7        AND     A               ; is this a normal barrel?
24C7  CAD024    JP      Z,#24D0         ; yes, jump ahead

24CA  3E03      LD      A,#03           ; else blue barrel, A := 3
24CC  32B962    LD      (#62B9),A       ; store into #62B9 - used for releasing fires ?
24CF  AF        XOR     A               ; A := #00
```

```
24D0  DD7700    LD      (IX+#00),A      ; clear out the barrel active indicator
24D3  DD7703    LD      (IX+#03),A      ; clear out the barrel X position
24D6  218260    LD      HL,#6082        ; load HL with boom sound address
24D9  3603      LD      (HL),#03        ; play boom sound for 3 units
24DB  E1        POP     HL              ; get HL from stack
24DC  3A4863    LD      A,(#6348)       ; turns to 1 when the oil can is on fire
24DF  A7        AND     A               ; is oil can already on fire ?
24E0  C2BA21    JP      NZ,#21BA        ; yes, jump back, we are done

24E3  3C        INC     A               ; else A := 1
24E4  324863    LD      (#6348),A       ; set the oil can is on fire
24E7  C3BA21    JP      #21BA           ; jump back , we are done.

; called from main routine at #1992
; copies pie buffer to pie sprites

24EA  3E02      LD      A,#02           ; check level for conveyors
24EC  F7        RST     #30             ; if not conveyors, RET, else continue
24ED  CD2325    CALL    #2523           ; check for deployment of new pies
24F0  CD9125    CALL    #2591           ; update all pies positions based on direction of trays, remove pies in fire or off
edge
24F3  DD21A065  LD      IX,#65A0        ; load IX with start of pies
24F7  0606      LD      B,#06           ; for B = 1 to 6 pies
24F9  21B869    LD      HL,#69B8        ; load HL with hardware address for pies

24FC  DD7E00    LD      A,(IX+#00)      ; load A with sprite status
24FF  A7        AND     A               ; is this sprite active ?
2500  CA1C25    JP      Z,#251C         ; no, add 4 to L and loop again

2503  DD7E03    LD      A,(IX+#03)      ; load A with pie X position
2506  77        LD      (HL),A          ; store into sprite
2507  2C        INC     L               ; next address
2508  DD7E07    LD      A,(IX+#07)      ; load A with pie sprite value
250B  77        LD      (HL),A          ; store into sprite
250C  2C        INC     L               ; next address
250D  DD7E08    LD      A,(IX+#08)      ; load A with pie color
2510  77        LD      (HL),A          ; store into sprite
2511  2C        INC     L               ; next address
2512  DD7E05    LD      A,(IX+#05)      ; load A with pie Y position
2515  77        LD      (HL),A          ; store into sprite
2516  2C        INC     L               ; next address

2517  DD19      ADD     IX,DE           ; add offset for next pie
2519  10E1      DJNZ    #24FC           ; next B

251B  C9        RET                     ; return

251C  7D        LD      A,L             ; A := L
251D  C604      ADD     A,#04           ; add 4
251F  6F        LD      L,A             ; store into L
2520  C31725    JP      #2517           ; loop back for next pie

; called from #24ED above

2523  219B63    LD      HL,#639B        ; load HL with pie timer
2526  7E        LD      A,(HL)          ; get timer value
2527  A7        AND     A               ; time to release a pie ?
2528  C28F25    JP      NZ,#258F        ; no, decrease counter and return

252B  3A9A63    LD      A,(#639A)       ; load A with fire deployment indicator ???
252E  A7        AND     A               ; == 0 ? (are there no fires???)
252F  C8        RET     Z               ; yes, return, no pies until fires are released

; look for a pie to deploy

2530  0606      LD      B,#06           ; for B = 1 to 6 pies
2532  111000    LD      DE,#0010        ; load DE with offset of #10 (16 decimal)
2535  DD21A065  LD      IX,#65A0        ; load IX with start of pie sprites table

2539  DDCB0046  BIT     0,(IX+#00)      ; is this pie already onscreen?
253D  CA4525    JP      Z,#2545         ; no, jump ahead and deploy this pie

2540  DD19      ADD     IX,DE           ; else load offset for next pie
2542  10F5      DJNZ    #2539           ; next B

2544  C9        RET                     ; return [no room for more pies, 6 already onscreen]

; deploy a pie

2545  CD5700    CALL    #0057           ; load A with a random number
2548  FE60      CP      #60             ; < #60 ?
254A  DD36057C  LD      (IX+#05),#7C    ; store #7C into pie's Y position
254E  DA5825    JP      C,#2558         ; yes, skip next 3 steps

2551  3AA362    LD      A,(#62A3)       ; load A with master direction for middle conveyor
2554  3D        DEC     A               ; is this tray moving outwards ?
2555  C26E25    JP      NZ,#256E        ; no, skip ahead

2558  DD3605CC  LD      (IX+#05),#CC    ; store #CC into pie's Y position
255C  3AA662    LD      A,(#62A6)       ; load A with master direction vector for lower conveyor
255F  07        RLCA                    ; is this tray moving to the right ?

2560  DD360307  LD      (IX+#03),#07    ; set pie X position to 7
2564  D27625    JP      NC,#2576        ; if tray moving right, skip ahead
```

```
2567  DD3603F8  LD      (IX+#03),#F8    ; set pie X position to #F8
256B  C37625    JP      #2576           ; skip ahead

256E  CD5700    CALL    #0057           ; load A with random number
2571  FE68      CP      #68             ; < #68 ?
2573  C36025    JP      #2560           ; use to decide to put on left or right side

2576  DD360001  LD      (IX+#00),#01    ; set pie active
257A  DD36074B  LD      (IX+#07),#4B    ; set pie sprite value
257E  DD360908  LD      (IX+#09),#08    ; set pie size??? (width?)
2582  DD360A03  LD      (IX+#0A),#03    ; set pie size??? (height?)
2586  3E7C      LD      A,#7C           ; A := #7C
2588  329B63    LD      (#639B),A       ; store into pie timer for next pie deployment
258B  AF        XOR     A               ; A := 0
258C  329A63    LD      (#639A),A       ; store into ???

258F  35        DEC     (HL)            ; decrease pie timer
2590  C9        RET                     ; return

; called from #24F0 above
; updates all pies

2591  DD21A065  LD      IX,#65A0        ; load IX with pie sprite buffer
2595  111000    LD      DE,#0010        ; load DE with offset
2598  0606      LD      B,#06           ; for B = 1 to 6

259A  DDCB0046  BIT     0,(IX+#00)      ; active ?
259E  CABB25    JP      Z,#25BB         ; no, skip ahead and loop for next

25A1  DD7E03    LD      A,(IX+#03)      ; load A with pie's X position
25A4  67        LD      H,A             ; copy to H
25A5  C607      ADD     A,#07           ; Add 7
25A7  FE0E      CP      #0E             ; < #E ? (pie < 6 or pie > #F9)
25A9  DAD625    JP      C,#25D6         ; yes, skip ahead to handle

25AC  DD7E05    LD      A,(IX+#05)      ; load A with pie Y position
25AF  FE7C      CP      #7C             ; is this the top level pie?
25B1  CAC025    JP      Z,#25C0         ; yes, skip ahead

25B4  3AA663    LD      A,(#63A6)       ; load A with pie direction vector for lower pie level
25B7  84        ADD     A,H             ; add vector to original position
25B8  DD7703    LD      (IX+#03),A      ; store into pie X position

25BB  DD19      ADD     IX,DE           ; add offset for next sprite
25BD  10DB      DJNZ    #259A           ; next B

25BF  C9        RET                     ; return

25C0  7C        LD      A,H             ; load A with pie X position
25C1  FE80      CP      #80             ; is the pie in the center fire?
25C3  CAD625    JP      Z,#25D6         ; yes, skip ahead

25C6  3AA563    LD      A,(#63A5)       ; load A with direction for upper left pie tray
25C9  D2CF25    JP      NC,#25CF        ; if pie < #80, use this address and skip next step

25CC  3AA463    LD      A,(#63A4)       ; else load A with direction for upper right tray

25CF  84        ADD     A,H             ; add vector to pie position
25D0  DD7703    LD      (IX+#03),A      ; store into pie X position
25D3  C3BB25    JP      #25BB           ; loop for next sprite

; pie in center fire or reached edge

25D6  21B869    LD      HL,#69B8        ; load HL with start of pie sprites
25D9  3E06      LD      A,#06           ; A := 6
25DB  90        SUB     B               ; subtract the pie number that is removed.  zero ?
25DC  CAE725    JP      Z,#25E7         ; yes, skip ahead

25DF  2C        INC     L
25E0  2C        INC     L
25E1  2C        INC     L
25E2  2C        INC     L               ; else HL := HL + 4
25E3  3D        DEC     A               ; decrease A
25E4  C3DC25    JP      #25DC           ; loop again

25E7  AF        XOR     A               ; A := 0
25E8  DD7700    LD      (IX+#00),A      ; clear pie active indicator
25EB  DD7703    LD      (IX+#03),A      ; clear pie X position
25EE  77        LD      (HL),A          ; clear sprite from screen
25EF  C3BB25    JP      #25BB           ; jump back and continue

; called from main routine at #19AA

25F2  3E02      LD      A,#02           ; load A with 2 = 0010 binary
25F4  F7        RST     #30             ; return if not conveyors

25F5  CD0226    CALL    #2602           ; handle top conveyor and pulleys
25F8  CD2F26    CALL    #262F           ; handle middle conveyor and pulleys
25FB  CD7926    CALL    #2679           ; handle lower conveyor and pulleys
25FE  CDD32A    CALL    #2AD3           ; handle mario's different speeds when on a conveyor
2601  C9        RET                     ; return

; called from #16D5, #25F5

2602  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
```

```
2605  0F        RRCA                    ; is the counter odd?
2606  DA1626    JP      C,#2616         ; yes, skip ahead

2609  21A062    LD      HL,#62A0        ; load HL with top conveyor counter
260C  35        DEC     (HL)            ; decrease.  time to reverse?
260D  C21626    JP      NZ,#2616        ; no, skip next 3 steps

2610  3680      LD      (HL),#80        ; reset counter
2612  2C        INC     L               ; HL := #62A1 = master direction vector for top tray
2613  CDDE26    CALL    #26DE           ; reverse the direction of this tray

2616  21A162    LD      HL,#62A1        ; load HL with master direction vector for top conveyor
2619  CDE926    CALL    #26E9           ; load A with direction vector for this frame
261C  32A363    LD      (#63A3),A       ; store A into direction vector for top conveyor
261F  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
2622  E61F      AND     #1F             ; mask bits
2624  FE01      CP      #01             ; == 1 ?
2626  C0        RET     NZ              ; no, return

2627  11E469    LD      DE,#69E4        ; else load DE with start of pulley sprites
262A  EB        EX      DE,HL           ; DE <> HL
262B  CDA626    CALL    #26A6           ; animate the pulleys
262E  C9        RET                     ; return

; called from #25F8 above

262F  21A362    LD      HL,#62A3        ; load HL with address of master direction vector for middle conveyor
2632  3A0562    LD      A,(#6205)       ; load A with mario's Y position
2635  FEC0      CP      #C0             ; is mario slightly above the lower conveyor?
2637  DA6F26    JP      C,#266F         ; yes, skip ahead.  in this case the upper trays don't vary

263A  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
263D  0F        RRCA                    ; roll right, is there a carry bit?
263E  DA4C26    JP      C,#264C         ; yes, skip ahead

2641  2D        DEC     L               ; load HL with middle conveyor counter
2642  35        DEC     (HL)            ; decrease it.  at zero?
2643  C24C26    JP      NZ,#264C        ; no, skip ahead

2646  36C0      LD      (HL),#C0        ; yes, reset the counter to #C0
2648  2C        INC     L               ; HL := #62A3 = master direction vector for middle conveyor
2649  CDDE26    CALL    #26DE           ; reverse the direction of this tray

264C  21A362    LD      HL,#62A3        ; load HL with master direction vector for upper left
264F  CDE926    CALL    #26E9           ; load A with direction vector for this frame
2652  32A563    LD      (#63A5),A       ; store into pie tray vector (upper right)
2655  ED44      NEG                     ; negate.  upper two pie trays move opposite directions
2657  32A463    LD      (#63A4),A       ; store into pie tray vector (upper left)
265A  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
265D  E61F      AND     #1F             ; mask bits, now between 0 and #1F.  zero?
265F  C0        RET     NZ              ; no, return

2660  2D        DEC     L               ; HL := #62A2 = middle conveyor counter
2661  11EC69    LD      DE,#69EC        ; load DE with middle pulley sprites
2664  EB        EX      DE,HL           ; DE <> HL
2665  CDA626    CALL    #26A6           ; animate the pulleys
2668  E67F      AND     #7F             ; mask bits, A now betwen #7F and 0 (turns off bit 7)
266A  21ED69    LD      HL,#69ED        ; load HL with ???
266D  77        LD      (HL),A          ; store A
266E  C9        RET                     ; return

266F  CB7E      BIT     7,(HL)          ; is this tray moving left ?
2671  C24C26    JP      NZ,#264C        ; yes, don't change anything

2674  36FF      LD      (HL),#FF        ; else change tray so it is moving left
2676  C34C26    JP      #264C           ; loop back to continue

; called from #25FB

2679  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
267C  0F        RRCA                    ; rotate right.  is there a carry?
267D  DA8D26    JP      C,#268D         ; yes, skip ahead

2680  21A562    LD      HL,#62A5        ; no, load HL with this counter
2683  35        DEC     (HL)            ; count it down.  zero?
2684  C28D26    JP      NZ,#268D        ; no, skip ahead

2687  36FF      LD      (HL),#FF        ; yes, reset counter to #FF
2689  2C        INC     L               ; HL := #62A6 = master direction vector for lower level
268A  CDDE26    CALL    #26DE           ; reverse direction of this tray

268D  21A662    LD      HL,#62A6        ; load HL with master direction vector for lower level
2690  CDE926    CALL    #26E9           ; load A with direction vector for this frame
2693  32A663    LD      (#63A6),A       ; store A into pie direction for lower level
2696  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
2699  E61F      AND     #1F             ; mask bits.  now between 0 and #1F
269B  FE02      CP      #02             ; == 2 ? (1/32 chance?)
269D  C0        RET     NZ              ; no, return

269E  11F469    LD      DE,#69F4        ; load DE with pulley sprite start
26A1  EB        EX      DE,HL           ; DE <> HL
26A2  CDA626    CALL    #26A6           ; call sub below to animate the pulleys [why?  it should just continue here]
26A5  C9        RET                     ; return

; called from #26A2, above with HL preloaded with pulley sprite address and DE preloaded with conveyor direction
```

```
; animates the pulleys

26A6  2C        INC     L               ; load HL with pulley sprite value
26A7  1A        LD      A,(DE)          ; load A with master conveyor direction
26A8  17        RLA                     ; rotate left.  carry set?
26A9  DAC526    JP      C,#26C5         ; yes, skip ahead to handle that direction

26AC  7E        LD      A,(HL)          ; load A with current sprite
26AD  3C        INC     A               ; increase it to animate
26AE  FE53      CP      #53             ; == #53 ? at end of sprite range?
26B0  C2B526    JP      NZ,#26B5        ; no, skip next step

26B3  3E50      LD      A,#50           ; A := #50 = reset sprite to first

26B5  77        LD      (HL),A          ; store result sprite
26B6  7D        LD      A,L             ; A := L = #E5
26B7  C604      ADD     A,#04           ; add 4 = #E9 for next sprite
26B9  6F        LD      L,A             ; HL now has next sprite
26BA  7E        LD      A,(HL)          ; load A with sprite value
26BB  3D        DEC     A               ; decrease to animate
26BC  FECF      CP      #CF             ; == #CF ? end of sprites?
26BE  C2C326    JP      NZ,#26C3        ; no, skip next step

26C1  3ED2      LD      A,#D2           ; A := #D2 = reset sprite to first

26C3  77        LD      (HL),A          ; store into sprite
26C4  C9        RET                     ; return

; from #26A9 when conveyor direction is other way

26C5  7E        LD      A,(HL)          ; load A with sprite value
26C6  3D        DEC     A               ; decrease to animate
26C7  FE4F      CP      #4F             ; == #4F ? end of sprites?
26C9  C2CE26    JP      NZ,#26CE        ; no, skip next step

26CC  3E52      LD      A,#52           ; A := #52 = first sprite

26CE  77        LD      (HL),A          ; store into sprite
26CF  7D        LD      A,L             ; A := L
26D0  C604      ADD     A,#04           ; add 4
26D2  6F        LD      L,A             ; L := A.  HL now has next sprite in set
26D3  7E        LD      A,(HL)          ; load A with sprite value
26D4  3C        INC     A               ; increase to animate
26D5  FED3      CP      #D3             ; == #D3? end of sprites?
26D7  C2DC26    JP      NZ,#26DC        ; no, skip next step

26DA  3ED0      LD      A,#D0           ; yes, A := #D0 = reset sprite to first

26DC  77        LD      (HL),A          ; store sprite
26DD  C9        RET                     ; return

; called from #268A with HL == #62A6 = master direction vector for lower level

26DE  CB7E      BIT     7,(HL)          ; is this direction moving right ?
26E0  CAE626    JP      Z,#26E6         ; yes, skip next 2 steps

26E3  3602      LD      (HL),#02        ; store 2 into (HL) - reverses the pie tray direction (now moving right)
26E5  C9        RET                     ; return

26E6  36FE      LD      (HL),#FE        ; store #FE into (HL) - reverses the pie tray direction (now moving left)
26E8  C9        RET                     ; return

; called when deciding which way to switch the pie tray direction vectors
; HL is preloaded with the master direction vector for the tray

26E9  3A1A60    LD      A,(FrameCounter)       ; load with clock counts down from #FF to 00 over and over...
26EC  E601      AND     #01             ; mask bits.  now either 0 or 1.  zero?
26EE  C8        RET     Z               ; yes, return.  every other frame the pie tray is stationary

26EF  CB7E      BIT     7,(HL)          ; check bit 7 of (HL) - this is the master direction for this tray
26F1  3EFF      LD      A,#FF           ; load A with vector for tray moving to left
26F3  C2F826    JP      NZ,#26F8        ; not zero, skip next step

26F6  3E01      LD      A,#01           ; load A with vector for tray moving to right
26F8  77        LD      (HL),A          ; store result
26F9  C9        RET                     ; return

; arrive here from main routine at #19A7

26FA  3E04      LD      A,#04           ; A := 4 = 0100 binary
26FC  F7        RST     #30             ; only continue here if elevators, else RET

; elevators only

26FD  3A0562    LD      A,(#6205)       ; load A with mario's Y position
2700  FEF0      CP      #F0             ; is mario too low ?
2702  D27F27    JP      NC,#277F        ; yes, then mario dead

2705  3A2962    LD      A,(#6229)       ; else load A with level number
2708  3D        DEC     A               ; decrement and check for zero
2709  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
270C  C21A27    JP      NZ,#271A        ; if level <> 1 then jump ahead

; slow elevators for level 1, japanese rom only?
```

```
270F  E603      AND     #03             ; mask bits of timer, now between 0 and 3
2711  FE01      CP      #01             ; == 1 ?
2713  CA1E27    JP      Z,#271E         ; yes, skip ahead and return

2716  DA2227    JP      C,#2722         ; if greater, then jump ahead and move elevators ?

2719  C9        RET                     ; else return

271A  0F        RRCA                    ; rotate right the timer
271B  DA2227    JP      C,#2722         ; if carry jump ahead and move the elevators (50% of time)

271E  CD4527    CALL    #2745           ; handle if mario is riding elevators
2721  C9        RET                     ; return

2722  CD9727    CALL    #2797           ; move elevators
2725  CDDA27    CALL    #27DA           ; check for and set elevators that have reset
2728  0606      LD      B,#06           ; For B = 1 to 6
272A  111000    LD      DE,#0010        ; load offset
272D  215869    LD      HL,#6958        ; load starting value for elevator sprites
2730  DD210066  LD      IX,#6600        ; memory where elevator values are stored

; update elevator sprites

2734  DD7E03    LD      A,(IX+#03)      ; load X position value for elevator
2737  77        LD      (HL),A          ; store into sprite value X position
2738  2C        INC     L
2739  2C        INC     L
273A  2C        INC     L               ; HL now has sprite Y value
273B  DD7E05    LD      A,(IX+#05)      ; load A with elevator Y position
273E  77        LD      (HL),A          ; store into sprite Y position
273F  2C        INC     L               ; next position
2740  DD19      ADD     IX,DE           ; next elevator
2742  10F0      DJNZ    #2734           ; Next B

2744  C9        RET                     ; return

; called from #271E

2745  3A9863    LD      A,(#6398)       ; load A with elevator riding indicator
2748  A7        AND     A               ; is mario riding an elevator?
2749  C8        RET     Z               ; no, return

274A  3A1662    LD      A,(#6216)       ; load A with jumping status
274D  A7        AND     A               ; is mario jumping ?
274E  C0        RET     NZ              ; yes, return

; arrive here when mario riding on either elevator

274F  3A0362    LD      A,(#6203)       ; load A with mario's X position. eg 37 for first, 75 for second
2752  FE2C      CP      #2C             ; position < left edge of first elevator ?
2754  DA6627    JP      C,#2766         ; yes, jump ahead

2757  FE43      CP      #43             ; else is position < right edge of first elevator ?
2759  DA6F27    JP      C,#276F         ; yes, jump ahead for first elevator checks

275C  FE6C      CP      #6C             ; else is position < left edge of second elevator?
275E  DA6627    JP      C,#2766         ; yes, jump ahead

2761  FE83      CP      #83             ; else is position < right edge of second elevator ?
2763  DA8727    JP      C,#2787         ; yes, jump ahead for second elevator checks

; arrive here when mario jumps off of an elevator ?

2766  AF        XOR     A               ; A := 0
2767  329863    LD      (#6398),A       ; clear elevator riding indicator
276A  3C        INC     A               ; A := 1
276B  322162    LD      (#6221),A       ; store into mario falling indicator ?
276E  C9        RET                     ; return

; arrive here when mario riding on first elevator

276F  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
2772  FE71      CP      #71             ; top of elevator ? (death)
2774  DA7F27    JP      C,#277F         ; yes, die

2777  3D        DEC     A               ; else decrement (move mario up)
2778  320562    LD      (#6205),A       ; store into Mario's Y position
277B  324F69    LD      (#694F),A       ; store into mario sprite Y value
277E  C9        RET                     ; return

277F  AF        XOR     A               ; A := 0
2780  320062    LD      (#6200),A       ; Make mario dead
2783  329863    LD      (#6398),A       ; clear elevator riding indicator
2786  C9        RET                     ; return

; riding on second elevator

2787  3A0562    LD      A,(#6205)       ; load A with mario's Y position
278A  FEE8      CP      #E8             ; at bottom of elevator ? (death)
278C  D27F27    JP      NC,#277F        ; yes, set death and return

278F  3C        INC     A               ; else increment (move mario down)
2790  320562    LD      (#6205),A       ; store back into mario's Y position
2793  324F69    LD      (#694F),A       ; store into mario sprite Y value
2796  C9        RET                     ; return
```

```
; called from #2722
; moves elevators ???

2797  0606      LD      B,#06           ; for B = 1 to 6 (for each elevator)
2799  111000    LD      DE,#0010        ; load DE with offset
279C  DD210066  LD      IX,#6600        ; load IX with start of sprite addr. for elevators

27A0  DDCB0046  BIT     0,(IX+#00)      ; is this elevator active?
27A4  CAC227    JP      Z,#27C2         ; no, skip ahead and loop for next

27A7  DDCB0D5E  BIT     3,(IX+#0D)      ; is this elevator moving down ?
27AB  CAC727    JP      Z,#27C7         ; yes, skip ahead

; elevator is moving up

27AE  DD7E05    LD      A,(IX+#05)      ; load A with elevator Y position
27B1  3D        DEC     A               ; decrement (move up)
27B2  DD7705    LD      (IX+#05),A      ; store result
27B5  FE60      CP      #60             ; at top of elevator ?
27B7  C2C227    JP      NZ,#27C2        ; no, skip next 2 steps

27BA  DD360377  LD      (IX+#03),#77    ; set X position to right side of elevators
27BE  DD360D04  LD      (IX+#0D),#04    ; set direction to down

27C2  DD19      ADD     IX,DE           ; add offset for next elevator
27C4  10DA      DJNZ    #27A0           ; next B
27C6  C9        RET                     ; return

; elevator is moving down

27C7  DD7E05    LD      A,(IX+#05)      ; load A with elevator Y position
27CA  3C        INC     A               ; increase (move down)
27CB  DD7705    LD      (IX+#05),A      ; store result
27CE  FEF8      CP      #F8             ; at bottom of shaft ?
27D0  C2C227    JP      NZ,#27C2        ; no, loop for next

27D3  DD360000  LD      (IX+#00),#00    ; yes, make this elevator inactive
27D7  C3C227    JP      #27C2           ; jump back and loop for next elevator

; called from #2725

; [IF elevator_counter <> 0 THEN ( elevator_counter--  ; RETURN ) ELSE (

27DA  21A762    LD      HL,#62A7        ; load HL with elevator counter address
27DD  7E        LD      A,(HL)          ; load A with elevator counter
27DE  A7        AND     A               ; == 0 ?
27DF  C20628    JP      NZ,#2806        ; no, skip ahead, decrease counter and return

27E2  0606      LD      B,#06           ; for B = 1 to 6 elevators
27E4  DD210066  LD      IX,#6600        ; load IX with sprite addr. for elevators

27E8  DDCB0046  BIT     0,(IX+#00)      ; is this elevator active ?
27EC  CAF427    JP      Z,#27F4         ; no, skip ahead and reset

27EF  DD19      ADD     IX,DE           ; add offset for next elevator
27F1  10F5      DJNZ    #27E8           ; next B
27F3  C9        RET                     ; return

27F4  DD360001  LD      (IX+#00),#01    ; make elevator active
27F8  DD360337  LD      (IX+#03),#37    ; set X position to left side shaft
27FC  DD3605F8  LD      (IX+#05),#F8    ; set Y position to bottom of shaft
2800  DD360D08  LD      (IX+#0D),#08    ; set direction to up
2804  3634      LD      (HL),#34        ; reset elevator counter to #34

2806  35        DEC     (HL)            ; decrease elevator counter
2807  C9        RET                     ; return

; called from main routine at #19B3
; checks for collisions with hostiles sprites

2808  FD210062  LD      IY,#6200        ; load IY with start of mario sprite
280C  3A0562    LD      A,(#6205)       ; load A with mario's Y position
280F  4F        LD      C,A             ; copy to C
2810  210704    LD      HL,#0407        ; H := 4, L := 7
2813  CD6F28    CALL    #286F           ; checks for collisions based on the screen.  A := 1 if collision, otherwise zero
2816  A7        AND     A               ; was there a collision ?
2817  C8        RET     Z               ; no, return

; mario collided with hostile sprite

2818  3D        DEC     A               ; else A := 0
2819  320062    LD      (#6200),A       ; store into mario life indicator, mario is dead
281C  C9        RET                     ; return

; called from main routine at #19B6

281D  0602      LD      B,#02           ; for B = 1 to 2 hammers
281F  111000    LD      DE,#0010        ; load DE with counter offset
2822  FD218066  LD      IY,#6680        ; load IY with sprite address start ?

2826  FDCB0146  BIT     0,(IY+#01)      ; is the hammer being used ?
282A  C23228    JP      NZ,#2832        ; yes, then do stuff ahead

282D  FD19      ADD     IY,DE           ; else look at next one
```

```
282F  10F5      DJNZ    #2826           ; next B

2831  C9        RET                     ; return

; hammer is active, do stuff for it

2832  FD4E05    LD      C,(IY+#05)      ; C := +5 (X position???)
2835  FD6609    LD      H,(IY+#09)      ; H := +9 (size?  width?)
2838  FD6E0A    LD      L,(IY+#0A)      ; L := +A (size?  height?)
283B  CD6F28    CALL    #286F           ; checks for collisions based on the screen.  A := 1 if collision, otherwise zero
283E  A7        AND     A               ; was there a collision?
283F  C8        RET     Z               ; no, return

; hammer hit something

2840  325063    LD      (#6350),A       ; store A into item hit indicator ???
2843  3AB963    LD      A,(#63B9)       ; load A with the number of total items checked for collision?
2846  90        SUB     B               ; subract the number of item hit ?
2847  325463    LD      (#6354),A       ; store into ???
284A  7B        LD      A,E             ; load A with offset for each item
284B  325363    LD      (#6353),A       ; store into ???
284E  DD225163  LD      (#6351),IX      ; store IX into ???
2852  C9        RET                     ; return

; called when mario jumping, checks for items being jumped over
; arrive at apex of jump
; called from #1C20

2853  FD210062  LD      IY,#6200        ; load IY with start of mario array
2857  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
285A  C60C      ADD     A,#0C           ; add #0C (12 decimal)
285C  4F        LD      C,A             ; copy to C
285D  3A1060    LD      A,(InputState)       ; load A with copy of input (see RawInput). except when jump pressed, bit 7 is
set momentarily.
2860  E603      AND     #03             ; mask bits, now between 0 and 3
2862  210805    LD      HL,#0508        ; H := #05, L := #08.  [H is the left-right window for jumping items, L is the up-
down window?]
2865  CA6B28    JP      Z,#286B         ; if masked input was zero, skip next step

; player moving joystick left or right while jumping

2868  210813    LD      HL,#1308        ; H := #13 (19 decimal) , L := #08. [ why is L set again ???]  [H is the left-right
window, increased if joystick moved left or right]

286B  CD883E    CALL    #3E88           ; check for items being jumped based on which screen this is [seems like a patch ?
what was original code? CALL #286F ?]
286E  C9        RET                     ; return

; called when hammer active from #283B - check for hammer collision with enemy sprites

286F  3A2762    LD      A,(#6227)       ; load A with screen number
2872  E5        PUSH    HL              ; save HL

2873  EF        RST     #28             ; jump to address below depending on screen:

2874  00 00                            ; unused
2876  80 28                             ; #2880 - girders
2878  B0 28                             ; #28B0 - conveyors
287A  E0 28                             ; #28E0 - elevators
287C  01 29                             ; #2901 - rivets
287E  00 00                             ; unused

; girders - check for collisions with barrels and fires and oil can

2880  E1        POP     HL              ; restore HL
2881  060A      LD      B,#0A           ; B := #0A (10 decimal).  one for each barrel
2883  78        LD      A,B             ; A := #0A
2884  32B963    LD      (#63B9),A       ; store counter for use later
2887  112000    LD      DE,#0020        ; load DE with offset of #20
288A  DD210067  LD      IX,#6700        ; load IX with start of barrels
288E  CD1329    CALL    #2913           ; check for collisions with barrels
2891  0605      LD      B,#05           ; B := 5
2893  78        LD      A,B             ; A := 5
2894  32B963    LD      (#63B9),A       ; store counter for use later
2897  1E20      LD      E,#20           ; E := #20
2899  DD210064  LD      IX,#6400        ; load IX with start of fires
289D  CD1329    CALL    #2913           ; check for collisions with fires
28A0  0601      LD      B,#01           ; B := 1
28A2  78        LD      A,B             ; A := 1
28A3  32B963    LD      (#63B9),A       ; store counter for use later
28A6  1E00      LD      E,#00           ; E := #00
28A8  DD21A066  LD      IX,#66A0        ; load IX with oil can fire location
28AC  CD1329    CALL    #2913           ; check for collision with oil can fire
28AF  C9        RET                     ; return

; jump here from #3E8C when jumping/hammering ? on the pie factory

28B0  E1        POP     HL              ; restore HL
28B1  0605      LD      B,#05           ; B := 5 fires
28B3  78        LD      A,B             ; A := 5 fires
28B4  32B963    LD      (#63B9),A       ; store counter for use later
28B7  112000    LD      DE,#0020        ; load DE with offset
28BA  DD210064  LD      IX,#6400        ; load IX with start of fires
28BE  CD1329    CALL    #2913           ; check for collisions with fires
28C1  0606      LD      B,#06           ; B := 6
```

```
28C3  78        LD      A,B             ; A := 6
28C4  32B963    LD      (#63B9),A       ; store counter for use later
28C7  1E10      LD      E,#10           ; E := #10
28C9  DD21A065  LD      IX,#65A0        ; load IX with start of pies
28CD  CD1329    CALL    #2913           ; check for collisions with pies
28D0  0601      LD      B,#01           ; B := 1
28D2  78        LD      A,B             ; A := 1
28D3  32B963    LD      (#63B9),A       ; store counter for use later
28D6  1E00      LD      E,#00           ; E := 0
28D8  DD21A066  LD      IX,#66A0        ; load IX with oil can address
28DC  CD1329    CALL    #2913           ; check for collision with oil can fire
28DF  C9        RET                     ; return

; jump here from #2873 or #3E8C when on the elevators

28E0  E1        POP     HL              ; restore HL
28E1  0605      LD      B,#05           ; B := 5
28E3  78        LD      A,B             ; A := 5
28E4  32B963    LD      (#63B9),A       ; store counter for use later
28E7  112000    LD      DE,#0020        ; load offset
28EA  DD210064  LD      IX,#6400        ; load start of addresses for fires
28EE  CD1329    CALL    #2913           ; check for collisions with fires
28F1  060A      LD      B,#0A           ; B := #0A
28F3  78        LD      A,B             ; A := #0A
28F4  32B963    LD      (#63B9),A       ; store counter for use later
28F7  1E10      LD      E,#10           ; E := #10
28F9  DD210065  LD      IX,#6500        ; load IX with start of addresses for springs
28FD  CD1329    CALL    #2913           ; check for collisions with springs
2900  C9        RET                     ; return

; jump here from #3E8C when on the rivets
; check for collisions with firefoxes and squares next to kong

2901  E1        POP     HL              ; restore HL
2902  0607      LD      B,#07           ; B := 7
2904  78        LD      A,B             ; A := 7
2905  32B963    LD      (#63B9),A       ; store 7 into counter for use later
2908  112000    LD      DE,#0020        ; load DE with offset
290B  DD210064  LD      IX,#6400        ; load IX with start of firefox arrays
290F  CD1329    CALL    #2913           ; check for collisions with firefoxes/squares
2912  C9        RET                     ; return

; core routine gets called a lot
; uses IX and DE and IY
; uses B for loop counter
; uses C for a memory location start
; HL are used
; seems to return a value in A as either 0 or 1
; check for sprite collision ???


2913  DDE5      PUSH    IX              ; push IX to stack

; start of loop

2915  DDCB0046  BIT     0,(IX+#00)      ; is this sprite active?
2919  CA4C29    JP      Z,#294C         ; no, add offset in DE and loop again

291C  79        LD      A,C             ; no, load A with C
291D  DD9605    SUB     (IX+#05)        ; subtract the Y value of item 2
2920  D22529    JP      NC,#2925        ; if no carry, skip next step

2923  ED44      NEG                     ; A = 0 - A (negate with 2's complement)

2925  3C        INC     A               ; A := A + 1
2926  95        SUB     L               ; subtract L [???]
2927  DA3029    JP      C,#2930         ; on carry, skip next 2 steps

292A  DD960A    SUB     (IX+#0A)        ; subtract +#0A value height???
292D  D24C29    JP      NC,#294C        ; if no carry, add offset in DE and loop again

2930  FD7E03    LD      A,(IY+#03)      ; load A with X position of item 1
2933  DD9603    SUB     (IX+#03)        ; subtract X position of item 2.  carry?
2936  D23B29    JP      NC,#293B        ; no, skip next step

2939  ED44      NEG                     ; A = 0 - A (negate with 2's complement)

293B  94        SUB     H               ; subtract H
293C  DA4529    JP      C,#2945         ; on carry, skip next 2 steps

293F  DD9609    SUB     (IX+#09)        ; subtract +#09 value width???
2942  D24C29    JP      NC,#294C        ; if no carry, add offset in DE and loop again

; else a collision

2945  3E01      LD      A,#01           ; A := 1 - code for collision
2947  DDE1      POP     IX              ; restore IX
2949  33        INC     SP
294A  33        INC     SP              ; adjust SP for higher level subroutine
294B  C9        RET                     ; return to higher subroutine

294C  DD19      ADD     IX,DE           ; add offset for next sprite
294E  10C5      DJNZ    #2915           ; Next B

2950  AF        XOR     A               ; A := 0 - code for no collision
```

```
2951  DDE1      POP     IX              ; restore IX
2953  C9        RET                     ; return

; arrive here when jumping at top of jump, check for hammer grab

2954  3E0B      LD      A,#0B           ; A := #0B = 1011 binary
2956  F7        RST     #30             ; if level is elevators RET from this sub now.  no hammers on elevators.
2957  CD7429    CALL    #2974           ; load A with 1 if hammer is grabbed, 0 if no grab
295A  321862    LD      (#6218),A       ; store into hammer grabbing indicator
295D  0F        RRCA
295E  0F        RRCA                    ; rotate right twice.  if hammer grabbed, A is now #40
295F  328560    LD      (#6085),A       ; play sound for bonus
2962  78        LD      A,B             ; A := B .  this indicates which hammer was grabbed if any
2963  A7        AND     A               ; was a hammer grabbed?
2964  C8        RET     Z               ; no, return

2965  FE01      CP      #01             ; was lower hammer on girders & conveyors, or upper hammer on rivets, grabbed?
2967  CA6F29    JP      Z,#296F         ; yes, skip next 2 steps

296A  DD360101  LD      (IX+#01),#01    ; set 1st hammer active
296E  C9        RET                     ; return

296F  DD361101  LD      (IX+#11),#01    ; set 2nd hammer active
2973  C9        RET                     ; return

; called from #2957 above
; check for hammer grab ?

2974  FD210062  LD      IY,#6200        ; load IY with start of mario sprite values
2978  3A0562    LD      A,(#6205)       ; load A with mario's Y position
297B  4F        LD      C,A             ; copy to C
297C  210804    LD      HL,#0408        ; H := 4, L := 8
297F  0602      LD      B,#02           ; B := 2 for the 2 hammers (?)
2981  111000    LD      DE,#0010        ; offset for each hammer
2984  DD218066  LD      IX,#6680        ; load IX with start of hammer sprites ?
2988  CD1329    CALL    #2913           ; check for collision with hammer
298B  C9        RET                     ; return

; called from #323E
; fire moving.  check for girder edge near fire
; sets A := 0 if fire is free to move
; sets A := 1 if fire is next to edge of girder

298C  2AC863    LD      HL,(#63C8)      ; load HL with address of this fire
298F  7D        LD      A,L             ; A := L
2990  C60E      ADD     A,#0E           ; add #E
2992  6F        LD      L,A             ; store result.  HL now has the fire's X position
2993  56        LD      D,(HL)          ; load D with the fire's X position
2994  2C        INC     L               ; next HL = fire's Y position
2995  7E        LD      A,(HL)          ; load A with the fire's Y position
2996  C60C      ADD     A,#0C           ; add #C to offset
2998  5F        LD      E,A             ; store into E
2999  EB        EX      DE,HL           ; DE <> HL
299A  CDF02F    CALL    #2FF0           ; convert HL into VRAM memory location
299D  7E        LD      A,(HL)          ; load A with the screen element at this location
299E  FEB0      CP      #B0             ; > #B0 ?
29A0  DAAC29    JP      C,#29AC         ; yes, skip next 5 steps, set A := 1 and return

29A3  E60F      AND     #0F             ; else mask bits, now between 0 and #F
29A5  FE08      CP      #08             ; <= 8 ?
29A7  D2AC29    JP      NC,#29AC        ; yes, skip next 2 steps, set A := 1 and return

29AA  AF        XOR     A               ; A := 0 = clear signal
29AB  C9        RET                     ; return

29AC  3E01      LD      A,#01           ; A := 1 = fire near girder edge
29AE  C9        RET                     ; return

; called from #2B23 during a jump

29AF  3E04      LD      A,#04           ; A := 4 = 0100
29B1  F7        RST     #30             ; only continue here if we are on the elevators, else RET

29B2  FD210062  LD      IY,#6200        ; load IY with mario's array
29B6  3A0562    LD      A,(#6205)       ; load A with mario's Y position
29B9  4F        LD      C,A             ; copy to C
29BA  210804    LD      HL,#0408        ; H := 4, L := 8
29BD  CD222A    CALL    #2A22           ; check for collision with elevators
29C0  A7        AND     A               ; was there a collision?
29C1  CA202A    JP      Z,#2A20         ; no, load B with #00 and return

; arrive here when landing near an elevator
; B has the index of the elevator that we hit

29C4  3E06      LD      A,#06           ; A := 6
29C6  90        SUB     B               ; subtract B.  zero ?
29C7  CAD029    JP      Z,#29D0         ; yes, skip ahead

29CA  DD19      ADD     IX,DE           ; else add offset for next elevator
29CC  3D        DEC     A               ; decrease counter
29CD  C3C729    JP      #29C7           ; loop again

; IX now has the array start for the elevator mario trying to land on

29D0  DD7E05    LD      A,(IX+#05)      ; load A with elevator's height Y position
```

```
29D3  D604      SUB     #04             ; subtract 4
29D5  57        LD      D,A             ; copy to D
29D6  3A0C62    LD      A,(#620C)       ; load A with mario's jump height ?
29D9  C605      ADD     A,#05           ; add 5
29DB  BA        CP      D               ; compare.  is mario high enough to land ?
29DC  D2EE29    JP      NC,#29EE        ; no, skip ahead

29DF  7A        LD      A,D             ; load A with elevator's height - 4
29E0  D608      SUB     #08             ; subtract 8
29E2  320562    LD      (#6205),A       ; store A into Mario's Y position
29E5  3E01      LD      A,#01           ; A := 1
29E7  47        LD      B,A             ; B := 1
29E8  329863    LD      (#6398),A       ; set elevator riding indicator ?
29EB  33        INC     SP
29EC  33        INC     SP              ; increase SP twice so the RET skips one level
29ED  C9        RET                     ; returns to higher subroutine (#1C08)

29EE  3A0C62    LD      A,(#620C)       ; load A with mario's jump height
29F1  D60E      SUB     #0E             ; subtract #0E (14 decimal)
29F3  BA        CP      D               ; compare to elevator height - 4. is mario hitting his head on the bottom of the
elevator ?
29F4  D21B2A    JP      NC,#2A1B        ; if so, mario is dead.  set dead and return.

29F7  3A1062    LD      A,(#6210)       ; load A with mario's jump direction.
29FA  A7        AND     A               ; == 0 ?  Is mario jumping to the right ?
29FB  3A0362    LD      A,(#6203)       ; load A with mario's X position
29FE  CA082A    JP      Z,#2A08         ; if jumping to the right then skip ahead

2A01  F607      OR      #07             ; else mask bits, turn on all 3 lower bits
2A03  D604      SUB     #04             ; subtract 4
2A05  C30E2A    JP      #2A0E           ; skip next 3 steps

2A08  D608      SUB     #08             ; subtract 8
2A0A  F607      OR      #07             ; turn on all 3 lower bits
2A0C  C604      ADD     A,#04           ; add 4

; used when riding an elevator

2A0E  320362    LD      (#6203),A       ; set mario's X position
2A11  324C69    LD      (#694C),A       ; set mario's sprite X position
2A14  3E01      LD      A,#01           ; A := 1
2A16  0600      LD      B,#00           ; B := 0
2A18  33        INC     SP
2A19  33        INC     SP              ; set stack to next higher subroutine return
2A1A  C9        RET                     ; return to higher level (#1C08)

; arrive from #29F4 when mario dies trying to jump onto elevator

2A1B  AF        XOR     A               ; A := 0
2A1C  320062    LD      (#6200),A       ; set mario dead
2A1F  C9        RET                     ; return

; arrive from #29C1

2A20  47        LD      B,A             ; B := 0
2A21  C9        RET                     ; return

; called from #29BD

2A22  0606      LD      B,#06           ; B := 6
2A24  111000    LD      DE,#0010        ; load DE with offset
2A27  DD210066  LD      IX,#6600        ; load IX with elevator array start
2A2B  CD1329    CALL    #2913           ; check for collision with elevators
2A2E  C9        RET                     ; return

; sub called during a barrel roll from #2057
; only called when barrel going over edge to next girder or for crazy barrel ?
; returns with A loaded with 0 or 1 depending on ???

2A2F  DD7E03    LD      A,(IX+#03)      ; load A with Barrel's X position
2A32  67        LD      H,A             ; Store into H
2A33  DD7E05    LD      A,(IX+#05)      ; load A with Barrel's Y position
2A36  C604      ADD     A,#04           ; Add 4
2A38  6F        LD      L,A             ; Store in L
2A39  E5        PUSH    HL              ; Save HL to stack
2A3A  CDF02F    CALL    #2FF0           ; convert HL into VRAM memory address
2A3D  D1        POP     DE              ; load DE with HL = barrel position X,Y
2A3E  7E        LD      A,(HL)          ; load A with the graphic at this location


B0 = Girder with hole in center used in rivets screen
B6 = white line on top
B7 = wierd icon?
B8 = red line on bottom
C0 - C7 = girder with ladder on bottom going up
D0 - D7 = ladder graphic with girder under going up and out
DD = HE  (help graphic)
DE = EL
DF = P!
E1 - E7 = grider graphic going up and out
EC - E8 = blank ?
EF = P!
EE = EL (part of help graphic)
ED = HE (help graphic)
F6 - F0 = girder graphic in several vertical phases coming up from bottom
```

```
F7 = bottom yellow line
FA - F8 = blank ?
FB = ? (actually a question mark)
FC = right red edge
FD = left red edge
FE = X graphic
FF = Extra Mario Icon


2A3F  FEB0      CP      #B0             ; < #B0 ?
2A41  DA7B2A    JP      C,#2A7B         ; yes, skip ahead,  clear A to 0 and return - nothing to do.

2A44  E60F      AND     #0F             ; mask bits.  now between 0 and #F
2A46  FE08      CP      #08             ; < 8 ?
2A48  D27B2A    JP      NC,#2A7B        ; no, skip ahead, clear A to 0 and return - nothing to do.

2A4B  7E        LD      A,(HL)          ; load A with graphic at this location
2A4C  FEC0      CP      #C0             ; == girder with ladder on bottom going up ?
2A4E  CA7B2A    JP      Z,#2A7B         ; yes, clear A to 0 and return - nothing to do.

2A51  DA692A    JP      C,#2A69         ; < this value ?  if so, skip ahead

2A54  FED0      CP      #D0             ; > ladder graphic with girder under going up and out ?
2A56  DA6E2A    JP      C,#2A6E         ; yes, skip ahead to handle

2A59  FEE0      CP      #E0             ; > grider graphic going up and out ?
2A5B  DA632A    JP      C,#2A63         ; yes, skip next 2 steps

2A5E  FEF0      CP      #F0             ; > girder graphic in several vertical phases coming up from bottom ?
2A60  DA6E2A    JP      C,#2A6E         ; yes, skip ahaed to handle

; arrive when crazy barrel hitting top of girder ?

2A63  E60F      AND     #0F             ; mask bits, now between 0 and #F
2A65  3D        DEC     A               ; decrease
2A66  C3722A    JP      #2A72           ; skip ahead

; arrive when ???

2A69  3EFF      LD      A,#FF           ; A := #FF
2A6B  C3722A    JP      #2A72           ; skip next 2 steps

; arrive when ???

2A6E  E60F      AND     #0F             ; mask bits, now between 0 and #F
2A70  D609      SUB     #09             ; subtract 9

; other conditions all arrive here
; A is loaded with a number between #F6 and #E

2A72  4F        LD      C,A             ; C := A
2A73  7B        LD      A,E             ; A := E = barrel X position
2A74  E6F8      AND     #F8             ; mask bits.  lower 3 bits are cleared
2A76  81        ADD     A,C             ; add C
2A77  BB        CP      E               ; compare to barrel's X position.  less?
2A78  DA7D2A    JP      C,#2A7D         ; yes, skip next 2 steps

2A7B  AF        XOR     A               ; A := 0
2A7C  C9        RET                     ; return

2A7D  D604      SUB     #04             ; subtract 4
2A7F  DD7705    LD      (IX+#05),A      ; store A into Y position
2A82  3E01      LD      A,#01           ; A := 1
2A84  C9        RET                     ; return

; called from main routine at #19A1

2A85  3A1562    LD      A,(#6215)       ; load ladder status
2A88  A7        AND     A               ; is mario on a ladder ?
2A89  C0        RET     NZ              ; yes, return

2A8A  3A1662    LD      A,(#6216)       ; load jumping status
2A8D  A7        AND     A               ; is mario jumping ?
2A8E  C0        RET     NZ              ; yes, return

2A8F  3A9863    LD      A,(#6398)       ; load A with elevator status
2A92  FE01      CP      #01             ; is mario riding an elevator?
2A94  C8        RET     Z               ; yes, return

2A95  3A0362    LD      A,(#6203)       ; load A with mario's X position
2A98  D603      SUB     #03             ; subtract 3
2A9A  67        LD      H,A             ; store into H
2A9B  3A0562    LD      A,(#6205)       ; load A with Mario's Y position
2A9E  C60C      ADD     A,#0C           ; add #0C = 13 decimal
2AA0  6F        LD      L,A             ; store into L
2AA1  E5        PUSH    HL              ; save to stack
2AA2  CDF02F    CALL    #2FF0           ; load HL with screen position of mario's feet
2AA5  D1        POP     DE              ; restore , DE now has the sprite X,Y addresses
2AA6  7E        LD      A,(HL)          ; load A with the screen item at mario's feet
2AA7  FEB0      CP      #B0             ; > #B0 ?
2AA9  DAB42A    JP      C,#2AB4         ; yes, skip next 4 steps

2AAC  E60F      AND     #0F             ; else mask bits, now between 0 and #F
2AAE  FE08      CP      #08             ; > 8 ?
2AB0  D2B42A    JP      NC,#2AB4        ; no, skip next step
```

```
2AB3  C9        RET                     ; else return

; arrive when mario near an [left?] edge

2AB4  7A        LD      A,D             ; load A with mario's X position
2AB5  E607      AND     #07             ; mask bits, now between 0 and 7.  zero?
2AB7  CACD2A    JP      Z,#2ACD         ; yes, skip ahead, mario is falling

2ABA  012000    LD      BC,#0020        ; BC := 20
2ABD  ED42      SBC     HL,BC           ; subtract from HL.  now HL is the next column?
2ABF  7E        LD      A,(HL)          ; load A with the screen element of this location
2AC0  FEB0      CP      #B0             ; > #B0 ?
2AC2  DACD2A    JP      C,#2ACD         ; yes, skip ahead, mario is falling

2AC5  E60F      AND     #0F             ; else mask bits, now betwen 0 and F
2AC7  FE08      CP      #08             ; > 8 ?
2AC9  D2CD2A    JP      NC,#2ACD        ; no, mario is falling, skip ahead
2ACC  C9        RET                     ; return

; mario is falling

2ACD  3E01      LD      A,#01           ; A := 1
2ACF  322162    LD      (#6221),A       ; store into mario falling indicator
2AD2  C9        RET                     ; return

; called from #25FE

2AD3  3A0362    LD      A,(#6203)       ; load A with mario's X position
2AD6  47        LD      B,A             ; copy to B
2AD7  3A0562    LD      A,(#6205)       ; load A with mario's Y position
2ADA  FE50      CP      #50             ; is mario on upper level ?
2ADC  CAEA2A    JP      Z,#2AEA         ; yes, skip ahead

2ADF  FE78      CP      #78             ; mario on upper pie tray?
2AE1  CAF62A    JP      Z,#2AF6         ; yes, skip ahead

2AE4  FEC8      CP      #C8             ; mario on lower pie tray ?
2AE6  CAF02A    JP      Z,#2AF0         ; yes, skip ahead

2AE9  C9        RET                     ; else return

2AEA  3AA363    LD      A,(#63A3)       ; load A with top conveyor direction vector [why?  level complete here?]
2AED  C3022B    JP      #2B02           ; skip ahead

2AF0  3AA663    LD      A,(#63A6)       ; load A with pie direction lower level
2AF3  C3022B    JP      #2B02           ; skip ahead

2AF6  78        LD      A,B             ; load A with mario X position
2AF7  FE80      CP      #80             ; is mario on the left side of the fire?
2AF9  3AA563    LD      A,(#63A5)       ; load A with upper right pie tray vector
2AFC  D2022B    JP      NC,#2B02        ; no, skip next step

2AFF  3AA463    LD      A,(#63A4)       ; else load A with upper left pie tray vector

2B02  80        ADD     A,B             ; add vector to mario's X position
2B03  320362    LD      (#6203),A       ; set mario's X position
2B06  324C69    LD      (#694C),A       ; set mario's sprite X position
2B09  CD1F24    CALL    #241F           ; loads DE with something depending on mario's position
2B0C  210362    LD      HL,#6203        ; load HL with mario's X position
2B0F  1D        DEC     E               ; E == 1 ?
2B10  CA182B    JP      Z,#2B18         ; yes, skip ahead

2B13  15        DEC     D               ; else D == 1 ?
2B14  CA1A2B    JP      Z,#2B1A         ; yes, skip ahead
2B17  C9        RET                     ; return

2B18  35        DEC     (HL)            ; decrease mario's X position
2B19  C9        RET                     ; return

2B1A  34        INC     (HL)            ; increase
2B1B  C9        RET                     ; return

; called from #1C05

2B1C  DD210062  LD      IX,#6200        ; set IX for mario's array
2B20  CD292B    CALL    #2B29           ; do stuff for jumping.  certain crieria will set A and B and return without the rest
of this sub.
2B23  CDAF29    CALL    #29AF           ; handle jump stuff for elevators
2B26  AF        XOR     A               ; A := 0
2B27  47        LD      B,A             ; B := 0
2B28  C9        RET                     ; return

; arrive here when a jump is in progress
; called from #2B20 above

2B29  3A2762    LD      A,(#6227)       ; load A with screen number
2B2C  3D        DEC     A               ; are we on the girders?
2B2D  C2532B    JP      NZ,#2B53        ; No, skip ahead

; jump on girders

2B30  3A0362    LD      A,(#6203)       ; load A with mario's x position
2B33  67        LD      H,A             ; copy to H
2B34  3A0562    LD      A,(#6205)       ; load A with mario's y position
```

```
2B37  C607      ADD     A,#07           ; add 7 to y position
2B39  6F        LD      L,A             ; copy to L
2B3A  CD9B2B    CALL    #2B9B           ; check for ???
2B3D  A7        AND     A               ; == 0 ?
2B3E  CA512B    JP      Z,#2B51         ; yes, skip ahead and return

2B41  7B        LD      A,E             ; A := E
2B42  91        SUB     C               ; subtract C (???)
2B43  FE04      CP      #04             ; < 4 ?
2B45  D2742B    JP      NC,#2B74        ; no, skip ahead, clear A and B, and return

2B48  79        LD      A,C             ; A := C
2B49  D607      SUB     #07             ; subtract 7
2B4B  320562    LD      (#6205),A       ; store A into mario's Y position
2B4E  3E01      LD      A,#01           ; A : = 1
2B50  47        LD      B,A             ; B := 1

2B51  E1        POP     HL              ; move stack pointer back 1 level
2B52  C9        RET                     ; return to higher sub (EG #1C08)

; arrive from #2B2D when jumping, not on girders, via call from #2B20

2B53  3A0362    LD      A,(#6203)       ; load A with mario X position
2B56  D603      SUB     #03             ; subtract 3
2B58  67        LD      H,A             ; store into H
2B59  3A0562    LD      A,(#6205)       ; load A with mario's Y position
2B5C  C607      ADD     A,#07           ; add 7
2B5E  6F        LD      L,A             ; store into L
2B5F  CD9B2B    CALL    #2B9B           ; check for ???
2B62  FE02      CP      #02             ; A == 2 ?
2B64  CA7A2B    JP      Z,#2B7A         ; yes, skip ahead

2B67  7A        LD      A,D             ; A := D
2B68  C607      ADD     A,#07           ; add 7
2B6A  67        LD      H,A             ; H := A
2B6B  6B        LD      L,E             ; L := E
2B6C  CD9B2B    CALL    #2B9B           ; check for ???
2B6F  A7        AND     A               ; A == 0 ?
2B70  C8        RET     Z               ; yes, return

2B71  C37A2B    JP      #2B7A           ; else skip ahead

2B74  3E00      LD      A,#00           ; A := 0
2B76  0600      LD      B,#00           ; B := 0
2B78  E1        POP     HL              ; move stack pointer to return to higher sub
2B79  C9        RET                     ; return

2B7A  3A1062    LD      A,(#6210)       ; load A with mario's jump direction
2B7D  A7        AND     A               ; jumping to the right ?
2B7E  3A0362    LD      A,(#6203)       ; load A with mario's X position
2B81  CA8B2B    JP      Z,#2B8B         ; if jumping right then skip next 3 steps

2B84  F607      OR      #07             ; mask bits, turn on lower 3 bits
2B86  D604      SUB     #04             ; subtract 4
2B88  C3912B    JP      #2B91           ; skip ahead

2B8B  D608      SUB     #08             ; subtract 8
2B8D  F607      OR      #07             ; mask bits, turn on lower 3 bits
2B8F  C604      ADD     A,#04           ; add 4

2B91  320362    LD      (#6203),A       ; set mario's X position
2B94  324C69    LD      (#694C),A       ; set mario's sprite X position
2B97  3E01      LD      A,#01           ; A := 1
2B99  E1        POP     HL              ; move stack pointer to return to higher sub
2B9A  C9        RET                     ; return

; called from #2B3A and #2B6C and #2B5F above

2B9B  E5        PUSH    HL              ; save HL
2B9C  CDF02F    CALL    #2FF0           ; convert HL into VRAM address
2B9F  D1        POP     DE              ; restore into DE
2BA0  7E        LD      A,(HL)          ; load A with the screen item in VRAM
2BA1  FEB0      CP      #B0             ; > #B0 ? (???)
2BA3  DAD92B    JP      C,#2BD9         ; yes, skip ahead, set results to zero and return

2BA6  E60F      AND     #0F
2BA8  FE08      CP      #08
2BAA  D2D92B    JP      NC,#2BD9        ; yes, skip ahead, set results to zero and return

2BAD  7E        LD      A,(HL)          ; load A with the screen item in VRAM
2BAE  FEC0      CP      #C0             ; == #C0 ?
2BB0  CAD92B    JP      Z,#2BD9         ; yes, skip ahead, set results to zero and return

2BB3  DADC2B    JP      C,#2BDC         ; < #C0 ?  Yes, skip ahead to handle

2BB6  FED0      CP      #D0             ; < #D0 ?
2BB8  DACB2B    JP      C,#2BCB         ; yes, skip ahead to handle

2BBB  FEE0      CP      #E0             ; < #E0 ?
2BBD  DAC52B    JP      C,#2BC5         ; yes, skip ahead to handle

2BC0  FEF0      CP      #F0             ; < #F0 ?
2BC2  DACB2B    JP      C,#2BCB         ; yes, skip ahead to handle (same as < #D0 )

; when landing or jumping from a girder ???
```

```
2BC5  E60F      AND     #0F             ; mask bits, now between 0 and #F
2BC7  3D        DEC     A               ; decrease.  now #FF or between 0 and #E
2BC8  C3CF2B    JP      #2BCF           ; skip ahead

; when jumping his head (harmlessly) into a girder above him?

2BCB  E60F      AND     #0F             ; mask bits, now between 0 and #F
2BCD  D609      SUB     #09             ; subtract 9.  now between #F7 and 6

2BCF  4F        LD      C,A             ; C := A
2BD0  7B        LD      A,E             ; A := E = original Y location
2BD1  E6F8      AND     #F8             ; mask bits.  we dont care about 3 least sig. bits
2BD3  81        ADD     A,C             ; add C
2BD4  4F        LD      C,A             ; C := A
2BD5  BB        CP      E               ; < E (original Y location) ?
2BD6  DAE12B    JP      C,#2BE1         ; no, skip ahead

; mario is jumping clear, nothing in his way

2BD9  AF        XOR     A               ; A := 0
2BDA  47        LD      B,A             ; B := 0
2BDB  C9        RET                     ; return

; mario is jumping and about to land on a conveyor or a girder on the rivets

2BDC  7B        LD      A,E             ; A := E = original Y location
2BDD  E6F8      AND     #F8             ; mask bits.  we dont care about 3 least sig. bits
2BDF  3D        DEC     A               ; decrease
2BE0  4F        LD      C,A             ; copy to C

; mario landing or his head passing through girder above

2BE1  3A0C62    LD      A,(#620C)       ; load A with mario's jump height
2BE4  DD9605    SUB     (IX+#05)        ; subtract the item's Y position (???) [EG IX = #6200 , so this is mario's Y
position)
2BE7  83        ADD     A,E             ; add E (original Y position)
2BE8  B9        CP      C               ; == C ?
2BE9  CAEF2B    JP      Z,#2BEF         ; yes, skip next step

; mario head passing or landing on a noneven girder

2BEC  D2F82B    JP      NC,#2BF8        ; < C ?  no, skip next 4 steps

;  arrive when landing

2BEF  79        LD      A,C             ; A := C = original location masked
2BF0  D607      SUB     #07             ; subtract 7 to adjust for mario' height
2BF2  320562    LD      (#6205),A       ; store A into mario's Y position
2BF5  C3FD2B    JP      #2BFD           ; skip next 3 steps

; arrive when mario has his head passing through girder above

2BF8  3E02      LD      A,#02           ; A := 2
2BFA  0600      LD      B,#00           ; B := 0
2BFC  C9        RET                     ; return

; arrive when ?

2BFD  3E01      LD      A,#01           ; A := 1
2BFF  47        LD      B,A             ; B := 1
2C00  E1        POP     HL
2C01  E1        POP     HL              ; set stack pointer to return to higher subs
2C02  C9        RET                     ; return

; called from main routine at #1989

2C03  3E01      LD      A,#01           ; \ Return if screen is not barrels
2C05  F7        RST     #30             ; /
2C06  D7        RST     #10             ; Return if Mario is not alive

2C07  3A9363    LD      A,(#6393)       ; \  Return if we are already in the process of deploying a barrel, no need to deploy
another one
2C0A  0F        RRCA                    ;  |
2C0B  D8        RET     c               ; /

2C0C  3AB162    LD      A,(#62B1)       ; \  Return if bonus timer is 0, no more barrels are deployed at this time
2C0F  A7        AND     A               ;  |
2C10  C8        RET     Z               ; /

2C11  4f        LD      c,A             ; otherwise load C with current timer value
2C12  3Ab062    LD      A,(#62B0)       ; load a with initial clock value
2C15  d602      SUB     #02             ; subtract 2
2C17  b9        CP      c               ; compare with C = current timer
2C18  dA7b2C    JP      C,#2C7B         ; if carry, jump ahead - we are within first 2 clicks of the round - special barrels
for this.

2C1B  3A8263    LD      A,(#6382)       ; else load A with crazy / blue barrel indicator
2C1E  cb4f      BIT     1,A             ; test bit 1 - is this the second barrel after the first crazy ?
2C20  c2862C    JP      NZ,#2C86        ; if it is, then deploy normal barrel; this barrel is never crazy.

2C23  3A8063    LD      A,(#6380)       ; if not, then load A with difficulty from 1 to 5
2C26  47        LD      B,A             ; For B = 1 to difficulty
2C27  3A1A60    LD      A,(FrameCounter)       ; load A with timer value.  this clock counts down from #FF to 00 over and
over...
```

```
2C2A  E61F      AND     #1F             ; zero out left 3 bits.  the result is between 0 and #1F

2C2C  B8        CP      B               ; compare with Loop counter B (between 1 and 5) ... is higher as time decreases
2C2D  CA332C    JP      Z,#2C33         ; if it equal then jump ahead to check for a crazy barrel

2C30  10FA      DJNZ    #2C2C           ; else Next B

2C32  C9        RET                     ; Return without crazy barrel (?)

; chances of arriving here depend on difficulty D/32 chance .  high levels this is 5/32 = 16%

2C33  3AB062    LD      A,(#62B0)       ; load A with initial clock value
2C36  CB3F      SRL     A               ; Shift Right (div 2)
2C38  B9        CP      C               ; is the current timer value < 1/2 initial clock value ?
2C39  DA412C    JP      C,#2C41         ; NO, skip next 3 steps

2C3C  3A1960    LD      A,(RngTimer2)       ; Yes, Load A with this timer value (random)
2C3F  0F        RRCA                    ; Test Bit 1 of this
2C40  D0        RET     NC              ; If bit 1 is not set, return . this gives 50% extra chance of no crazy barrel when
clock is getting low

2C41  CD5700    CALL    #0057           ; else load A with a random number

;; hack to increase crazy barrels
;; 2C41  3E 00          LD A, #00
;; 2C43  00             NOP

;; hack to increase crazy barrels:
;; 2C44 E600    AND     #00             ; mask all 4 bits to zero
;;

2C44  E60F      AND     #0F             ; mask out left 4 bits to zero.  A becomes a number between 0 and #F
2C46  C2862C    JP      NZ,#2C86        ; If result is not zero, deploy a normal barrel.  this routine sets #6382 to 0,
                                        ; loads A with 3 and returns to #2C4F

; else get a crazy barrel
; can arrive here from #2C7E = first click of round is always crazy barrel

2C49  3E01      LD      A,#01           ; else A := 1 = crazy barrel code

; arrive here from second barrel that is not crazy.  A is preloaded with 2.  From #2C83

2C4B  328263    LD      (#6382),A       ; set a barrel in motion for next barrel, bit 1=crazy, 2 = second barrel which is
always normal, 0 for normal barrel
2C4E  3C        INC     A               ; Increment A for the deployment

2C4F  328F63    LD      (#638F),A       ; store A into the state of the barrel deployment between 3 and 0
2C52  3E01      LD      A,#01           ; A := 1
2C54  329263    LD      (#6392),A       ; set barrel deployment indicator
2C57  3Ab262    LD      A,(#62B2)       ; load A with blue barrel counter
2C5A  B9        CP      C               ; compare with current timer
2C5B  C0        RET     NZ              ; return if not equal

2C5C  D608      SUB     #08             ; if equal then this will be a blue barrel.  decrement A by 8
2C5E  32B262    LD      (#62B2),A       ; put back into blue barrel counter
2C61  112000    LD      DE,#0020        ; now check if all 5 fires are out
2C64  210064    LD      HL,#6400        ; #6400 by 20's contian 1 if these fires exist

2C67  0605      LD      B,#05           ; FOR B = 1 to 5

2C69  7E        LD      A,(HL)          ; get fire status
2C6A  A7        AND     A               ; is this fire onscreen?
2C6B  CA722C    JP      Z,#2C72         ; no, skip next 3 steps; we don't have 5 fires onscreen and therefore have room for a
blue barrel

2C6E  19        ADD     HL,DE           ; yes, add #20 offset to test next fire and loop again
2C6F  10F8      DJNZ    #2C69           ; next B

2C71  C9        RET                     ; not a blue barrel, return

2C72  3A8263    LD      A,(#6382)       ; load A with crazy/blue barrel indicator
2C75  f680      or      #80             ; or with #80  - set leftmost bit on to indicate blue barrel is next
2C77  328263    LD      (#6382),A       ; store into crazy/blue barrel indicator
2C7A  c9        RET                     ; return with blue barrel

; we arrive here if timer is within first 2 clicks when deploying a barrel from #2C18

2C7B  C602      ADD     A,#02           ; A := A + 2 (A had the initial clock value -2, now it has the initial clock value)
2C7D  B9        CP      C               ; compare to current timer value - are we starting this round now?
2C7E  CA492C    JP      Z,#2C49         ; yes, do a crazy barrel

2C81  3E02      LD      A,#02           ; else A := 2 for the second barrel; it is always normal
2C83  C34B2C    JP      #2C4B           ; jump back and continue deployment

; arrive here when the second barrel is being deployed?
; from #2C20

2C86  AF        XOR     A               ; A := 0
2C87  328263    LD      (#6382),A       ; barrel indicator to 0 == normal barrel
2C8A  3E03      LD      A,#03           ; A := 3 -- use for upcoming deployement indicator == position #3
2C8C  C34F2C    JP      #2C4F           ; Jump back

; called from main routine #1986

2C8F  3E01      LD      A,#01           ; A := 1 = code for girders
```

```
2C91  F7        RST     #30             ; if screen is girders, continue.  else RET
2C92  D7        RST     #10             ; if mario is alive, continue.  else RET
2C93  3A9363    LD      A,(#6393)       ; load A with barrel deployment indicator
2C96  0F        RRCA                    ; is a barrel being deployed ?
2C97  DA152D    JP      C,#2D15         ; yes, skip ahead

2C9A  3A9263    LD      A,(#6392)       ; else load A with other barrel deployment indicator
2C9D  0F        RRCA                    ; deployed ?
2C9E  D0        RET     NC              ; no, return

; else a barrel is being deployed

2C9F  DD210067  LD      IX,#6700        ; load IX with start of barrel memory
2CA3  112000    LD      DE,#0020        ; incrementer gets #20
2CA6  060A      LD      B,#0A           ; For B = 1 to #0A (all 10 barrels)

2CA8  DD7E00    LD      A,(IX+#00)      ; load A with +0 indicator
2CAB  0F        RRCA                    ; is this barrel already rolling ?
2CAC  DAB32C    JP      C,#2CB3         ; yes, then jump ahead and test next barrel

2CAF  0F        RRCA                    ; else is this barrel already being deployed ?
2CB0  D2B82C    JP      NC,#2CB8        ; no, then jump ahead

2CB3  DD19      ADD     IX,DE           ; Increase to next barrel
2CB5  10F1      DJNZ    #2CA8           ; Next B

2CB7  C9        RET                     ; return

; arrive here when a barrel is being deployed

2CB8  DD22AA62  LD      (#62AA),IX      ; save this barrel indicator into #62AA.  it is recalled at #2D55
2CBC  DD360002  LD      (IX+#00),#02    ; set deployement indicator
2CC0  1600      LD      D,#00           ; D := 0
2CC2  3E0A      LD      A,#0A           ; A := #0A
2CC4  90        SUB     B               ; A = A - B ;  B has the number of the barrel A now will be 0 if this is the first
barrel, #0A if the last
2CC5  87        ADD     A,A             ; A = A * 2
2CC6  87        ADD     A,A             ; A = A * 2 (A is now 4 times what it was)
2CC7  5F        LD      E,A             ; copy this to E
2CC8  218069    LD      HL,#6980        ; load HL with starting sprite address for the barrels
2CCB  19        ADD     HL,DE           ; Now add in offset depending on the barrel number ( will vary from 0 to #28 by 4's)
2CCC  22AC62    LD      (#62AC),HL      ; store this info in #62AC. will vary from #80 to #A8
2CCF  3E01      LD      A,#01           ; A := 1
2CD1  329363    LD      (#6393),A       ; set barrel deployment indicator
2CD4  110105    LD      DE,#0501        ; load DE with task #5, parameter 1 update onscreen bonus timer and play sound &
change to red if below 1000
2CD7  CD9F30    CALL    #309F           ; insert task
2CDA  21B162    LD      HL,#62B1        ; load bonus counter into HL
2CDD  35        DEC     (HL)            ; decrement bonus counter.  Is it zero?
2CDE  c2E62C    JP      NZ,#2Ce6        ; no, skip next 2 steps

2CE1  3E01      LD      A,#01           ; A := 1
2CE3  328663    LD      (#6386),A       ; store into bonus timer out indicator

2CE6  7E        LD      A,(HL)          ; load A with bonus counter
2CE7  FE04      CP      #04             ; bonus <= 400 ?
2CE9  D2f62C    JP      NC,#2Cf6        ; no, skip ahead

2CEC  21A869    LD      HL,#69A8        ; else load HL with extra barrels sprites
2CEF  87        ADD     A,A
2CF0  87        ADD     A,A             ; A := A * 4
2CF1  5F        LD      E,A             ; copy to E
2CF2  1600      LD      D,#00           ; D := 0.  DE now has offset based on timer
2CF4  19        ADD     HL,DE           ; compute which sprite to remove based on timer
2CF5  72        LD      (HL),D          ; clear the sprite

; IX holds 6700 +N*20 = start of barrel N info
; a barrel is being deployed

2CF6  DD360715  LD      (IX+#07),#15    ; set barrel sprite value to #15
2CFA  DD36080B  LD      (IX+#08),#0B    ; set barrel color to #0B
2CFE  DD361500  LD      (IX+#15),#00    ; set +15 indicator to 0 = normal barrel,  [1 = blue barrel]
2D02  3A8263    LD      A,(#6382)       ; load A with Crazy/Blue barrel indicator
2D05  07        RLCA                    ; is this a blue barrel ?
2D06  D2152D    JP      NC,#2D15        ; No blue barrel, then skip next 3 steps

; blue barrel

2D09  DD360719  LD      (IX+#07),#19    ; set sprite for blue barrel
2D0D  DD36080C  LD      (IX+#08),#0C    ; set sprite color to blue
2D11  DD361501  LD      (IX+#15),#01    ; set blue barrel indicator

2D15  21AF62    LD      HL,#62AF        ; load HL with deployment timer
2D18  35        DEC     (HL)            ; count it down.  is the timer expired?
2D19  C0        RET     NZ              ; no, return

2D1A  3618      LD      (HL),#18        ; else reset the counter back to #18
2D1C  3A8F63    LD      A,(#638F)       ; load A with the deployment indiacator.  2 = kong grabbing, 1 = kong holding, 0 =
deploying, 3 = kong empty
2D1F  A7        AND     A               ; is a barrel being deployed right now?
2D20  CA512D    JP      Z,#2D51         ; yes, jump ahead

2D23  4F        LD      C,A             ; else copy A to C
2D24  213239    LD      HL,#3932        ; load HL with table data start
2D27  3A8263    LD      A,(#6382)       ; load A with crazy/blue barrel indicator
```

```
2D2A  0F        RRCA                    ; Is this a crazy barrel?
2D2B  DA2F2D    JP      C,#2D2F         ; yes, skip next step

2D2E  0D        DEC     C               ; no, Decrement C

2D2F  79        LD      A,C
2D30  87        ADD     A,A
2D31  87        ADD     A,A
2D32  87        ADD     A,A
2D33  4F        LD      C,A
2D34  87        ADD     A,A
2D35  87        ADD     A,A
2D36  81        ADD     A,C
2D37  5F        LD      E,A             ; A is #50 when barrel is crazy, #28 when normal
2D38  1600      LD      D,#00           ; D: = 0
2D3A  19        ADD     HL,DE           ; HL becomes #3982 when barrel is crazy, 395A when normal, 3932 when deploying all
the way.  this will skip the final animation when dropping crazy barrel (?)
2D3B  CD4E00    CALL    #004E           ; update kong's sprites
2D3E  218F63    LD      HL,#638F        ; load HL with deployment indicator
2D41  35        DEC     (HL)            ; Decrease indicator
2D42  C2512D    JP      NZ,#2D51        ; if indicator is not zero then jump ahead

2D45  3E01      LD      A,#01           ; else A := 1
2D47  32AF62    LD      (#62AF),A       ; Store into ???
2D4A  3A8263    LD      A,(#6382)       ; load A with crazy/blue barrel indicator
2D4D  0F        RRCA                    ; Is this a crazy barrel?
2D4E  DA832D    JP      C,#2D83         ; yes, jump ahead and load HL with #39CC and store into #62A8 and #62A9 and resume on
#2D54

2D51  2AA862    LD      HL,(#62A8)      ; else load HL with (???)

2D54  7E        LD      A,(HL)          ; load A with value in HL.  crazy barrel this value is #BB
2D55  DD2AAA62  LD      IX,(#62AA)      ; load IX with Barrel start address saved above
2D59  ED5BAC62  LD      DE,(#62AC)      ; load DE with sprite variable start  EG #6980.  set in #2CCC
2D5D  FE7F      CP      #7F             ; A == #7F ? (time to deploy out of kong's hands ?)
2D5F  CA8C2D    JP      Z,#2D8C         ; yes, jump ahead

2D62  4F        LD      C,A             ; else copy A into C
2D63  E67F      AND     #7F             ; mask out leftmost bit.  result between 0 and  #7F
2D65  12        LD      (DE),A          ; store into sprite X position
2D66  DD7E07    LD      A,(IX+#07)      ; load A with barrel sprite value
2D69  CB79      BIT     7,C             ; test bit 7 of C
2D6B  CA702D    JP      Z,#2D70         ; yes, skip next step

2D6E  EE03      XOR     #03             ; no, toggle the rightmost 2 bits

2D70  13        INC     DE              ; DE now has sprite value
2D71  12        LD      (DE),A          ; store new sprite
2D72  DD7707    LD      (IX+#07),A      ; store into barrel sprite value
2D75  DD7E08    LD      A,(IX+#08)      ; load A with barrel color
2D78  13        INC     DE              ; DE now has sprite color value
2D79  12        LD      (DE),A          ; store color into sprite
2D7A  23        INC     HL              ; increase HL.  EG #39CD for crazy barrel
2D7B  7E        LD      A,(HL)          ; load A with this value.  EG #4D for crazy barrel
2D7C  13        INC     DE              ; DE now has Y position
2D7D  12        LD      (DE),A          ; store into sprite Y position
2D7E  23        INC     HL              ; increase HL .  EG #39CE for crazy barrel
2D7F  22A862    LD      (#62A8),HL      ; store into 62A8.  EG 62A8 = CE, 62A9 = 39
2D82  C9        RET                     ; return

; arrive here because this barrel is crazy from #2D4E

2D83  21CC39    LD      HL,#39CC        ; load HL with crazy barrel data

        ; 39CC  BB
        ; 39CD  4D

2D86  22A862    LD      (#62A8),HL      ; Load #62A8 and #62A9 with #39 and #CC
2D89  C3542D    JP      #2D54           ; jump back

; jump here from #2D5F
; kong is releasing a barrel (?)

2D8C  21C339    LD      HL,#39C3        ; load HL with start of table data address
2D8F  22A862    LD      (#62A8),HL      ; store into ???
2D92  DD360101  LD      (IX+#01),#01    ; set crazy barrel indicator
2D96  3A8263    LD      A,(#6382)       ; load A with crazy/blue barrel indicator

2D99  0F        RRCA                    ; roll right.  is this a crazy barrel?
2D9A  DAA52D    JP      C,#2DA5         ; yes, skip next 2 steps

2D9D  DD360100  LD      (IX+#01),#00    ; no , clear crazy indicator
2DA1  DD360202  LD      (IX+#02),#02    ; load motion indicator with 2 (rolling right)

2DA5  DD360001  LD      (IX+#00),#01    ; barrel is now active
2DA9  DD360F01  LD      (IX+#0F),#01    ;
2DAD  AF        XOR     A               ; A := 0
2DAE  DD7710    LD      (IX+#10),A      ; clear this indicator (???)
2DB1  DD7711    LD      (IX+#11),A
2DB4  DD7712    LD      (IX+#12),A
2DB7  DD7713    LD      (IX+#13),A
2DBA  DD7714    LD      (IX+#14),A
2DBD  329363    LD      (#6393),A       ; clear barrel deployment indicator
2DC0  329263    LD      (#6392),A       ; clear barrel deployment indicator
2DC3  1A        LD      A,(DE)          ; load A with kong hand sprite X position
```

```
2DC4  DD7703    LD      (IX+#03),A      ; store in barrel's X position
2DC7  13        INC     DE
2DC8  13        INC     DE
2DC9  13        INC     DE              ; DE := DE + 3 = DE now has kong hand sprite Y position
2DCA  1A        LD      A,(DE)          ; load A with kong hand Y position
2DCB  DD7705    LD      (IX+#05),A      ; store in barrel's Y position
2DCE  215C38    LD      HL,#385C        ; load HL with table data start
2DD1  CD4E00    CALL    #004E           ; update kong's sprites
2DD4  210B69    LD      HL,#690B        ; load HL with start of Kong sprite
2DD7  0EFC      LD      C,#FC           ; load c with offset of -4
2DD9  FF        RST     #38             ; move kong
2DDA  C9        RET                     ; return

; deploys fireball/firefoxes
; Arrive here from main routine at #1995

2DDB  3E0A      LD      A,#0A           ; A := binary 1010 = code for rivets and conveyors
2DDD  F7        RST     #30             ; returns immediately on girders and elevators, else continue

2DDE  D7        RST     #10             ; only continue if mario alive
2DDF  3A8063    LD      A,(#6380)       ; \  load B with (internal_difficulty+1)/2 (get's value between 1 and 3)
2DE2  3C        INC     A               ;  |
2DE3  A7        AND     A               ;  | clear carry flag
2DE4  1F        RRA                     ;  |
2DE5  47        LD      B,A             ; /
2De6  3A2762    LD      A,(#6227)       ; \  Increment B by 1 if we are on conveyors (to get value between 2 and 4)
2De9  fe02      CP      #02             ;  |
2Deb  2001      JR      NZ,#2DEE        ;  |
2Ded  04        INC     b               ; /

2DEE  3EFE      LD      A,#FE           ; \  Load A with #FF>>(B-1) (note the first rotate right doesn't count towards the
bit shift because the
2DF0  37        SCF                     ;  | carry flag is set)
2DF1  1F        RRA                     ;  |
2DF2  A7        AND     A               ;  | clear carry flag
2DF3  10FC      DJNZ    #2DF1           ; /

2DF5  47        LD      B,A             ; \  The result of the above indicates the interval in frames between deploying
successive fires.
2DF6  3A1A60    LD      A,(FrameCounter)       ;  | On rivets we proceed every 256 frames for internal difficulty 1 and 2,
128 frames for internal difficulty
2DF9  A0        AND     B               ;  | 3 and 4 and 64 frames for internal difficulty 5. On conveyors these values are
cut in half.
2DFA  C0        RET     NZ              ; /

2DFB  3E01      LD      A,#01           ; Time to deploy a fire. Load A with 1
2DFD  32A063    LD      (#63A0),A       ; deploy a firefox/fireball
2E00  329A63    LD      (#639A),A       ; set deployment indicator ?
2E03  C9        RET                     ; return

; called from main routine at #198F
; called during the elevators.  used to move the bouncers ????

2E04  3E04      LD      A,#04           ; A := 4 (0100 binary) to check for elevators screen
2E06  F7        RST     #30             ; if not elevators it will return to program

2E07  D7        RST     #10             ; if mario is alive, continue, else RET

2E08  DD210065  LD      IX,#6500        ; load IX with start of bouncer memory area
2E0C  FD218069  LD      IY,#6980        ; start of sprite memory for bouncers
2E10  060A      LD      B,#0A           ; for B = 1 to #0A (ten) .  do for all ten sprites

2E12  DD7E00    LD      A,(IX+#00)      ; load A with sprite status
2E15  0F        RRCA                    ; is the sprite active ?
2E16  D2A72E    JP      NC,#2EA7        ; no, jump ahead and check to deploy a new one

2E19  3A1A60    LD      A,(FrameCounter)       ; else load A with timer

; FrameCounter - Timer constantly counts down from FF to 00 and then FF to 00 again and again ... 1 count per frame
; result is that each of the boucners have their sprites changed once every 16 clicks, or every 1/16 of sec.?

2E1C  E60F      AND     #0F             ; mask out left 4 bits.  result between 0 and F
2E1E  C2292E    JP      NZ,#2E29        ; if not zero, jump ahead..

2E21  FD7E01    LD      A,(IY+#01)      ; load A with sprite value
2E24  EE07      XOR     #07             ; flip the right 3 bits
2E26  FD7701    LD      (IY+#01),A      ; store result = change the bouncer fom open to closed

2E29  DD7E0D    LD      A,(IX+#0D)      ; load A with +D = either 1 or 4.  1 when going across , 4 when going down.
2E2C  FE04      CP      #04             ; is it == 4 ? (going down?)
2E2E  CA842E    JP      Z,#2E84         ; yes, jump ahead

2E31  DD3403    INC     (IX+#03)        ; no, increase X position
2E34  DD3403    INC     (IX+#03)        ; increase X position again
2E37  DD6E0E    LD      L,(IX+#0E)
2E3A  DD660F    LD      H,(IX+#0F)      ; load HL with table address for bouncer offsets of Y positions for each pixel across
2E3D  7E        LD      A,(HL)          ; load table data
2E3E  4F        LD      C,A             ; copy to C
2E3F  FE7F      CP      #7F             ; == #7F ? (end code ?)
2E41  CA9C2E    JP      Z,#2E9C         ; yes, jump ahead, reset HL to #39AA, play bouncer sound, and continue at #2E4B

2E44  23        INC     HL              ; next HL
2E45  DD8605    ADD     A,(IX+#05)      ; add item's Y position
2E48  DD7705    LD      (IX+#05),A      ; store into item's Y position
```

```
2E4B  DD750E    LD      (IX+#0E),L
2E4E  DD740F    LD      (IX+#0F),H      ; store the updated HL for next time
2E51  DD7E03    LD      A,(IX+#03)      ; load A with X position
2E54  FEB7      CP      #B7             ; < #B7 ?
2E56  DA6C2E    JP      C,#2E6C         ; no, skip ahead

2E59  79        LD      A,C             ; yes, A := C
2E5A  FE7F      CP      #7F             ; == #7F (end code?)
2E5C  C26C2E    JP      NZ,#2E6C        ; no, skip ahead

2E5F  DD360D04  LD      (IX+#0D),#04    ; set +D to 4 (???)
2E63  AF        XOR     A               ; A := 0
2E64  328360    LD      (#6083),A       ; clear sound of bouncer
2E67  3E03      LD      A,#03           ; load sound duration of 3
2E69  328460    LD      (#6084),A       ; play sound for falling bouncer

2E6C  DD7E03    LD      A,(IX+#03)      ; load A with X position
2E6F  FD7700    LD      (IY+#00),A      ; store into sprite
2E72  DD7E05    LD      A,(IX+#05)      ; load A with Y position
2E75  FD7703    LD      (IY+#03),A      ; store into sprite

2E78  111000    LD      DE,#0010        ; set offset to add
2E7B  DD19      ADD     IX,DE           ; next sprite (IX)
2E7D  1E04      LD      E,#04           ; E := 4
2E7F  FD19      ADD     IY,DE           ; next sprite (IY)
2E81  108F      DJNZ    #2E12           ; Next Bouncer

2E83  C9        RET                     ; return

; arrive when bouncer is going straight down
; need to check when falling off bottom of screen

2E84  3E03      LD      A,#03           ; A := 3
2E86  DD8605    ADD     A,(IX+#05)      ; add to Sprite's y position (move down 3)
2E89  DD7705    LD      (IX+#05),A      ; store result
2E8C  FEF8      CP      #F8             ; are we at the bottom of screen?
2E8E  DA6C2E    JP      C,#2E6C         ; No, jump back to program

2E91  DD360300  LD      (IX+#03),#00    ; yes, reset the sprite
2E95  DD360000  LD      (IX+#00),#00    ; reset
2E99  C36C2E    JP      #2E6C           ; jump back to program

; arrive from #2E41

2E9C  21AA39    LD      HL,#39AA        ; load HL with start of table data
2E9F  3E03      LD      A,#03           ; load sound duration of 3
2EA1  328360    LD      (#6083),A       ; play sound for bouncer
2EA4  C34B2E    JP      #2E4B           ; jump back

; jump here from #2E16

2EA7  3A9663    LD      A,(#6396)       ; load A with bouncer release flag
2EAA  0F        RRCA                    ; time to deploy a bouncer?
2EAB  D2782E    JP      NC,#2E78        ; no, jump back

; deploy new bouncer

2EAE  AF        XOR     A               ; A := 0
2EAF  329663    LD      (#6396),A       ; reset bouncer release flag
2EB2  DD360550  LD      (IX+#05),#50    ; set bouncer's Y position to #50
2EB6  DD360D01  LD      (IX+#0D),#01    ; set value to sprite bouncing across, not down
2EBA  CD5700    CALL    #0057           ; load A with random number
2EBD  E60F      AND     #0F             ; mask bits, result is between 0 and #F
2EBF  C6F8      ADD     A,#F8           ; add #F8 = result is now between #F8 and #07
2EC1  DD7703    LD      (IX+#03),A      ; store A into initial X position for bouncer sprite
2EC4  DD360001  LD      (IX+#00),#01    ; set sprite as active
2EC8  21AA39    LD      HL,#39AA        ; values #39 and #AA to be inserted below.  #39AA is the start of table data for Y
offsets to add for each movement
2ECB  DD750E    LD      (IX+#0E),L      ;
2ECE  DD740F    LD      (IX+#0F),H      ; store HL into +E and +F
2ED1  C3782E    JP      #2E78           ; jump back

; arrive from main routine at #1998
; checks for hammer grabs etc ?

2ED4  3E0B      LD      A,#0B           ; B = # 1011 binary
2ED6  F7        RST     #30             ; continue here on girders, conveyors, rivets only.  elevators RET from this sub, it
has no hammers.
2ED7  D7        RST     #10             ; continue here only if mario is alive, otherwise RET from this sub

2ED8  11186A    LD      DE,#6A18        ; load DE with hardware address of hammer sprite
2EDB  DD218066  LD      IX,#6680        ; load IX with software address of hammer sprite
2EDF  DD7E01    LD      A,(IX+#01)      ; load A with 1st hammer active indicator
2EE2  0F        RRCA                    ; rotate right.  carry set?  (is this hammer active?)
2EE3  DAED2E    JP      C,#2EED         ; yes, skip next 2 steps

2EE6  111C6A    LD      DE,#6A1C        ; else load DE with hardware address of 2nd hammer sprite
2EE9  DD219066  LD      IX,#6690        ; load IX with 2nd hammer sprite

2EED  DD360E00  LD      (IX+#0E),#00    ; store 0 into +#E == ???
2EF1  DD360FF0  LD      (IX+#0F),#F0    ; store #F0 into +#F (???)
2EF5  3A1762    LD      A,(#6217)       ; load A with hammer indicator
2EF8  0F        RRCA                    ; is the hammer already active?
2EF9  D2972F    JP      NC,#2F97        ; no, skip ahead and check for new hammer grab
```

```
2EFC  AF        XOR     A               ; A := 0
2EFD  321862    LD      (#6218),A       ; store into grabbing the hammer indicator. the grab is complete.
2F00  218960    LD      HL,#6089        ; load HL with music address
2F03  3604      LD      (HL),#04        ; set music for hammer
2F05  DD360906  LD      (IX+#09),#06    ; set width ?
2F09  DD360A03  LD      (IX+#0A),#03    ; set height ?
2F0D  061E      LD      B,#1E           ; B := #1E
2F0F  3A0762    LD      A,(#6207)       ; load A with mario movement indicator/sprite value
2F12  CB27      SLA     A               ; shift left.  is bit 7 on?
2F14  D21B2F    JP      NC,#2F1B        ; no, skip next 2 steps

2F17  F680      OR      #80             ; turn on bit 7 in A
2F19  CBF8      SET     7,B             ; turn on bit 7 in B

2F1B  F608      OR      #08             ; turn on bit 3 in A
2F1D  4F        LD      C,A             ; copy to C
2F1E  3A9463    LD      A,(#6394)       ; load A with hammer timer
2F21  CB5F      BIT     3,A             ; is bit 3 on in A?
2F23  CA432F    JP      Z,#2F43         ; no, skip ahead

; animate the hammer

2F26  CBC0      SET     0,B
2F28  CBC1      SET     0,C
2F2A  DD360905  LD      (IX+#09),#05    ; set width?
2F2E  DD360A06  LD      (IX+#0A),#06    ; set height?
2F32  DD360F00  LD      (IX+#0F),#00    ;
2F36  DD360EF0  LD      (IX+#0E),#F0    ; set offset for left side of mario (#F0 == -#10)
2F3A  CB79      BIT     7,C             ; is mario facing left?
2F3C  CA432F    JP      Z,#2F43         ; yes, skip next step

2F3F  DD360E10  LD      (IX+#0E),#10    ; set offset for right side of mario

2F43  79        LD      A,C             ; A := C
2F44  324D69    LD      (#694D),A       ; store into mario sprite value
2F47  0E07      LD      C,#07           ; C := 7
2F49  219463    LD      HL,#6394        ; load HL with hammer timer
2F4C  34        INC     (HL)            ; increase.  at zero?
2F4D  C2B72F    JP      NZ,#2FB7        ; no skip ahead

; hammer is changing or ending

2F50  219563    LD      HL,#6395        ; load HL with hammer length.
2F53  34        INC     (HL)            ; increase
2F54  7E        LD      A,(HL)          ; get the value
2F55  FE02      CP      #02             ; is the hammer all used up?
2F57  C2BE2F    JP      NZ,#2FBE        ; no, skip ahead and change its color every 8 frames

; arrive here when hammer runs out

2F5A  AF        XOR     A               ; A := 0
2F5B  329563    LD      (#6395),A       ; clear hammer length
2F5E  321762    LD      (#6217),A       ; store into hammer indicator
2F61  DD7701    LD      (IX+#01),A      ; clear hammer active indicator
2F64  3A0362    LD      A,(#6203)       ; load A with mario's X position
2F67  ED44      NEG                     ; take negative
2F69  DD770E    LD      (IX+#0E),A      ; store into +E
2F6C  3A0762    LD      A,(#6207)       ; load A with mario movement indicator/sprite value
2F6F  324D69    LD      (#694D),A       ; store into mario sprite value
2F72  DD360000  LD      (IX+#00),#00    ; clear hammer active bit
2F76  3A8963    LD      A,(#6389)       ; load A with previous background music
2F79  328960    LD      (#6089),A       ; set music with what it was before the hammer was grabbed

;

2F7C  EB        EX      DE,HL           ; DE <> HL
2F7D  3A0362    LD      A,(#6203)       ; load A with mario's X position
2F80  DD860E    ADD     A,(IX+#0E)      ; add hammer offset
2F83  77        LD      (HL),A          ; store into Hammer X position
2F84  DD7703    LD      (IX+#03),A      ; store into hammer X position
2F87  23        INC     HL              ; next
2F88  70        LD      (HL),B          ; store sprite graphic value
2F89  23        INC     HL              ; next
2F8A  71        LD      (HL),C          ; store into hammer color
2F8B  23        INC     HL              ; next
2F8C  3A0562    LD      A,(#6205)       ; load A with mario's Y position
2F8F  DD860F    ADD     A,(IX+#0F)      ; add hammer offset
2F92  77        LD      (HL),A          ; store into hammer Y position
2F93  DD7705    LD      (IX+#05),A      ; store into hammer Y position
2F96  C9        RET                     ; return

; arrive from #2EF9, check for grabbing hammer ?

2F97  3A1862    LD      A,(#6218)       ; load A with 0, turns to 1 while mario is grabbing the hammer until he lands
2F9A  0F        RRCA                    ; is mario grabbing the hammer?
2F9B  D0        RET     NC              ; no, return

; arrive here when hammer is grabbed

2F9C  DD360906  LD      (IX+#09),#06    ; set width ?
2FA0  DD360A03  LD      (IX+#0A),#03    ; set height ?
2FA4  3A0762    LD      A,(#6207)       ; load A with mario movement indicator/sprite value
2FA7  07        RLCA                    ; rotate left the high bit into carry flag
2FA8  3E3C      LD      A,#3C           ; A := #3C
2FAA  1F        RRA                     ; rotate right the carry bit back in
```

```
2FAB  47        LD      B,A             ; copy to B
2FAC  0E07      LD      C,#07           ; C := 7
2FAE  3A8960    LD      A,(#6089)       ; load A with background music value
2FB1  328963    LD      (#6389),A       ; save so it can be restored when hammer runs out.  see #2F76
2FB4  C37C2F    JP      #2F7C           ; return to program

; arrive from #2F4D

2FB7  3A9563    LD      A,(#6395)       ; load A with hammer length
2FBA  A7        AND     A               ; == 0 ?  (full strength)
2FBB  CA7C2F    JP      Z,#2F7C         ; yes, jump back now

; change hammer color ?
; hammer is half strength

2FBE  3A1A60    LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
2FC1  CB5F      BIT     3,A             ; check bit 3 (?).  zero ?  will do this every 8 frames
2FC3  CA7C2F    JP      Z,#2F7C         ; yes, jump back now

2FC6  0E01      LD      C,#01           ; else C := 1 to change hammer color
2FC8  C37C2F    JP      #2F7C           ; jump back

; arrive here from main routine #19BF
; this is the last subroutine from there
; for non-girder levels, this sub
; checks for bonus timer changes
; if the bonus counts down, it also
; sets a possible new fire to be released
; sets a bouncer to be deployed
; updates the bonus timer onscreen
; checks for bonus time running out

2FCB  3E0E      LD      A,#0E           ; A := #E = 1110 binary
2FCD  F7        RST     #30             ; is this the girders?  if so, return immediately

2FCE  21B462    LD      HL,#62B4        ; else load HL with timer
2FD1  35        DEC     (HL)            ; count down timer.  at zero?
2FD2  C0        RET     NZ              ; no, return

2FD3  3E03      LD      A,#03           ; else A := 3
2FD5  32B962    LD      (#62B9),A       ; store into fire release - a new fire can be released
2FD8  329663    LD      (#6396),A       ; store into bouncer release - a new bouncer can be deployed
2FDB  110105    LD      DE,#0501        ; load task #5, parameter #1 = update onscreen bonus timer and play sound & change to
red if below 1000
2FDE  CD9F30    CALL    #309F           ; insert task
2FE1  3AB362    LD      A,(#62B3)       ; load A with intial timer value.
2FE4  77        LD      (HL),A          ; reset the timer
2fe5  21B162    LD      HL,#62B1        ; load HL with bonus timer
2fe8  35        DEC     (HL)            ; Decrement.  is the bonus timer zero?
2fe9  c0        RET     NZ              ; no, return

2fea  3E01      LD      A,#01           ; else time has run out.  A := 1
2fec  328663    LD      (#6386),A       ; set time has run out indicator
2fef  c9        RET                     ; return

; called during a barrel roll
; HL contains the X and Y position of the barrel.  Y has been inflated by 4
; called from #2A3A
; called from #2AA2 with HL preloaded with mario's position offset a bit
; returns with HL modified in some special way
;

2FF0  7D        LD      A,L             ; load A with Y position (inflated by 4)
2FF1  0F        RRCA                    ; Roll right 3 times
2FF2  0F        RRCA                    ;
2FF3  0F        RRCA                    ;
2FF4  E61F      AND     #1F             ; mask out left 3 bits to zero (number has been divided by 8)
2FF6  6F        LD      L,A             ; Load L with this new position
2FF7  7C        LD      A,H             ; load A with barrel's X position
2FF8  2F        CPL                     ; A is inverted (1's complement)
2FF9  E6F8      AND     #F8             ; Mask out right 3 bits to zero
2FFB  5F        LD      E,A             ; load E with result
2FFC  AF        XOR     A               ; A := 0
2FFD  67        LD      H,A             ; H := 0
2FFE  CB13      RL      E               ; rotate E left
3000  17        RLA                     ; Rotate A left [does nothing?  A is 0]
3001  CB13      RL      E               ; rotate E left again
3003  17        RLA                     ; rotate A left again ?
3004  C674      ADD     A,#74           ; Add #74 to A.   A = #74 now ?
3006  57        LD      D,A             ; Store this in D
3007  19        ADD     HL,DE           ; Add DE into HL
3008  C9        RET                     ; return


;
; called here in the middle of a barrlel being rolled left or right...
; or when mario is moving
; called from four locations
; A is preloaded with ?
;

3009  57        LD      D,A             ; D := A
300A  0F        RRCA                    ; roll right.  is A odd?
300B  DA2230    JP      C,#3022         ; yes, skip ahead
```

```
; A is even

300E  0E93      LD      C,#93           ; C := #93
3010  0F        RRCA
3011  0F        RRCA                    ; roll right twice
3012  D21730    JP      NC,#3017        ; no carry, skip next step

3015  0E6C      LD      C,#6C           ; C := #6C

3017  07        RLCA                    ; roll left
3018  DA3130    JP      C,#3031         ; if carry, skip ahead

301B  79        LD      A,C             ; A := C
301C  E6F0      AND     #F0             ; mask bits, 4 lowest bits set to zero
301E  4F        LD      C,A             ; store back into C
301F  C33130    JP      #3031           ; skip ahead

; arrive from #300B when A is odd

3022  0EB4      LD      C,#B4           ; C := #B4
3024  0F        RRCA
3025  0F        RRCA                    ; rotate A right twice.  carry set ?
3026  D22B30    JP      NC,#302B        ; no, skip next step

3029  0E1E      LD      C,#1E           ; C := #1E

302B  CB50      BIT     2,B             ; is bit 2 on B at zero?
302D  CA3130    JP      Z,#3031         ; yes, skip next step

3030  05        DEC     B               ; else decrease B

3031  79        LD      A,C             ; A := C
3032  0F        RRCA
3033  0F        RRCA                    ; rotate right twice
3034  4F        LD      C,A             ; C := A
3035  E603      AND     #03             ; mask bits, now between 0 and 3
3037  B8        CP      B               ; == B ?
3038  C23130    JP      NZ,#3031        ; no, loop again

303B  79        LD      A,C             ; A := C
303C  0F        RRCA
303D  0F        RRCA                    ; rotate right twice
303E  E603      AND     #03             ; mask bits, now between 0 and 3
3040  FE03      CP      #03             ; == 3 ?
3042  C0        RET     NZ              ; no, return

3043  CB92      RES     2,D             ; clear bit 2 of D (copy of original input A)
3045  15        DEC     D               ; decrease.  zero?
3046  C0        RET     NZ              ; no, return

3047  3E04      LD      A,#04           ; else A := 4
3049  C9        RET                     ; return

; called from #0AF0 and #0B38
; rolls up kong's ladder during intro

304A  11E0FF    LD      DE,#FFE0        ; load DE with offset
304D  3A8E63    LD      A,(#638E)       ; load A with kong ladder climb counter
3050  4F        LD      C,A             ; copy to C
3051  0600      LD      B,#00           ; B := 0
3053  210076    LD      HL,#7600        ; load HL with screen RAM address
3056  CD6430    CALL    #3064           ; roll up left ladder
3059  21C075    LD      HL,#75C0        ; load HL with screen RAM address
305C  CD6430    CALL    #3064           ; roll up right ladder
305F  218E63    LD      HL,#638E        ; load HL with kong ladder climb counter
3062  35        DEC     (HL)            ; decrease
3063  C9        RET                     ; return

; called from #3056 and #305C above

3064  09        ADD     HL,BC           ; add offset based on how far up kong is
3065  7E        LD      A,(HL)          ; get value from screen
3066  19        ADD     HL,DE           ; add offset
3067  77        LD      (HL),A          ; store value to screen
3068  C9        RET                     ; return

; arrive from #0A79 when intro screen indicator == 3 or 5

3069  DF        RST     #18             ; count down timer and only continue here if zero, else RET
306A  2AC063    LD      HL,(#63C0)      ; load HL with timer ???
306D  34        INC     (HL)            ; increase
306E  C9        RET                     ; return

; called from 3 locations

306F  21AF62    LD      HL,#62AF        ; load HL with kong climbing counter
3072  34        INC     (HL)            ; increase
3073  7E        LD      A,(HL)          ; load A with the counter
3074  E607      AND     #07             ; mask bits.  now between 0 and 7.  zero?
3076  C0        RET     NZ              ; no, return

; animate kong climbing up the ladder

3077  210B69    LD      HL,#690B        ; load HL with kong sprite array
307A  0EFC      LD      C,#FC           ; C := -4
```

```
307C  FF        RST     #38             ; move kong
307D  0E81      LD      C,#81           ; C := #81
307F  210969    LD      HL,#6909        ; load HL with kong's right leg address sprite
3082  CD9630    CALL    #3096           ; animate kong sprite
3085  211D69    LD      HL,#691D        ; load HL with kong's right arm address sprite
3088  CD9630    CALL    #3096           ; animate kong sprite
308B  CD5700    CALL    #0057           ; load A with random number
308E  E680      AND     #80             ; mask bits, now either 0 or #80
3090  212D69    LD      HL,#692D        ; load HL with sprite of girl under kong's arms
3093  AE        XOR     (HL)            ; toggle the sprite
3094  77        LD      (HL),A          ; store result - toggles the girl to make her wiggle randomly
3095  C9        RET                     ; return

; called from #3082 and #3088 above

3096  0602      LD      B,#02           ; For B = 1 to 2

3098  79        LD      A,C             ; A := C
3099  AE        XOR     (HL)            ; toggle with the bits in this memory location
309A  77        LD      (HL),A          ; store A into this location
309B  19        ADD     HL,DE           ; add offset for next location
309C  10FA      DJNZ    #3098           ; Next B

309E  C9        RET                     ; return

; insert task
; DE are loaded with task # and parameter
; tasks are decoded at #02E3
; tasks are pushed into #60C0 through #60FF

309F  E5        PUSH    HL              ; save HL
30A0  21C060    LD      HL,#60C0        ; load HL with start of task list [why?  L is set later, only H needs to be loaded
here]
30A3  3AB060    LD      A,(#60B0)       ; load A with task pointer
30A6  6F        LD      L,A             ; HL now has task pointer full address
30A7  CB7E      BIT     7,(HL)          ; test high bit 7 of the task at this address.  zero?
30A9  CABB30    JP      Z,#30BB         ; yes, skip ahead, restore HL and return. [when would this happen??? if task list is
full???]

30AC  72        LD      (HL),D          ; else store task number into task list
30AD  2C        INC     L               ; next HL
30AE  73        LD      (HL),E          ; store task parameter
30AF  2C        INC     L               ; next HL
30B0  7D        LD      A,L             ; load A with low byte of task pointer
30B1  FEC0      CP      #C0             ; is A > #C0 ? (did the task list roll over?)
30B3  D2B830    JP      NC,#30B8        ; no, skip next instruction

30B6  3EC0      LD      A,#C0           ; yes, reset A to #C0 for start of task list

30B8  32B060    LD      (#60B0),A       ; store A into task list pointer

30BB  E1        POP     HL              ; restore HL
30BC  C9        RET                     ; return to program

; arrive here from #1615 when rivets cleared
; clears all sprites for firefoxes, hammers and bonus items

30BD  215069    LD      HL,#6950        ; load HL with start of hammers
30C0  0602      LD      B,#02           ; B := 2
30C2  CDE430    CALL    #30E4           ; clear hammers ?
30C5  2E80      LD      L,#80           ; L := #80
30C7  060A      LD      B,#0A           ; B := #A
30C9  CDE430    CALL    #30E4           ; clear barrels ?
30CC  2EB8      LD      L,#B8           ; L := #B8
30CE  060B      LD      B,#0B           ; B := #B
30D0  CDE430    CALL    #30E4           ; clear firefoxes ?
30D3  210C6A    LD      HL,#6A0C        ; load HL with start of bonus items
30D6  0605      LD      B,#05           ; B := 5
30D8  C3E430    JP      #30E4           ; clear bonus items

; called from #12DF
; clears mario and elevators from the screen

30DB  214C69    LD      HL,#694C        ; load address for mario sprite X position
30DE  3600      LD      (HL),#00        ; clear this memory = move mario off screen
30E0  2E58      LD      L,#58           ; HL := #6958 = elevator sprite start
30E2  0606      LD      B,#06           ; for B = 1 to 6

30E4  7D        LD      A,L             ; load A with low byte addr

30E5  3600      LD      (HL),#00        ; clear this sprite position to zero = move off screen
30E7  C604      ADD     A,#04           ; add 4 for next sprite
30E9  6F        LD      L,A             ; store into HL
30EA  10F9      DJNZ    #30E5           ; next B

30EC  C9        RET                     ; return

; called from main routine at #198C

30ED  CDFA30    CALL    #30FA           ; Check internal difficulty and timers and return here based on difficulty a
percentage of the time
30F0  CD3C31    CALL    #313C           ; Deploy fire if fire deployment flag is set
30F3  CDB131    CALL    #31B1           ; Process all movement for all fireballs
30F6  CDF334    CALL    #34F3           ; update all fires and firefoxes
30F9  C9        RET                     ; return
```

```
; This routine is used to adjust the fireball speed based on the internal difficulty. It works by forcing the entire fireball
movement routine to
; be skipped on certain frames, returning directly back to the main routine in such cases. The higher the internal difficlty,
the less often it
; short-circuits back to the main routine, the faster they will move.
; called from #30ED ABOVE

30FA  3A8063    LD      A,(#6380)       ; \  Jump if internal difficulty is less than 6 (Is it possible to not jump here?)
30FD  FE06      CP      #06             ;  |
30FF  3802      JR      C,#3103         ; /

3101  3E05      LD      A,#05           ; load A with 5 = max internal difficulty
3103  EF        RST     #28             ; jump to address based on internal difficulty

3104  10 31                    0       ; #3110
3106  10 31                    1       ; #3110
3108  1B 31                    2       ; #311B
310A  26 31                    3       ; #3126
310C  26 31                    4       ; #3126
310E  31 31                    5       ; #3131

; internal difficulty == 0 or 1. In this case, the fireball movement routine is only executed every other frame, so that
fireballs move slowly.

3110  3A 1A 60  LD      A,(FrameCounter)       ; load A with this clock counts down from #FF to 00 over and over...
3112  60        LD      H,B             ; load H with B == ??? from previous subroutine ???? [what is this doing here ?]
3113  E601      AND     #01             ; \  If lowest bit of timer is 0 Return and continue as normal
3115  FE01      CP      #01             ;  |
3117  C8        RET     Z               ; /

3118  33        INC     SP              ; \  Else return to #198F instead of #30F0, skipping fireball movement routine
3119  33        INC     SP              ;  |
311A  C9        RET                     ; /

; internal difficulty == 2. Here the fireball movement routine is executed for 5 consecutive frames out of every 8 frames.

311B  3A1A60    LD      A,(FrameCounter)       ; \  If the lowest 3 bits of timer are less than 5 (equal to 0, 1, 2, 3, or 4)
then return and continue as
311E  E607      AND     #07             ;  | normal
3120  FE05      CP      #05             ;  |
3122  F8        RET     M               ; /

3123  33        INC     SP              ; \  Else return to #198F instead of #30F0, skipping fireball movement routine
3124  33        INC     SP              ;  |
3125  C9        RET                     ; /

; difficulty == 3 or 4. Here the fireball movement routine is executed for 3 out of every 4 frames.

3126  3A1A60    LD      A,(FrameCounter)       ; \  If the lowest 2 bits of the timer are not 11 then return and continue as
normal
3129  E603      AND     #03             ;  |
312B  FE03      CP      #03             ;  |
312D  F8        RET     M               ; /

312E  33        INC     SP              ; \  Else return to #198F instead of #30F0, skipping fireball movement routine
312F  33        INC     SP              ;  |
3130  C9        RET                     ; /

; difficulty == 5. Here the fireball movement routine is executed for 7 out of every 8 frames.

3131  3A1A60    LD      A,(FrameCounter)       ; \  If the lowest 3 bits of the timer are not 111 then return and continue as
normal
3134  E607      AND     #07             ;  |
3136  FE07      CP      #07             ;  |
3138  F8        RET     M               ; /

3139  33        INC     SP              ; \  Else return to #198F instead of #30F0, skipping fireball movement routine
313A  33        INC     SP              ;  |
313B  C9        RET                     ; /

; This routine checks the fire deployment flag and deploys the actual fireball if it is set (as long as there is a free
slot). It also keeps an
; updated count of the number of fireballs on screen and sets the color of fireballs based on the hammer status.
; called from #30F0

313C  DD210064  LD      IX,#6400        ; load IX with start of fire address
3140  AF        XOR     A               ; \ Reset # of fires onscreen to 0, this routine will count them.
3141  32A163    LD      (#63A1),A       ; /
3144  0605      LD      B,#05           ; For B = 1 to 5 firefoxes
3146  112000    LD      DE,#0020        ; load DE with offset to add for next firefox

3149  DD7E00    LD      A,(IX+#00)      ; \  Jump if sprite slot is unused to maybe deploy a fire there.
314C  FE00      CP      #00             ;  |
314E  CA7C31    JP      Z,#317C         ; /

3151  3AA163    LD      A,(#63A1)       ; \  This fire slot is active. Increment count for # of fires onscreen
3154  3C        INC     A               ;  |
3155  32A163    LD      (#63A1),A       ; /
3158  3E01      LD      A,#01           ; \  Set fire color to #01 (normal) if hammer is not active, and #00 (blue) if hammer
is active
315A  DD7708    LD      (IX+#08),A      ;  |
315D  3A1762    LD      A,(#6217)       ;  |
3160  FE01      CP      #01             ;  |
3162  C26A31    JP      NZ,#316A        ;  |
```

```
3165  3E00      LD      A,#00           ;  |
3167  DD7708    LD      (IX+#08),A      ; /

316A  DD19      ADD     IX,DE           ; next sprite
316C  10DB      DJNZ    #3149           ; next B

316E  21A063    LD      HL,#63A0        ; \ Clear fire deployment flag
3171  3600      LD      (HL),#00        ; /
3173  3AA163    LD      A,(#63A1)       ; \  Return all the way back to the main routine if no fires are active, otherwise
just return.
3176  FE00      CP      #00             ;  |
3178  C0        RET     NZ              ;  |
3179  33        INC     SP              ;  |
317A  33        INC     SP              ;  |
317B  C9        RET                     ; /

; arrive here from #314E
317C  3AA163    LD      A,(#63A1)       ; \  Jump back and don't deploy fire if there are already 5 fires active (Can this
ever happen here?)
317F  FE05      CP      #05             ;  |
3181  CA6A31    JP      Z,#316A         ; /
3184  3A2762    LD      A,(#6227)       ; \  Jump ahead if screen is not conveyors (i.e., the screen is rivets)
3187  FE02      CP      #02             ;  |
3189  C29531    JP      NZ,#3195        ; /
318C  3AA163    LD      A,(#63A1)       ; \  Return if current count of # of fires == internal difficulty, on conveyors we
never have more fireballs
318F  4F        LD      C,A             ;  | on screen than the internal difficulty
3190  3A8063    LD      A,(#6380)       ;  |
3193  B9        CP      C               ;  |
3194  C8        RET     Z               ; /
3195  3AA063    LD      A,(#63A0)       ; \  Jump back and don't deploy fire if fire deployment flag is not set
3198  FE01      CP      #01             ;  |
319A  C26A31    JP      NZ,#316A        ; /

319D  DD7700    LD      (IX+#00),A      ; Deploy a fire. Set status indicator to 1 = active
31A0  DD7718    LD      (IX+#18),A      ; Set spawning indicator to 1
31A3  AF        XOR     A               ; \ Clear fire deployment flag
31A4  32A063    LD      (#63A0),A       ; /
31A7  3AA163    LD      A,(#63A1)       ; \  Increment count of # of active fires
31AA  3C        INC     A               ;  |
31AB  32A163    LD      (#63A1),A       ; /
31AE  C36A31    JP      #316A           ; jump back and loop for next

; This subroutine handles all movement for all fireballs.
; called from #30F3

31B1  CDDD31    CALL    #31DD           ; Check if freezers should enter freezer mode
31B4  AF        XOR     A               ; \ Index of fireball being processed := 0
31B5  32A263    LD      (#63A2),A       ; /
31B8  21E063    LD      HL,#63E0        ; \ Address of fireball data array for current fireball being processed := #63E0 =
#6400 - #20
31BB  22C863    LD      (#63C8),HL      ; / This gets incremented by #20 at the start of the following loop

; Loop start
31BE  2AC863    LD      HL,(#63C8)      ; \  Move on to next fireball by incrementing address of fireball data array for
current fireball by #20
31C1  012000    LD      BC,#0020        ;  |
31C4  09        ADD     HL,BC           ;  |
31C5  22C863    LD      (#63C8),HL      ; /
31C8  7E        LD      A,(HL)          ; \  Jump if fireball is not active
31C9  A7        AND     A               ;  |
31CA  CAD031    JP      Z,#31D0         ; /

31CD  CD0232    CALL    #3202           ; Handle all movement for this fire

31D0  3AA263    LD      A,(#63A2)       ; \  Increment index of current fireball being processed
31D3  3C        INC     A               ;  |
31D4  32A263    LD      (#63A2),A       ; /
31D7  FE05      CP      #05             ; \ Loop if index is less than 5
31D9  C2BE31    JP      NZ,#31BE        ; /

31DC  C9        RET                     ; return

; This subroutine checks if fires 2 and 4 should enter freezer mode. They always both enter at the same time and they enter
with a 25% probability
; every 256 frames (note that this is 256 actual frames, not 256 fireball code execution frames).
; called from #31B1 above

31DD  3A8063    LD      A,(#6380)       ; \  Return if internal difficulty is < 3, no freezers are allowed until difficulty
3.
31E0  FE03      CP      #03             ;  |
31E2  F8        RET     M               ; /

31E3  CDF631    CALL    #31F6           ; Check if we should enter freezer mode (25% probability every 256 frames of entering
freezer mode)
31E6  FE01      CP      #01             ; \ Return if should not enter freezer mode
31E8  C0        RET     NZ              ; /

31E9  213964    LD      HL,#6439        ; \  Set freezer indicator of 2nd fire to #02 to enable freezer mode
31EC  3E02      LD      A,#02           ;  |
31EE  77        LD      (HL),A          ; /

31EF  217964    LD      HL,#6479        ; \  Set freezer indicator of 4th fire to #02 to enable freezer mode
31F2  3E02      LD      A,#02           ;  |
31F4  77        LD      (HL),A          ; /
```

```
31F5  C9        RET                     ; return

; Every 256 frames this subroutine has a 25% chance of loading 1 into A. Otherwise a value not equal to 1 is loaded.
; called from #31E3

31F6  3A1860    LD      A,(RngTimer1)       ; \  Return with 1 not loaded in A if lowest 2 bits of RNG are not 01. (75%
probability of returning)
31F9  E603      AND     #03             ;  |
31FB  FE01      CP      #01             ;  |
31FD  C0        RET     NZ              ; /

31FE  3A1A60    LD      A,(FrameCounter)       ; \ Else return A with timer that constantly counts down from FF to 00  ... 1
count per frame
3201  C9        RET                     ; /

; This subroutine handles all movement for a single fireball.
; called from #31CD above

3202  DD2AC863  LD      IX,(#63C8)      ; Load IX with address of fireball data array for current fireball
3206  DD7E18    LD      A,(IX+#18)      ; \  Jump if fireball is currently in the process of spawning
3209  FE01      CP      #01             ;  |
320B  CA7A32    JP      Z,#327A         ; /

320E  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball is currently on a ladder
3211  FE04      CP      #04             ;  |
3213  F23032    JP      P,#3230         ; /

3216  DD7E19    LD      A,(IX+#19)      ; \  Jump if freezer mode is enguaged for this fireball
3219  FE02      CP      #02             ;  |
321B  CA7E32    JP      Z,#327E         ; /

321E  CD0F33    CALL    #330F           ; Check if fireball should randomly reverse direction
3221  3A1860    LD      A,(RngTimer1)       ; \  Jump and do not climb any ladder with 75% probability, so a ladder is
climbed with 25% probability.
3224  E603      AND     #03             ;  | Note that left moving fireballs always skip the ladder climbing check and
instead jump to the end of
3226  C23332    JP      NZ,#3233        ; /  this subroutine without updating position.

3229  DD7E0D    LD      A,(IX+#0D)      ; \  Jump to end of subroutine if fireball is moving left. This is reached with 25%
probability so left-moving
322C  A7        AND     A               ;  | fireballs skip all movement with 25% probability, so their speed is randomized
but averages 25% slower
322D  CA5732    JP      Z,#3257         ; /  than the speed of right-moving fireballs.

; Fireball is on a ladder or about to mount ladder (as long as doing so is permitted).
3230  CD3D33    CALL    #333D           ; Handle fireball mounting/dismounting of ladders

3233  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball is currently on a ladder
3236  FE04      CP      #04             ;  |
3238  F29132    JP      P,#3291         ; /

; Fireball is moving left or right
323B  CDAD33    CALL    #33AD           ; Handle fire movement left or right, animate fireball, and adjust Y-position for
slanted girders
323E  CD8C29    CALL    #298C           ; Load A with 1 if girder edge nearby, 0 otherwise
3241  FE01      CP      #01             ; \ Jump if we have reached the edge of a girder
3243  CA9732    JP      Z,#3297         ; /

3246  DD2AC863  LD      IX,(#63C8)      ; Load IX with address of fireball slot for this fireball
324A  DD7E0E    LD      A,(IX+#0E)      ; \  Jump if X-position is < #10 (i.e., fireball has reached left edge of screen)
324D  FE10      CP      #10             ;  |
324F  DA8C32    JP      C,#328C         ; /

3252  FEF0      CP      #F0             ; \ Jump if X-position is >= #F0 (i.e., fireball has reached right edge of screen)
3254  D28432    JP      NC,#3284        ; /

3257  DD7E13    LD      A,(IX+#13)      ; \  Jump if our index into the Y-position adjustment table hasn't reached 0 yet
325A  FE00      CP      #00             ;  |
325C  C2B932    JP      NZ,#32B9        ; /

325F  3E11      LD      A,#11           ; Reset index into Y-position adjustment table

3261  DD7713    LD      (IX+#13),A      ; Store updated index into Y-position adjustment table
3264  1600      LD      D,#00           ; \  Index the Y-position adjustment table using +#13 to get in A the amount to
adjust the Y-position by to
3266  5F        LD      E,A             ;  | make the fireball bob up and down
3267  217A3A    LD      HL,#3A7A        ;  |
326A  19        ADD     HL,DE           ;  |
326B  7E        LD      A,(HL)          ; /

        ; 3A7A:  FF 00 FF FF FE FE FE FE FE FE FE FE FE FE FF FF 00

326C  DD460E    LD      B,(IX+#0E)      ; \ Copy effective X-position into actual X-position (these two are always the same)
326F  DD7003    LD      (IX+#03),B      ; /
3272  DD4E0F    LD      C,(IX+#0F)      ; \  Compute the actual Y-position by adding the adjustment to the effective Y-
position
3275  81        ADD     A,C             ;  |
3276  DD7705    LD      (IX+#05),A      ; /
3279  C9        RET                     ; return

; Arrive from #320B when fireball is spawning
327A  CDBD32    CALL    #32BD           ; Handle fireball movement while spawning
327D  C9        RET                     ; return

; Arrive from #321B when freezer mode is enguaged
```

```
327E  CDD632    CALL    #32D6           ; Handle freezing fireball
3281  C32932    JP      #3229           ; Jump back to program

; Arrive from #3254 when fireball has reached right edge of screen
3284  3E02      LD      A,#02           ; Set direction to "special" left

3286  DD770D    LD      (IX+#0D),A      ; Store new direction, either 1 for right or 2 for left
3289  C35732    JP      #3257           ; Jump back

; Arrive from #324F when fireball has reached left edge of screen
328C  3E01      LD      A,#01           ; Set direction to right
328E  C38632    JP      #3286           ; Jump back

; Fireball is moving up or down a ladder
3291  CDE733    CALL    #33E7           ; Handle fireball movement up/down the ladder and animate the fireball
3294  C35732    JP      #3257           ; Jump back

; Arrived from #3243 when fire is at edge of girder
3297  DD2AC863  LD      IX,(#63C8)      ; Load IX with address of fireball slot for this fireball
329B  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball direction is left
329E  FE01      CP      #01             ;  |
32A0  C2B132    JP      NZ,#32B1        ; /

32A3  3E02      LD      A,#02           ; Set direction to "special" left
32A5  DD350E    DEC     (IX+#0E)        ; Decrement fireball X-position, make fireball move left

32A8  DD770D    LD      (IX+#0D),A      ; Store new direction, either 1 for right, or 2 for left
32AB  CDC333    CALL    #33C3           ; Since we just moved a pixel, adjust Y-position for slanted girders on barrel screen
32AE  C35732    JP      #3257           ; Jump back

32B1  3E01      LD      A,#01           ; Set direction to right
32B3  DD340E    INC     (IX+#0E)        ; Incremement fireball X-position, make fireball move right
32B6  C3A832    JP      #32A8           ; Jump back

; Arrived from #325C
32B9  3D        DEC     A               ; Decrement index into Y-position adjustment table
32BA  C36132    JP      #3261           ; Jump back

; This subroutine is responsible for handling fireball movement while the fireball is spawning. Here the fireball may be
following a fixed trajectory
; such as when jumping out of an oil can for example.
; called from #327A

32BD  3A2762    LD      A,(#6227)       ; \  Jump if we are currently on barrels
32C0  FE01      CP      #01             ;  |
32C2  CACE32    JP      Z,#32CE         ; /

32C5  FE02      CP      #02             ; \ Jump if we are on conveyors
32C7  CAD232    JP      Z,#32D2         ; /

32CA  CDB934    CALL    #34B9           ; Spawn fireball in proper location on rivets
32CD  C9        RET                     ; return

32CE  CD2C34    CALL    #342C           ; Handle fireball movement while coming out of oilcan on barrels
32D1  C9        RET                     ; return

32D2  CD7834    CALL    #3478           ; Handle fireball movement while coming out of oilcan on conveyors
32D5  C9        RET                     ; return

; This subroutine handles a freezer when freezer mode is activated, including checking when to freeze and when to leave
freezer mode.
; Called from #327E

32D6  DD7E1C    LD      A,(IX+#1C)      ; \  Jump if fireball freeze timer is non-zero, meaning we are frozen and waiting for
the timer to reach 0
32D9  FE00      CP      #00             ;  | to unfreeze.
32DB  C2FD32    JP      NZ,#32FD        ; /

32DE  DD7E1D    LD      A,(IX+#1D)      ; \  We reach this when a fireball is not frozen, but freezer mode is activated. Jump
if the freeze flag is
32E1  FE01      CP      #01             ;  | not set (This flag is only set when the fireball reaches the top of a ladder).
32E3  C20B33    JP      NZ,#330B        ; /

; It is time to maybe freeze the fireball at the top of a ladder.
32E6  DD361D00  LD      (IX+#1D),#00    ; Reset the freeze flag to zero
32EA  3A0562    LD      A,(#6205)       ; \  Jump if Mario is above fireball, in this case we leave freezer mode immediately
without freezing.
32ED  DD460F    LD      B,(IX+#0F)      ;  |
32F0  90        SUB     B               ;  |
32F1  DA0333    JP      C,#3303         ; /

32F4  DD361CFF  LD      (IX+#1C),#FF    ; Freeze the fireball for 256 fireball execution frames

32F8  DD360D00  LD      (IX+#0D),#00    ; Set direction to "frozen"
32FC  C9        RET                     ; return

; Jump here from #32DB when fireball still frozen
32FD  DD351C    DEC     (IX+#1C)        ; Decrement freeze timer
3300  C2F832    JP      NZ,#32F8        ; Jump if it is still not time to unfreeze

; It is time to unfreeze
3303  DD361900  LD      (IX+#19),#00    ; Clear the freezer mode flag
3307  DD361C00  LD      (IX+#1C),#00    ; Clear the freeze timer
```

```
330B  CD0F33    CALL    #330F           ; Check if fireball should randomly freeze out in the open (note this is the same as
the direction reversal
                                        ; routine for non-freezing fireballs, only now setting direction to 00 indicates
"frozen" instead of "left")
330E  C9        RET                     ; return

; This subroutine randomly reversed direction of fire every 43 fireball execution frames. Note that this is not actual
frames, the actual number of
; frames will vary based on internal difficulty.
; called from #321E and from #330B

330F  DD7E16    LD      A,(IX+#16)      ; \  Jump without reversing if direction reverse timer hasn't reached 0 yet
3312  FE00      CP      #00             ;  |
3314  C23233    JP      NZ,#3332        ; /

3317  DD36162B  LD      (IX+#16),#2B    ; Reset direction reverse counter to #2B
331B  DD360D00  LD      (IX+#0D),#00    ; \  Set fireball direction to be left (or frozen for freezers) and jump with 50%
probability
331F  3A1860    LD      A,(RngTimer1)       ;  |
3322  0F        RRCA                    ;  |
3323  D23233    JP      NC,#3332        ; /

3326  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if direction fireball direction is 1, which is impossible, so this is a
NOP.
3329  FE01      CP      #01             ;  |
332B  CA3633    JP      Z,#3336         ; /

332E  DD360D01  LD      (IX+#0D),#01    ; Else set fireball direction to be right
3332  DD3516    DEC     (IX+#16)        ; Decrement direction reverse timer
3335  C9        RET                     ; return

; jump here from #332B [never arrive here , buggy software]
3336  DD360D02  LD      (IX+#0D),#02    ; Set fireball direction to be "special" left
333A  C33233    JP      #3332           ; jump back

; This subroutine serves two purposes. If a fireball is currently on a ladder it checks to see if the fireball has reached
the other end of the ladder
; and if so dismounts the ladder. Otherwise, if the fireball is not on a ladder it checks to see if there are any ladders
nearby that can be taken,
; and if so it mounts the ladder.
; called from #3230

333D  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball is climbing up a ladder
3340  FE08      CP      #08             ;  |
3342  CA7133    JP      Z,#3371         ; /

3345  FE04      CP      #04             ; \ Jump if fireball is climbing down a ladder
3347  CA8A33    JP      Z,#338A         ; /

; Else firefox is not on a ladder, but will mount one if permitted to do so
334A  CDA133    CALL    #33A1           ; Return without taking ladder if fireball is on the top girder and the screen is not
rivets
334D  DD7E0F    LD      A,(IX+#0F)      ; \  D := Y-position of bottom of fireball
3350  C608      ADD     A,#08           ;  |
3352  57        LD      D,A             ; /
3353  DD7E0E    LD      A,(IX+#0E)      ; A := fireball's X-position
3356  011500    LD      BC,#0015        ; BC := #0015, the number of ladders to check
3359  CD6E23    CALL    #236E           ; Check for ladders nearby, return if none, else A := 0 if at bottom of ladder, A :=
1 if at top
335C  A7        AND     A               ; \ Jump if there is a ladder nearby to go up
335D  CA9933    JP      Z,#3399         ; /

; Else there is a ladder nearby to go down
3360  DD701F    LD      (IX+#1F),B      ; Store B into +#1F = Y-position of bottom of ladder
3363  3A0562    LD      A,(#6205)       ; \  Return without taking the ladder if Mario is at or above the Y-position of the
fireball
3366  47        LD      B,A             ;  |
3367  DD7E0F    LD      A,(IX+#0F)      ;  |
336A  90        SUB     B               ;  |
336B  D0        RET     NC              ; /

336C  DD360D04  LD      (IX+#0D),#04    ; Else set direction to descending ladder
3370  C9        RET                     ; return

; Arrived because fireball is moving up a ladder
3371  DD7E0F    LD      A,(IX+#0F)      ; \  Return if fireball is not at the top of the ladder
3374  C608      ADD     A,#08           ;  |
3376  DD461F    LD      B,(IX+#1F)      ;  |
3379  B8        CP      B               ;  |
337A  C0        RET     NZ              ; /

; Fireball at top of ladder
337B  DD360D00  LD      (IX+#0D),#00    ; Set fireball direction to left
337F  DD7E19    LD      A,(IX+#19)      ; \  If freezer mode is enguaged then set the freeze flag and return, otherwise just
return.
3382  FE02      CP      #02             ;  |
3384  C0        RET     NZ              ;  |
3385  DD361D01  LD      (IX+#1D),#01    ;  |
3389  C9        RET                     ; /

; Arrive because fireball is moving down a ladder
338A  DD7E0F    LD      A,(IX+#0F)      ; \  Return if fireball is not at the bottom of the ladder
338D  C608      ADD     A,#08           ;  |
338F  DD461F    LD      B,(IX+#1F)      ;  |
3392  B8        CP      B               ;  |
```

```
3393  C0        RET     NZ              ; /

3394  DD360D00  LD      (IX+#0D),#00    ; Fireball has reached the bottom, set the direction to left
3398  C9        RET                     ; return

; Arrive because there is a ladder nearby to go up
3399  DD701F    LD      (IX+#1F),B      ; Store B into +#1F = Y-position of top of ladder
339C  DD360D08  LD      (IX+#0D),#08    ; Else set direction to ascending ladder
33A0  C9        RET                     ; return

; This subroutine returns to the higher subroutine (causing a ladder to NOT be taken) if a fireball is on the top girder and
we are not on rivets.
; called from #334A

33A1  3E07      LD      A,#07           ; \ Return if immediately we are on rivets, fireballs do not get stuck on the top in
this case
33A3  F7        RST     #30             ; /

33A4  DD7E0F    LD      A,(IX+#0F)      ; \ Return if Y-position is >= 59 (i.e., fireball is not on the top girder)
33A7  FE59      CP      #59             ;  |
33A9  D0        RET     NC              ; /

33AA  33        INC     SP              ; \  Else return to higher subroutine. This prevents fireballs from coming down on
conveyors & girders once
33AB  33        INC     SP              ;  | they reach the top level.
33AC  C9        RET                     ; /

; This subroutine handles movemnt of a fireball to the left and right. It also animates the fireball and adjusts its Y-
position if travelling up/down
; a slanted girder on the barrel screen.
; called from #323B

33AD  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball direction is right
33B0  FE01      CP      #01             ;  |
33B2  CAD933    JP      Z,#33D9         ; /

; Fireball is moving left
33B5  DD7E07    LD      A,(IX+#07)      ; \  Set direction bit in fireball graphics to face left
33B8  E67F      AND     #7F             ;  |
33BA  DD7707    LD      (IX+#07),A      ; /
33BD  DD350E    DEC     (IX+#0E)        ; Decrement X-position

33C0  CD0934    CALL    #3409           ; Animate the fireball
; Fall into below subroutine

; This subroutine adjusts a fireball's Y-position based on movement up/down a slanted girder on the barrel screen.
; called from #32AB

33C3  3A2762    LD      A,(#6227)       ; \  Return if we are not on barrels
33C6  fe01      CP      #01             ;  |
33C8  c0        RET     NZ              ; /

33C9  DD660E    LD      H,(IX+#0E)      ; Load H with fireball X-position
33CC  DD6E0F    LD      L,(IX+#0F)      ; Load L with fireball Y-position
33CF  DD460D    LD      B,(IX+#0D)      ; Load B with fireball direction
33D2  CD3323    CALL    #2333           ; Check for fireball moving up/down a slanted girder ?
33D5  DD750F    LD      (IX+#0F),L      ; Store adjusted Y-position
33D8  C9        RET                     ; return

; Fireball is moving right
33D9  DD7E07    LD      A,(IX+#07)      ; \  Set direction bit in fireball graphics to face right
33DC  F680      OR      #80             ;  |
33DE  DD7707    LD      (IX+#07),A      ; /
33E1  DD340E    INC     (IX+#0E)        ; Increment X-position
33E4  C3C033    JP      #33C0           ; Jump back to program

; This subroutine handles fireball movement up and down ladders. Fireball movement up a ladder is 1/3 the speed of movement
down a ladder, and
; movement down a ladder is the same speed as movement to the right. The subroutine also animates the fireball as it climbs.
; called from #3291

33E7  CD0934    CALL    #3409           ; Animate the fireball
33EA  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball is moving down the ladder
33ED  FE08      CP      #08             ;  |
33EF  C20534    JP      NZ,#3405        ; /

33F2  DD7E14    LD      A,(IX+#14)      ; \  Jump if it is not time to climb one pixel yet
33F5  A7        AND     A               ;  |
33F6  C20134    JP      NZ,#3401        ; /

33F9  DD361402  LD      (IX+#14),#02    ; Reset ladder climb timer to 2
33FD  DD350F    DEC     (IX+#0F)        ; Decrement fireball's Y position, move up one pixel
3400  C9        RET                     ; return

3401  DD3514    DEC     (IX+#14)        ; Decrease ladder climb timer
3404  C9        RET                     ; return

3405  DD340F    INC     (IX+#0F)        ; Increment fireball's Y position, move down one pixel
3408  C9        RET                     ; return

; This subroutine handles fireball animation.
; called from #33E7 and from #33C0

3409  DD7E15    LD      A,(IX+#15)      ; \  Jump if it is not time to change animation frames yet
340C  A7        AND     A               ;  |
```

```
340D  C22834    JP      NZ,#3428        ; /

3410  DD361502  LD      (IX+#15),#02    ; Reset animation change timer
3414  DD3407    INC     (IX+#07)        ; \  Toggles the lowest 4 bits of +#07 between D and E, this toggles between two
possible graphics that
3417  DD7E07    LD      A,(IX+#07)      ;  | the fireball can use
341A  E60F      AND     #0F             ;  |
341C  FE0F      CP      #0F             ;  |
341E  C0        RET     NZ              ;  |
341F  DD7E07    LD      A,(IX+#07)      ;  |
3422  EE02      XOR     #02             ;  |
3424  DD7707    LD      (IX+#07),A      ; /
3427  C9        RET                     ; return

3428  DD3515    DEC     (IX+#15)        ; Decrement animation change timer
342B  C9        RET                     ; return

; The subroutine handles fireball movement as it spawns out of the oilcan on barrels.
; Called from #32CE

342C  DD6E1A    LD      L,(IX+#1A)      ; \ Load HL with address into Y-position table
342F  DD661B    LD      H,(IX+#1B)      ; /
3432  AF        XOR     A               ; \  Jump if HL is non-zero (i.e., if this is not the very first spawning frame)
3433  010000    LD      BC,#0000        ;  |
3436  ED4A      ADC     HL,BC           ;  |
3438  C24234    JP      NZ,#3442        ; /

343B  218C3A    LD      HL,#3A8C        ; We just began to spawn, load HL with address of start of Y-position table
343E  DD360326  LD      (IX+#03),#26    ; Initialize X position to #26, the X-position of the oilcan

; This table stores the Y-positions a fireball should have each frame to follow a parabolic arc used when fireballs are
coming out of oilcans.
        ; 3A8C:  E8 E5 E3 E2
        ; 3A90:  E1 E0 DF DE DD DD DC DC DC DC DC DC DD DD DE DF
        ; 3AA0:  E0 E1 E2 E3 E4 E5 E7 E9 EB ED F0 AA

3442  DD3403    INC     (IX+#03)        ; Increment X-position

3445  7E        LD      A,(HL)          ; \  Jump if we've reached the end of the Y-position table (marked by #AA)
3446  FEAA      CP      #AA             ;  |
3448  CA5634    JP      Z,#3456         ; /

344B  DD7705    LD      (IX+#05),A      ; Else store table data into fire's Y-position
344E  23        INC     HL              ; \  Advance to next table entry, for the next frame
344F  DD751A    LD      (IX+#1A),L      ;  |
3452  DD741B    LD      (IX+#1B),H      ; /
3455  C9        RET                     ; return

; Fire has completed its spawning and is now free-floating
3456  AF        XOR     A               ; A := 0
3457  DD7713    LD      (IX+#13),A      ; Clear fire animation height counter
345A  DD7718    LD      (IX+#18),A      ; Clear firefox spawning indicator
345D  DD770D    LD      (IX+#0D),A      ; Set direction to left
3460  DD771C    LD      (IX+#1C),A      ; Clear the still indicator
3463  DD7E03    LD      A,(IX+#03)      ; \ Make copy of X-position
3466  DD770E    LD      (IX+#0E),A      ; /
3469  DD7E05    LD      A,(IX+#05)      ; \ Make copy of Y-position
346C  DD770F    LD      (IX+#0F),A      ; /
346F  DD361A00  LD      (IX+#1A),#00    ; \ Clear address into Y-position spawning table
3473  DD361B00  LD      (IX+#1B),#00    ; / [these last two could have been written above with one less byte each]
3477  C9        RET                     ; return

; This subroutine handles fireball movement as it spawns out of the oilcan on conveyors.
; Called from #32D2

3478  DD6E1A    LD      L,(IX+#1A)      ; \ Load HL with address into Y-position table
347B  DD661B    LD      H,(IX+#1B)      ; /
347E  AF        XOR     A               ; \  Jump if HL is non-zero (i.e., if this is not the very first spawning frame)
347F  010000    LD      BC,#0000        ;  |
3482  ED4A      ADC     HL,BC           ;  |
3484  C29A34    JP      NZ,#349A        ; /

3487  21AC3A    LD      HL,#3AAC        ; load HL with start of table data
348A  3A0362    LD      A,(#6203)       ; \  Jump if Mario is on left side of the screen, in this case we spawn the fireball
on the left
348D  CB7F      BIT     7,A             ;  |
348F  CAA834    JP      Z,#34A8         ; /

3492  DD360D01  LD      (IX+#0D),#01    ; Set fireball direction to "right"
3496  DD36037E  LD      (IX+#03),#7E    ; Initialize X position to #7E

349A  DD7E0D    LD      A,(IX+#0D)      ; \  Jump if fireball moving left
349D  FE01      CP      #01             ;  |
349F  C2B334    JP      NZ,#34B3        ; /

34A2  DD3403    INC     (IX+#03)        ; Moving right, Increment X-position
34A5  C34534    JP      #3445           ; Jump back, remainder of subroutine shared with the above subroutine

34A8  DD360D02  LD      (IX+#0D),#02    ; Set fireball direction to "special" left (This isn't actually used at all after
spawning, since immediately
                                        ; after spawning it will check to reverse rection and receive a direction of either
"right" or "left".
34AC  DD360380  LD      (IX+#03),#80    ; Initialize X position to #80
34B0  C39A34    JP      #349A           ; Jump back [why there?  after setting direction, we should jump directly to #34B3]
```

```
34B3  DD3503    DEC     (IX+#03)        ; Moving left, Decrement X-position
34B6  C34534    JP      #3445           ; Jump back, remainder of subroutine shared with the above subroutine

; On rivets, this subroutine spawns a fireball on a random platform besides the very top on the side of the screen opposite
the the side that Mario
; is on.
; Called from #32CA when screen is elevators or rivets

34B9  3A2762    LD      A,(#6227)       ; \  Return if current screen is elevators (Can this ever happen?)
34BC  fe03      CP      #03             ;  |
34Be  c8        RET     Z               ; /

34Bf  3A0362    LD      A,(#6203)       ; \  Jump if bit 7 of Mario's X-position is set (i.e., Mario is on the right half of
the screen)
34C2  cb7f      BIT     7,A             ;  |
34C4  C2ED34    JP      NZ,#34ED        ; /

34C7  21C43A    LD      HL,#3AC4        ; Load HL with start of table data for spawning fireball on right side

; Possible X and Y positions to spawn a fireball on the right side of the screen
; First value is X position, 2nd value is Y position

; 3AC4:  EE F0  ; bottom, right
; 3AC6:  DB A0  ; middle, right
; 3AC8:  E6 C8  ; 2nd from bottom, right
; 3ACA:  D6 78  ; 2nd from top, right
; 3ACC:  EB F0  ; unused?
; 3ACE:  DB A0  ; unused?
; 3AD0:  E6 C8  ; unused?
; 3AD2:  E6 C8  ; unused?

; Possible X and Y positions to spawn a fireball on the left side of the screen
; First value is X position, 2nd value is Y position

; 3AD4:  1B C8  ; 2nd from bottom, left
; 3AD6:  23 A0  ; middle, left
; 3AD8:  2B 78  ; 2nd from top, left
; 3ADA:  12 F0  ; bottom, left
; 3ADC:  1B C8  ; unused?
; 3ADE:  23 A0  ; unused?
; 3AE0:  12 F0  ; unused?
; 3AE2:  1B C8  ; unused?



34CA  0600      LD      B,#00           ; \  Load BC with one of #0000, #0002, #0004, or #0006 randomly
34CC  3A1960    LD      A,(RngTimer2)       ;  |
34CF  E606      AND     #06             ;  |
34D1  4F        LD      C,A             ; /
34D2  09        ADD     HL,BC           ; add this result into HL to get offset into table
34D3  7E        LD      A,(HL)          ; \  Copy X-position from table into fireball X-position
34D4  DD7703    LD      (IX+#03),A      ;  |
34D7  DD770E    LD      (IX+#0E),A      ; /
34DA  23        INC     HL              ; next table entry
34DB  7E        LD      A,(HL)          ; \  Copy Y-position from table into fireball Y-position
34DC  DD7705    LD      (IX+#05),A      ;  |
34DF  DD770F    LD      (IX+#0F),A      ; /
34E2  AF        XOR     A               ; A := 0
34E3  DD770D    LD      (IX+#0D),A      ; Set fireball direction to left
34E6  DD7718    LD      (IX+#18),A      ; Clear fireball spawning indicator
34E9  DD771C    LD      (IX+#1C),A      ; Clear +1C = still indicator
34EC  C9        RET                     ; return

34ED  21D43A    LD      HL,#3AD4        ; Load HL with alternate start of table data for spawning fireball on left side.
34F0  C3CA34    JP      #34CA           ; Jump back

; update fires or firefoxes to hardware
; called from #30F6

34F3  210064    LD      HL,#6400        ; start of fire/firefox data
34F6  11D069    LD      DE,#69D0        ; start of firefox sprites (hardware)
34F9  0605      LD      B,#05           ; For B = 1 to 5

34FB  7E        LD      A,(HL)          ; get firefox data
34FC  A7        AND     A               ; is this sprite active ?
34FD  CA1E35    JP      Z,#351E         ; no, jump away and set for next sprite

3500  2C        INC     L
3501  2C        INC     L
3502  2C        INC     L               ; HL now points to firefox's X position (IX + #03)
3503  7E        LD      A,(HL)          ; load A with firefox X position
3504  12        LD      (DE),A          ; store into sprite X position
3505  3E04      LD      A,#04           ; A := 4
3507  85        ADD     A,L             ; add to L
3508  6F        LD      L,A             ; HL now points to firefox's Y position (IX + #07)
3509  1C        INC     E               ; next DE, now it has sprite Y position
350A  7E        LD      A,(HL)          ; load A with firefox Y position
350B  12        LD      (DE),A          ; store into hardaware sprite Y position
350C  2C        INC     L               ; next HL
350D  1C        INC     E               ; next DE
350E  7E        LD      A,(HL)          ; load A with firefox sprite color value
350F  12        LD      (DE),A          ; store sprite color
3510  2D        DEC     L
3511  2D        DEC     L
3512  2D        DEC     L               ; decrease HL by 3.  now it points to sprite value
```

```
3513  1C        INC     E               ; next DE
3514  7E        LD      A,(HL)          ; load A with sprite value
3515  12        LD      (DE),A          ; store sprite value to hardware
3516  13        INC     DE              ; next DE

3517  3E1B      LD      A,#1B           ; A := #1B
3519  85        ADD     A,L             ; add to L
351A  6F        LD      L,A             ; store into L.  HL how has #1B more.  The next sprite is referenced
351B  10DE      DJNZ    #34FB           ; Next Firefox

351D  C9        RET                     ; return

; arrive here when firefox is not being used, sets pointer for next sprite

351E  3E05      LD      A,#05           ; A := 5
3520  85        ADD     A,L             ; add to L
3521  6F        LD      L,A             ; store into L.  HL is now 5 more than before
3522  3E04      LD      A,#04           ; A := 4
3524  83        ADD     A,E             ; add to E
3525  5F        LD      E,A             ; store into E.  DE is now 4 more than before.  next sprite
3526  C31735    JP      #3517           ; jump back

; table data
; used for item scoring :  100, 200 , 300 etc
; called from #0525


3529:  00 00 00
352C:  00 01 00
352F:  00 02 00
3532:  00 03 00
3535:  00 04 00
3538:  00 05 00
353B:  00 06 00
353E:  00 07 00
3541:  00 08 00
3544:  00 09 00
3547:  00 00 00
354A:  00 10 00
354D:  00 20 00
3550:  00 30 00
3553:  00 40 00
3556:  00 50 00
3559:  00 60 00
355C:  00 70 00
355F:  00 80 00
3562:  00 90 00

;  table data .. loaded at #025A when game is powered on or reset
; transferred into #6100 to #61AA
; high score table

; first 2 bytes form a VRAM address. EG #7794 through #779C
; 3rd byte is the place.  1 through 5
; 4th and 5th bytes are either "ST" or "ND" or "RD" or "TH"
; 6th and 7th bytes are #10 for blank spaces
; 8th through 13th bytes are teh score digits
; 14 through end are #10 for blank spaces, ended by #3F end code
; after this is the actual score
; the last 2 bytes are ???

3565:  94 77 01 23 24 10 10 00 00 07 06 05 00 10 10 10 10 10 10 10 10 10 10 10 10 10 3F 00 50 76 00 F4 76
3587:  96 77 02 1E 14 10 10 00 00 06 01 00 00 10 10 10 10 10 10 10 10 10 10 10 10 10 3F 00 00 61 00 F6 76
35A9:  98 77 03 22 14 10 10 00 00 05 09 05 00 10 10 10 10 10 10 10 10 10 10 10 10 10 3F 00 50 59 00 F8 76
35CB:  9A 77 04 24 18 10 10 00 00 05 00 05 00 10 10 10 10 10 10 10 10 10 10 10 10 10 3F 00 50 50 00 FA 76
35ED:  9C 77 05 24 18 10 10 00 00 04 03 00 00 10 10 10 10 10 10 10 10 10 10 10 10 10 3F 00 00 43 00 FC 76
35ED:  9C 77 05 24 18 10 10 14 15 23 19 17 1E 10 12 29 10 20 11 25 1C 10 17 1F 15 23 10 3F 00 00 43 00 FC 76
                           D  E  S  I  G  N     B  Y     P  A  U  L     G  O  E  S
; data read at #1611
; used for high score entry ???

360F:                                              3B
3610:  5C 4B 5C 5B 5C 6B 5C 7B 5C 8B 5C 9B 5C AB 5C BB
3620:  5C CB 5C 3B 6C 4B 6C 5B 6C 6B 6C 7B 6C 8B 6C 9B
3630:  6C AB 6C BB 6C CB 6C 3B 7C 4B 7C 5B 7C 6B 7C 7B
3640:  7C 8B 7C 9B 7C AB 7C BB 7C CB 7C

; #364B is used from #05E9

364B:  8B 36            0       ; #368B "GAME OVER"
364D:  01 00            1       ; unused ?
364D:  3E 37            1       ; #373E "V1.00"
364F:  98 36            2       ; #3698 "PLAYER <I>"
3651:  A5 36            3       ; #36A5 "PLAYER <II>"
3653:  B2 36            4       ; #36B2 "RNDMZR"
3655:  BF 36            5       ; #36BF "CREDIT"
3657:  06 00            6       ; unused ?
3659:  CC 36            7       ; #36CC "RANDOMIZING THE BOARD!"
365B:  08 00            8       ; unused ?
365D:  E6 36            9       ; #36E6 "ONLY 1 PLAYER BUTTON"
365F:  FD 36            A       ; #36FD "1 OR 2 PLAYERS BUTTON"
3661:  0B 00            B       ; unused ?
3663:  15 37            C       ; #3715 "PUSH"
3665:  1C 37            D       ; #371C "NAME REGISTRATION"
3667:  30 37            E       ; #3730 "NAME:"
```

```
3669:  38 37            F       ; #3738 "---"
366B:  47 37            10      ; #3747 "A" through "J"
366D:  5D 37            11      ; #375D "K through "T"
366F:  73 37            12      ; #3773 "U" through "Z" and "RUBEND"
3671:  8B 37            13      ; #378B "REGI TIME"
3673:  00 61            14      ; #6100 High score entry 1 ?
3675:  22 61            15      ; #6122 High score entry 2 ?
3677:  44 61            16      ; #6144 High score entry 3 ?
3679:  66 61            17      ; #6166 High score entry 4 ?
367B:  88 61            18      ; #6188 High score entry 5?
367D:  9E 37            19      ; #379E "RANK SCORE NAME"
367D:  9D 37            19      ; #379D "# SCORE LEVEL NAME"
367F:  B6 37            1A      ; #37B6 "YOUR NAME WAS REGISTERED"
3681:  D2 37            1B      ; #37D2 "INSERT COIN"
3683:  E1 37            1C      ; #37E1 "PLAYER    COIN"
3685:  1D 00            1D      ; unused ?
3687:  00 3F            1E      ; #3F00 "(C) 1981-2022 NINTENDO"
3689:  09 3F            1F      ; #3F09 "NINTENDO OF AMERICA"
3689:  18 3F            1F      ; #3F18 "RNDMZR"

368A:                                   96 76 17 11 1D              ..GAM
3690:  15 10 10 1F 26 15 22 3F 94 76 20 1C 11 29 15 22  E..OVER...PLAYER
36A0:  10 30 32 31 3F 94 76 20 1C 11 29 15 22 10 30 33  .<I>...PLAYER.<2

36B0:  31 3F 80 76 18 19 17 18 10 23 13 1F 22 15 3F 9F  >...HIGH.SCORE..
36B0:  31 3F 80 76 10 10 22 1E 14 1D 2A 22 10 10 3F 9F  >...RNDMZR..

36C0:  75 13 22 15 14 19 24 10 10 10 10 3F 5E 77 18 1F  .CREDIT.......HO
36C0:  75 13 22 15 14 19 24 10 10 10 10 3F 55 77 22 11  .CREDIT.....RA

36D0:  27 10 18 19 17 18 10 13 11 1E 10 29 1F 25 10 17  W.HIGH.CAN.YOU.G
36D0:  1E 14 1F 1D 19 2A 19 1E 17 10 24 18 15 10 12 1F  NDOMIZING.THE.BO

36E0:  15 24 10 FB 10 3F 29 77 1F 1E 1C 29 10 01 10 20  ET.?....ONLY.1.P
36E0:  11 22 14 10 37 3F 29 77 1F 1E 1C 29 10 01 10 20  ARD.!...ONLY.1.P

36F0:  1C 11 29 15 22 10 12 25 24 24 1F 1E 3F 29 77 01  LAYER.BUTTON...1
3700:  10 1F 22 10 02 10 20 1C 11 29 15 22 23 10 12 25  .OR.2.PLAYERS.BU
3710:  24 24 1F 1E 3F 27 76 20 25 23 18 3F 06 77 1E 11  TTON...PUSH...NA
3720:  1D 15 10 22 15 17 19 23 24 22 11 24 19 1F 1E 3F  ME.REGISTRATION.

3730:  88 76 1E 11 1D 15 2E 3F E9 75 2D 2D 2D 10 10 10  ..NAME:...---...
3730:  88 76 1E 11 1D 15 2E 3F E9 75 2D 2D 2D 3F 3D 76  ..NAME:...---...

3740:  10 10 10 10 10 10 3F 0B 77 11 10 12 10 13 10 14  .....A.B.C.D
3740:  26 01 2B 00 00 10 3F 0B 77 11 10 12 10 13 10 14  V1.00....A.B.C.D

3750:  10 15 10 16 10 17 10 18 10 19 10 1A 3F 0D 77 1B  .E.F.G.H.I.J...K
3760:  10 1C 10 1D 10 1E 10 1F 10 20 10 21 10 22 10 23  .L.M.N.O.P.Q.R.S
3770:  10 24 3F 0F 77 25 10 26 10 27 10 28 10 29 10 2A  .T...U.V.W.X.Y.Z
3780:  10 2B 10 2C 44 45 46 47 48 10 3F F2 76 22 15 17  ...-RUBEND...REG

3790:  19 10 24 19 1D 15 10 10 30 03 00 31 10 3F 92 77  I.TIME.........
3790:  19 10 24 19 1D 15 10 10 30 03 00 31 3F 52 77 FA  I.TIME.........#

37A0:  22 11 1E 1B 10 10 23 13 1F 22 15 10 10 1E 11 1D  RANK..SCORE..NAM
37A0:  10 10 23 13 1F 22 15 10 10 1C 15 26 15 1C 10 10  ..SCORE..LEVEL..

37B0:  15 10 10 10 10 3F 72 77 29 1F 25 22 10 1E 11 1D  E.......YOUR.NAM
37B0:  1E 11 1D 15 3F 3F 72 77 29 1F 25 22 10 1E 11 1D  NAME....YOUR.NAM

37C0:  15 10 27 11 23 10 22 15 17 19 23 24 15 22 15 14  E.WAS.REGISTERED
37D0:  42 3F A7 76 19 1E 23 15 22 24 10 13 1F 19 1E 10  ....INSERT.COIN.
37E0:  3F 0A 77 10 10 20 1C 11 29 15 22 10 10 10 10 13  .....PLAYER....C
37F0:  1F 19 1E 3F FC 76 49 4A 10 1E 19 1E 24 15 1E 14  OIN......NINTEND
3800:  1F 10 10 10 10 3F                                O.....

3806:  7C 75 01 09 08 01 3F

; table data used for game intro

380D:  02 97 38 68 38   ; top level where girl sits
3812:  02 DF 54 10 54   ; kongs level girder
3817:  02 EF 6D 20 6D   ; 2nd girder down
381C:  02 DF 8E 10 8E   ; 3rd girder down
3821:  02 EF AF 20 AF   ; 4th girder down
3826:  02 DF D0 10 D0   ; 5th girder down
382B:  02 EF F1 10 F1   ; bottom girder
3830:  00 53 18 53 54   ; kong's ladder (left)
3835:  00 63 18 63 54   ; kong's ladder (right)
383A:  00 93 38 93 54   ; ladder to reach girl
383F:  00 83 54 83 F1   ; long ladder (left)
3834:  00 93 54 93 F1   ; long ladder (right)
3849:  AA               ; end of data code

; table data
; used for timer graphic and zero score inside

384A:  8D 7D 8C
384D:  6F 00 7C
3850:  6E 00 7C
3853:  6D 00 7C
3856:  6C 00 7C
3859:  8F 7F 8E
```

```
; table data
; used for animation of kong

385C:  47 27 08 50
3860:  2F A7 08 50
3864:  3B 25 08 50
3868:  00 70 08 48
386C:  3B 23 07 40
3870:  46 A9 08 44
3874:  00 70 08 48
3878:  30 29 08 44
387C:  00 70 08 48
3880:  00 70 0A 48

; table data used to draw the girl from #0D7A and #0B2A

3884:  6F 10 09 23
3888:  6F 11 0A 33

; used for animation of kong

388C:  50 34 08 3C
3890:  00 35 08 3C
3894:  53 32 08 40
3898:  63 33 08 40
389C:  00 70 08 48
38A0:  53 36 08 50
38A4:  63 37 08 50
38A8:  6B 31 08 41
38AC:  00 70 08 48
38B0:  6A 14 0A 48

; used when kong jump at end of intro

38B4:              FD FD FD FD FD FD FD FE FE FE FE FE
38C0:  FE FF FF FF FF 00 00 01 01 01
38CA:  7F                      ; end code


; used when kong jumps to left during intro at #0B70

38CB:                                  FF FF FF FF FF
38D0:  00 FF 00 00 01 00 01 01 01 01 01 7F

; used after kong has jumped
; used in #0DA7.  end code is #AA

38DC:  04 7F F0 10 F0
38E1:  02 DF F2 70 F8
38E6:  02 6F F8 10 F8
38EB:  AA

38EC:  04 DF D0 90 D0
38F1:  02 DF DC 20 D1
38F6:  AA

38F7:  FF FF FF FF FF   ; unused ?

38FC:  04 DF A8 20 A8
3901:  04 5F B0 20 B0
3906:  02 DF B0 20 BB
390B:  AA

390C:  04 DF 88 30 88
3911:  04 DF 90 B0 90
3916:  02 DF 9A 20 8F
391B:  AA

391C:  04 BF 68 20 68
3921:  04 3F 70 20 70
3926:  02 DF 6E 20 79
3927:  AA

392C:  02 DF 58 A0 55   ; top right ledge angled down
3931:  AA

; this is table data
; used for animation of kong
; used from #2D24

3932:  00 70 08 44
3936:  2B AC 08 4C
393A:  3B AE 08 4C
393E:  3B AF 08 3C
3942:  4B B0 07 3C
3946:  4B AD 08 4C
394A:  00 70 08 44
394E:  00 70 08 44
3952:  00 70 08 44
3956:  00 70 0A 44

; used to animate kong

395A:  47 27 08 4C
395E:  2F A7 08 4C
```

```
3962:  3B 25 08 4C
3966:  00 70 08 44
396A:  3B 23 07 3C
396E:  4B 2A 08 3C
3972:  4B 2B 08 4C
3976:  2B AA 08 3C
397A:  2B AB 08 4C
397E:  00 70 0A 44

; used for kong's middle deploy

3982:  00 70 08 44
3986:  4B 2C 08 4C
398A:  3B 2E 08 4C
398E:  3B 2F 08 3C
3992:  2B 30 07 3C
3996:  2B 2D 08 4C
399A:  00 70 08 44
399E:  00 70 08 44
39A2:  00 70 08 44
39A6:  00 70 0A 44

; used in #2E3D on elevators
; used for bouncers; each is an offset that is added to the Y position as it moves

39AA:  FD FD FD FE FE FE FE FF FF 00 FF 00 00 01 00 01 01 02 02 02 02 03 03 03
39C2:  7F       ; end code

; used in #2D8C for barrel release

39C3:  1E 4E BB 4C D8 4E 59 4E 7F

; table data having to do with crazy barrels.
; used in #2D83

39CC  BB               ; for crazy barrels
39CD  4D               ;
39CE  7F               ; deployed when #7F

; table data
; kong is beating his chest

39CF:  47 27 08 50
39D3:  2D 26 08 50
39D7:  3B 25 08 50
39DA:  00 70 08 48
39DF:  3B 24 07 40
39E3:  4B 28 08 40
39E7:  00 70 08 48
39EA:  30 29 08 44
39EF:  00 70 08 48
39F3:  00 70 0A 48

; table data for animation of kong #28 bytes (40 decimal)
; used in #0445
; the kong is beating his chest with right leg lifted

39F7:  49 A6 08 50 2F A7 08 50 3B 25 08 50 00 70 08 48
3A07:  3B 24 07 40 46 A9 08 44 00 70 08 48 2B A8 08 40
3A17:  00 70 08 48 00 70 0A 48

; table data for upside down kong after rivets cleared
; used in #1870
; #28 bytes = 40 bytes decimal

3A1F:  73 A7 88 60
3A23:  8B 27 88 60
3A27:  7F 25 88 60
3A2B:  00 70 88 68
3A2F:  7F 24 87 70
3A33:  74 29 88 6C
3A37:  00 70 88 68
3A3B:  8A A9 88 6C
3A3F:  00 70 88 68
3A43:  00 70 8A 68

; table data
; used when rivets are cleared

3A47:  05 AF F0 50 F0 AA
3A4D:  05 AF E8 50 E8 AA
3A53:  05 AF E0 50 E0 AA
3A59:  05 AF D8 50 D8 AA
3A5F:  05 B7 58 48 58 AA

; this table is used for the various screen patterns for the levels
; code 1 = girders, 4 = rivets, 2 = pies, 3 = elevators
; used from #1947 and from #1799 and from #09BA

3A65:  01 04                  ; level 1
3A67:  01 03 04               ; level 2
3A6A:  01 02 03 04            ; level 3
3A6E:  01 02 01 03 04         ; level 4
3A73:  01 02 01 03 01 04      ; level 5 +
3A79:  7F                     ; end code
```

```
; table data referenced in #3267

3A7A:  FF 00 FF FF FE FE FE FE FE FE FE FE FE FE FF FF 00

; table data referenced in #343B

3A8C:  E8 E5 E3 E2
3A90:  E1 E0 DF DE DD DD DC DC DC DC DC DC DD DD DE DF
3AA0:  E0 E1 E2 E3 E4 E5 E7 E9 EB ED F0 AA

; table data refeernced in #
; controls the positions of fires coming out of the oil can on the conveyors

3AAC:  80 7B 78 76 74 73 72 71 70 70 6F 6F 6F 70 70 71 72 73 74 75 76 77 78
3AC3:  AA               ; end code

; table data referenced in #34C7

3AC4:  EE F0 DB A0 E6 C8 D6 78 EB F0 DB A0 E6 C8 E6 C8

; table data referenced in #34ED

3AD4:  1B C8 23 A0 2B 78 12 F0 1B C8 23 A0 12 F0 1B C8

; table data for screen 1 girders

3AE4:  02 97 38 68 38   ; top girder where girl sits
3AE9:  02 9F 54 10 54   ; girder where kong sits
3AED:  02 DF 58 A0 55   ; 1st slanted girder at top right
3AF3:  02 EF 6D 20 79   ; 2nd slanted girder (has hammer at left side)
3AF8:  02 DF 9A 10 8E   ; 3rd slanted girder
3AFD:  02 EF AF 20 DB   ; 4th slanted girder
3B02:  02 DF DC 10 D0   ; 5th slanted girder (has hammer at right side)
3B07:  02 FF F0 80 F7   ; bottom slanted girder
3B0C:  02 7F F8 00 F8   ; bottom flat girder where mario starts
3B11:  00 CB 57 CB 6F   ; short ladder at top right
3B16:  00 CB 99 CB D1   ; short ladder at center right
3B1B:  00 CB DB CB F3   ; short ladder at bottom right
3B20:  00 63 18 63 54   ; kong's ladder (right)
3B25:  01 63 D5 63 F8   ; bottom broken ladder
3B2A:  00 33 78 33 90   ; short ladder at left side under top hammer
3B2F:  00 33 BA 33 D2   ; short ladder at left side above oil can
3B34:  00 53 18 53 54   ; kong's ladder (left)
3AF8:  02 DF DC 10 D0   ;
3AFD:  02 FF F0 80 F7   ; bottom slanted girder
3B02:  02 7F F8 00 F8   ; bottom flat girder where mario starts
3B07:  00 CB 57 CB 6F   ; short ladder at top right
3B0C:  00 CB DB CB F3   ; short ladder at bottom right
3B11:  00 63 18 63 54   ; kong's ladder (right)
3B16:  01 6B D5 6B F8   ;
3B1B:  00 53 18 53 54   ;
3B20:  00 93 38 93 54   ; ladder leading to girl
3B25:  01 6B 54 6B 75   ; fourth broken ladder near kong
3B2A:  02 EF AF 20 BB   ; 4th slanted girder
3B2F:  02 DF 9A 10 8E   ; 3rd slanted girder
3B34:  AA               ; end code

3B39:  01 53 92 53 B8   ; second broken ladder from bottom, on 3rd girder
3B3E:  00 5B 76 5B 92   ; longer ladder under the top left hammer
3B43:  00 73 B6 73 D6   ; longer ladder to left of bottom hammer
3B48:  00 83 95 83 B5   ; center longer ladder
3B4D:  00 93 38 93 54   ; ladder leading to girl
3B52:  01 DB 70 DB 98   ; third broken ladder on right side near top
3B57:  01 6B 54 6B 75   ; fourth broken ladder near kong
3B5C:  AA               ; AA code signals end of data

; table data for screen 2 conveyors

3B5D:  06 8F 90 70 90   ; central patch of XXX's
3B62:  06 8F 98 70 98   ; central patch of XXX's
3B67:  06 8F A0 70 A0   ; central patch of XXX's
3B6C:  00 63 18 63 58   ; kong's ladder (right)
3B71:  00 63 80 63 A8   ; center ladder to left of oil can fire
3B76:  00 63 D0 63 F8   ; bottom level ladder #2 of 4
3B7B:  00 53 18 53 58   ; kong's ladder (left)
3B80:  00 53 A8 53 D0   ; ladder under the hat
3B85:  00 9B 80 9B A8   ; center ladder to right of oil can fire
3B8A:  00 9B D0 9B F8   ; bottom level ladder #3 of 4
3B8F:  01 23 58 23 80   ; top broken ladder left side
3B94:  01 DB 58 DB 80   ; top broken ladder right side
3B99:  00 2B 80 2B A8   ; ladder on left platform with hammer
3B9E:  00 D3 80 D3 A8   ; ladder on right plantform with umbrella
3BA3:  00 A3 A8 A3 D0   ; ladder to right of bottom hammer
3BA8:  00 2B D0 2B F8   ; bottom level ladder #1 of 4
3BAD:  00 D3 D0 D3 F8   ; bottom level ladder #4 of 4
3BB2:  00 93 38 93 58   ; ladder leading to girl
3BB7:  02 97 38 68 38   ; girder where girl sits
3BBC:  03 EF 58 10 58   ; top conveyor girder
3BC1:  03 F7 80 88 80   ; top right conveyor next to oil can
3BC6:  03 77 80 08 80   ; top left conveyor next to oil can
3BCB:  02 A7 A8 50 A8   ; center ledge
3BD0:  02 E7 A8 B8 A8   ; right center ledge
3BD5:  02 3F A8 18 A8   ; left center ledge (has hammer)
3BDA:  03 EF D0 10 D0   ; main lower conveyor girder (has hammer)
```

```
3BDF:  02 EF F8 10 F8   ; bottom level girder
3BE4:  AA               ; end code

; table data for the elevators

3BE5:  00 63 18 63 58   ; kong's ladder (right)
3BEA:  00 63 88 63 D0   ; center ladder right
3BEF:  00 53 18 53 58   ; long's ladder (left)
3BF4:  00 53 88 53 D0   ; center ladder left
3BF9:  00 E3 68 E3 90   ; far top right ladder leading to purse
3BFE:  00 E3 B8 E3 D0   ; far bottom right ladder
3C03:  00 CB 90 CB B0   ; ladder leading to purse (lower level)
3C08:  00 B3 58 B3 78   ; ladder leading to kong's level
3C0D:  00 9B 80 9B A0   ; ladder to right of top right elevator
3C12:  00 93 38 93 58   ; ladder leading up to girl
3C17:  00 23 88 23 C0   ; long ladder on left side
3C1C:  00 1B C0 1B E8   ; bottom left ladder
3C21:  02 97 38 68 38   ; girder girl is on
3C26:  02 D7 58 10 58   ; kong's girder
3C2B:  02 EF 68 E0 68   ; girder where purse is
3C30:  02 D7 70 C8 70   ; girder to left of purse
3C35:  02 BF 78 B0 78   ; girder holding ladder that leads up to kong's level
3C3A:  02 A7 80 90 80   ; girder to right of top right elevator
3C3F:  02 67 88 48 88   ; top girder for central ladder section between elevators
3C34:  02 27 88 10 88   ; girder that holds the umbrella
3C39:  02 EF 90 C8 90   ; girder under the girder that has the purse
3C4E:  02 A7 A0 98 A0   ; bottom girder for section to right of top right elevator
3C53:  02 BF A8 B0 A8   ; small floating girder
3C58:  02 D7 B0 C8 B0   ; small girder
3C5D:  02 EF B8 E0 B8   ; small girder
3C62:  02 27 C0 10 C0   ; girder just above mario start
3C67:  02 EF D0 D8 D0   ; small girder on far right bottom
3C6C:  02 67 D0 50 D0   ; bottom girder for central ladder section between elevators
3C71:  02 CF D8 C0 D8   ; small girder
3C76:  02 B7 E0 A8 E0   ; small girder
3C7B:  02 9F E8 88 E8   ; floating girder where the right side elevator gets off
3C80:  02 27 E8 10 E8   ; girder where mario starts
3C85:  02 EF F8 10 F8   ; long bottom girder (mario dies if he gets that low)
3C8A:  AA               ; end code

; table data for the rivets

3C8B:  00 7B 80 7B A8   ; center ladder level 3
3C90:  00 7B D0 7B F8   ; bottom center ladder
3C95:  00 33 58 33 80   ; top left ladder
3C9A:  00 53 58 53 80   ; top left ladder (right side)
3C9F:  00 AB 58 AB 80   ; top right ladder (left side)
3CA4:  00 CB 58 CB 80   ; top right ladder
3CA9:  00 2B 80 2B A8   ; level 3 ladder left side
3CAE:  00 D3 80 D3 A8   ; level 3 ladder right side
3CB3:  00 23 A8 23 D0   ; level 2 ladder left side
3CB8:  00 5B A8 5B D0   ; level 2 ladder #2 of 4
3CBD:  00 A3 A8 A3 D0   ; level 2 ladder #3 of 4
3CC2:  00 DB A8 DB D0   ; level 2 ladder right side
3CC7:  00 1B D0 1B F8   ; bottom left ladder
3CCC:  00 E3 D0 E3 F8   ; bottom right ladder
3CD1:  05 B7 30 48 30   ; girder above kong
3CD6:  05 CF 58 30 58   ; girder kong stands on
3CDB:  05 D7 80 28 80   ; level 4 girder
3CE0:  05 DF A8 20 A8   ; level 3 girder
3CE5:  05 E7 D0 18 D0   ; level 2 girder
3CEA:  05 EF F8 10 F8   ; bottom level girder
3CEF:  AA               ; end code

; variable ladder definitions for screen 4 rivets

3B35:  7B 83 7B 83 80 A8   ; level 3 center ladder
3B3B:  7B 83 7B 83 D0 F8   ; bottom center ladder
3B41:  53 63 6B 73 A8 D0   ; level 2 ladder #2 of 4
3B47:  FF                  ; mirror - level 2 ladder #3 of 4
3B48:  33 43 33 43 58 80   ; top left ladder
3B4E:  FF                  ; mirror - top right ladder
3B4F:  53 5B 63 53 58 80   ; top left ladder right side
3B55:  FF                  ; mirror - top right ladder left side
3B56:  2B 3B 2B 3B 80 A8   ; level 3 ladder left side
3B5C:  FF                  ; mirror - level 3 ladder right side
3B5D:  23 33 43 23 A8 D0   ; level 2 ladder left side
3B63:  FF                  ; mirror - level 2 ladder right side
3B64:  1B 2B 3B 1B D0 F8   ; bottom left ladder
3B6A:  FF                  ; mirror - bottom right ladder
3B6B:  AA                  ; end code

; table data for screen 4 rivets

3B6C:  05 B7 30 48 30   ; girder above kong
3B71:  05 CF 58 30 58   ; girder kong stands on
3B76:  05 D7 80 28 80   ; level 4 girder
3B7B:  05 DF A8 20 A8   ; level 3 girder
3B80:  05 E7 D0 18 D0   ; level 2 girder
3B85:  05 EF F8 10 F8   ; bottom level girder
3B8A:  AA               ; end code

; variable ladder definitions for screen 2 pies

3B8B:  2B 2B 33 3B D0 F8   ; bottom ladder #1 of 4
```

```
3B91:  FF                ; mirror - bottom ladder #4 of 4
3B92:  4B 5B 6B 73 D0 F8 ; bottom ladder #2 of 4
3B98:  FF                ; mirror - bottom ladder #3 of 4
3B99:  53 63 53 23 A8 D0 ; ladder under the hat
3B9F:  FF                ; mirror - ladder right of bottom hammer
3BA0:  5B 6B 5B 6B 80 A8 ; center ladder left of oil can
3BA6:  FF                ; mirror - center ladder right of oil can
3BA7:  1B 2B 33 3B 80 A8 ; ladder left platform with hammer
3BAD:  FF                ; mirror - ladder right platform with umbrella
3BAE:  AA                ; end code

; table data for screen 2 pies

3BAF:  06 8F 90 70 90    ; central patch of XXX's
3BB4:  06 8F 98 70 98    ; central patch of XXX's
3BB9:  06 8F A0 70 A0    ; central patch of XXX's
3BBE:  00 63 18 63 58    ; kong's ladder (right)
3BC3:  00 53 18 53 58    ; kong's ladder (left)
3BC8:  01 23 58 23 80    ; top broken ladder left side
3BCD:  01 DB 58 DB 80    ; top broken ladder right side
3BD2:  00 93 38 93 58    ; ladder leading to girl
3BD7:  02 97 38 68 38    ; girder where girl sits
3BDC:  03 EF 58 10 58    ; top conveyor girder
3BE1:  03 F7 80 88 80    ; top right conveyor next to oil can
3BE6:  03 77 80 08 80    ; top left conveyor next to oil can
3BEB:  02 AF A8 50 A8    ; center ledge
3BF0:  02 E7 A8 C0 A8    ; right center ledge
3BF5:  02 3F A8 18 A8    ; left center ledge (has hammer)
3BFA:  03 EF D0 10 D0    ; main lower conveyor (has hammer)
3BFF:  02 EF F8 10 F8    ; bottom level girder
3C04:  AA                ; end code

; variable ladder definitions for screen 3 elevators

3C05:  1B 1B 1B 1B C0 E8 ; bottom left ladder
3C0B:  53 53 53 53 88 D0 ; center ladder left
3C11:  B3 AB B3 AB 58 78 ; ladder leading to kong's level
3C17:  9B 93 9B 93 80 B0 ; ladder to right of top right elevator
3C1D:  23 23 23 23 88 C0 ; long ladder at left side
3C23:  63 63 63 63 88 D0 ; center ladder right
3C29:  B3 AB B3 AB A8 E0 ; far top right ladder leading to purse
3C2F:  DB DB DB DB 90 B8 ; ladder leading to purse (lower level)
3C35:  E3 E3 E3 E3 B8 D0 ; far bottom right ladder
3C3B:  93 93 93 93 38 58 ; ladder leading up to girl
3C41:  53 53 53 53 18 58 ; kong's ladder (left)
3C47:  63 63 63 63 18 58 ; kong's ladder (right)
3C4D:  DB DB DB DB 58 70 ; additional ladder right side
3C53:  E3 E3 E3 E3 70 90 ; additional ladder right side
3C59:  AA                ; end code

; table data for screen 3 elevators

3C5A:  02 97 38 68 38    ; girder girl is on
3C5F:  02 B7 58 10 58    ; kong's girder
3C64:  02 EF 70 D8 70    ; girder where purse is
3C69:  02 CF 70 C0 70    ; girder to left of purse
3C6E:  02 B0 78 A7 78    ; girder holding ladder that leads up to kong's level
3C73:  02 9F 80 90 80    ; girder to right of top right elevator
3C78:  02 67 88 48 88    ; top girder for central ladder section
3C7D:  02 27 88 10 88    ; girder that holds the umbrella
3C82:  02 EF 90 D8 90    ; girder under the girder that has the purse
3C87:  02 9F B0 90 B0    ; bottom girder for section to right of top right elevator
3C8C:  02 B7 A8 A8 A8    ; small floating girder
3C91:  02 CF B0 C0 B0    ; small girder
3C96:  02 EF B8 D8 B8    ; small girder
3C9B:  02 27 C0 10 C0    ; girder just above mario start
3CA0:  02 EF D0 D8 D0    ; small girder on far right bottom
3CA5:  02 67 D0 50 D0    ; bottom girder for central ladder section
3CAA:  02 CF D8 C0 D8    ; small girder
3CAF:  02 B7 E0 A8 E0    ; small girder
3CB4:  02 9F E8 88 E8    ; floating girder where the right side elevator gets off
3CB9:  02 27 E8 10 E8    ; girder where mario starts
3CBE:  02 EF F8 10 F8    ; long bottom girder
3CC3:  02 EF 58 C8 58    ; additional girder top right
3CC8:  AA                ; end code

; code to display the sub-level in the high score list

3CC9  3610      LD      (HL),#10        ; draw a space at this position
3CCB  23        INC     HL              ; next position in high score row
3CCC  23        INC     HL              ; next position in high score row
3CCD  3A2962    LD      A,(#6229)       ; load A with level #
3CD0  010AFF    LD      BC,#FF0A        ; B := #FF, C := #0A (10 decimal)
3CD3  04        INC     B               ; increment B
3CD4  91        SUB     C               ; A < 10?
3CD5  30FC      JR      NC,#3CD3        ; no, loop again
3CD7  81        ADD     A,C             ; add 10 back to A to get a number from 0 to 9
3CD8  77        LD      (HL),A          ; draw the one position of the level # to high score row
3CD9  2B        DEC     HL              ; previous position in high score row
3CDA  78        LD      A,B             ; load A with B
3CDB  77        LD      (HL),A          ; draw the tens position of the level # to high score row
3CDC  23        INC     HL              ; next position in high score row
3CDD  23        INC     HL              ; next position in high score row
3CDE  362C      LD      (HL),#2C        ; draw hyphen to high score row
3CE0  23        INC     HL              ; next position in high score row
```

```
3CE1  3A2E62    LD      A,(#622E)       ; load A with board #
3CE4  3C        INC     A               ; increment A
3CE5  C3833F    JP      #3F83           ; jump to additional code (continue at #3F83)

3CE8  0000000000000000                  ; no operations - free space

3CF0:  10 82 85 8B 10 85 80 8B 10 87 85 8B 81 80 80 8B  .25m.50m.75m100m
3D00:  81 82 85 8B 81 85 80 8B                          125m150m

; used to draw the game logo in attract mode
; data called from #07F7
; data grouped in 3's
; first byte is a loop counter - how many things to draw, going down
; 2nd and 3rd bytes are coordinates to start

3D08:  05 88 77 01 68 77 01 6C 77 03 49 77              ; D
3D14:  05 08 77 01 E8 76 01 EC 76 05 C8 76              ; O
3D20:  05 88 76 02 69 76 02 4A 76 05 28 76              ; N
3D2C:  05 E8 75 01 CA 75 03 A9 75 01 88 75 01 8C 75     ; K
3D3B:  05 48 75 01 28 75 01 2A 75                       ; E (part 1)
3D44:  01 2C 75 01 08 75 01 0A 75 01 0C 75              ; E (part 2)
3D50:  03 C8 74 03 AA 74 03 88 74                       ; Y
3D59:  05 2F 77 05 0F 77 02 F0 76 02 CF 76 02 D2 76     ; K
3D68:  05 8F 76 05 6F 76 01 4F 76 01 53 76 05 2F 76     ; O
3D77:  05 EF 75 02 D0 75 02 B1 75 05 8F 75              ; N
3D83:  03 50 75 05 2F 75 01 0F 75 01 13 75              ; G (part 1)
3D8F:  01 EF 74 01 F1 74 01 F3 74 02 D1 74              ; G (part 2)
3D59:  05 30 77 05 10 77 02 F1 76 02 D0 76 02 D3 76     ; K - 1 position down
3D68:  05 90 76 05 70 76 01 50 76 01 54 76 05 30 76     ; O - 1 position down
3D77:  05 F0 75 02 D1 75 02 B2 75 05 90 75              ; N - 1 position down
3D83:  03 51 75 05 30 75 01 10 75 01 14 75              ; G (part 1) - 1 position down
3D8F:  01 F0 74 01 F2 74 01 F4 74 02 D2 74              ; G (part 2) - 1 position down
3D9B:  00                                               ; end code

; table code reference from #0F6F
; values are copied into #6280 through #6280 + #40

3D9C:                                       00 00 23 68
3DA0:  01 11 00 00 00 10 DB 68 01 40 00 00 08 01 01 01
3DB0:  01 01 01 01 01 01 00 00 00 00 00 00 80 01 C0 FF
3DC0:  01 FF FF 34 C3 39 00 67 80 69 1A 01 00 00 00 00
3DD0:  00 00 00 00 04 00 10 00 00 00 00 00

; data used for the barrel pile next to kong
; called from #0FD7

3DDC  1E 18 0B 4B       ; first barrel
3DE0  14 18 0B 4B       ; second barrel
3DE4  1E 18 0B 3B       ; third barrel
3DE8  14 18 0B 3B       ; fourth barrel

; the following is table data that gets copied to #6407 - location and other data of the fires?
; 05 is a loop varialbe
; 1C loops value corresponds to total length of table

3DEC  3D 01 03 02

; table data that also gets called from #1138
; DE is #6407 - Fire # 1 y value
; B is 05 and C is 1C

3DF0:  4D 01 04 01

3DF4:  27 70 01 E0 00 00        ; initial data for fires on girders ?
3DFA:  7F 40 01 78 02 00        ; initial data for conveyors to release a fire ?

; table data called from #0FF5.  4 bytes

3E00  27 49 0C F0               ; oil can for girders
3E04  7F 49 0C 88               ; oil can for conveyors ?

; another table called and copied into #6687-668A an #6697-#669A - has to do with the hammers
; B counter is #02 and C is #0C
; called from #122E
; 3E0C is called also from #1000

3E08:  1E 07            ; 1E is the hammer sprite value.  07 is hammer color
3E0A:  03 09            ; ???
3E0C:  24 64            ; position of top hammer for girders.  24 is X, 64 is Y
3E0E:  BB C0            ; bottom hammer for girders at BB, C0

3E10:  23 8D 7B B4      ; for conveyors

3E14:  1B 8C 7C 64      ; for rivets
3E18:  4B 0E 04 02      ; ???

; 2 ladder sprites for conveyors
; 46 = ladder

3E1C:  23 46 03 68              ; ladder at 23, 68
3E20:  DB 46 03 68              ; ladder at DB, 68

; the 6 conveyor pulleys

3E24:  17 50 00 5C              ; 50 = edge of conveyor pulley
```

```
3E28:  E7 D0 00 5C              ; D0 = edge of conveyor pulley inverted
3E2C:  8C 50 00 84
3E30:  73 D0 00 84
3E34:  17 50 00 D4
3E38:  E7 D0 00 D4

; bonus items on conveyors

3E3C  53 73 0A A0               ; position of hat on pies is 53,A0
3E40  8B 74 0A F0               ; position of purse on pies is 8B,F0
3E44  DB 75 0A A0               ; umbrella on the pies is at DB,A0

; bonus items for elevators

3E48  5B 73 0A C8               ;  hat at 5B,C8
3E4C  E3 74 0A 60               ;  purse at E3,60
3E4C  E3 74 0A 68               ;  purse repositioned at E3,68
3E50  1B 75 0A 80               ;  umbrella on elevator is 80,1B

; bonus items for rivets

3E54  DB 73 0A C8               ; hat on rivets at DB,C8
3E58  93 74 0A F0               ; purse on rivets at 93,F0
3E5C  33 75 0A 50               ; umbrella on rivets at 33,50

; used in elevators - called from #10CC

3E60:  44 03 08 04

; used in elevators, called from #11EC
; used for elevator sprites

3E64:  37 F4
3E66:  37 C0
3E68:  37 8C                    ; elevators on left all have X value of 37

3E6A:  77 70
3E6C:  77 A4
3E6E:  77 D8                    ; elevators on right all have X value of 77

; award points for jumping a barrels and items
; arrive from #1DD7
; A is preloaded with 1,3, or 7
; patch ?

3E70  110100    LD      DE,#0001        ; 100 points
3E73  067B      LD      B,#7B           ; sprite for 100
3E75  1F        RRA                     ; is the score set for 100 ?
3E76  D2281E    JP      NC,#1E28        ; yes, award points

3E79  1E03      LD      E,#03           ; else set 300 points
3E7B  067D      LD      B,#7D           ; sprite for 300
3E7D  1F        RRA                     ; is the score set for 300 ?
3E7E  D2281E    JP      NC,#1E28        ; yes, award points

3E81  1E05      LD      E,#05           ; else set 500 points [bug, should be 800]
3E83  067F      LD      B,#7F           ; sprite for 800
3E85  C3281E    JP      #1E28           ; award points


; called from #286B
; a patch ?

3E88  3A2762    LD      A,(#6227)       ; load A with screen number
3E8B  E5        PUSH    HL              ; save HL
3E8C  EF        RST     #28             ; jump to new location based on screen number

; data for above:

3E8D  00 00                             ; unused
3E8F  99 3E                             ; #3E99 - girders
3E91  B0 28                             ; #28B0 - pie
3E93  E0 28                             ; #28E0 - elevator
3E95  01 29                             ; #2901 - rivets
3E97  00 00                             ; unused

; checks for jumps over items on girders

3E99  E1        POP     HL              ; restore HL
3E9A  AF        XOR     A               ; A := 0
3E9B  326060    LD      (NumObstaclesJumped),A       ; clear counter for barrels jumped
3E9E  060A      LD      B,#0A           ; For B = 1 to #A barrels
3EA0  112000    LD      DE,#0020        ; load DE with offset
3EA3  DD210067  LD      IX,#6700        ; load IX with start of barrel info table
3EA7  CDC33E    CALL    #3EC3           ; call sub below.  check for barrels under jump

3EAA  0605      LD      B,#05           ; for B = 1 to 5 fires
3EAC  DD210064  LD      IX,#6400        ; start of fires table
3EB0  CDC33E    CALL    #3EC3           ; check for fires being jumped

3EB3  3A6060    LD      A,(NumObstaclesJumped)       ; load A with counter for items jumped
3EB6  A7        AND     A               ; nothing jumped ?
3EB7  C8        RET     Z               ; yes, return

3EB8  FE01      CP      #01             ; was 1 item jumped?
```

```
3EBA  C8        RET     Z               ; yes, return; 1 is the code for 100 pts

3EBB  FE03      CP      #03             ; were less than 3 items jumped ?
3EBD  3E03      LD      A,#03           ; A := 3  = code for 2 items, 300 pts score
3EBF  D8        RET     C               ; yes, return

3EC0  3E07      LD      A,#07           ; else A := 7 = code for 3+ items, awards 800 points
3EC2  C9        RET                     ; return

; subroutine called from #3EA7 above
; checks for mario jumping over barrels or fires
; H is preloaded with either 5 or #13 (19 decimal) for the area under mario ?
; C is preloaded with mario's Y position + #C (12 decimal)
; IX preloaded with start of array for fires or barrels, EG #6700 or #6400
; L is preloaded with height window value ?
; DE is preloaded with offset to add for next sprite

3EC3  DDCB0046  BIT     0,(IX+#00)      ; is this barrel/fire active?
3EC7  CAFA3E    JP      Z,#3EFA         ; no, jump ahead to try next one

3ECA  79        LD      A,C             ; load A with mario's adjusted Y position
3ECB  DD9605    SUB     (IX+#05)        ; subtract the fire/barrel Y position.  did the result go negative?
3ECE  D2D33E    JP      NC,#3ED3        ; no, skip next step

3ED1  ED44      NEG                     ; Negate A (A := 0 - A)

3ED3  3C        INC     A               ; increment A
3ED4  95        SUB     L               ; subtract L (height window?)  Is there a carry ?
3ED5  DADE3E    JP      C,#3EDE         ; yes, skip next two steps

3ED8  DD960A    SUB     (IX+#0A)        ; else subtract the items' height???
3EDB  D2FA3E    JP      NC,#3EFA        ; if out of range, jump ahead to try next one

; we are within the Y range, test X range next

3EDE  FD7E03    LD      A,(IY+#03)      ; load A with mario's X position
3EE1  DD9603    SUB     (IX+#03)        ; subtract the item's X position
3EE4  D2E93E    JP      NC,#3EE9        ; if no carry, skip next step

3EE7  ED44      NEG                     ; negate A

3EE9  94        SUB     H               ; subtract the horizontal window (5 or 19 pixels)
3EEA  DAF33E    JP      C,#3EF3         ; if out of range, skip next 2 steps

3EED  DD9609    SUB     (IX+#09)        ; subtract the item's width???
3EF0  D2FA3E    JP      NC,#3EFA        ; if out of range, skip ahead to try next one

; item was jumped

3EF3  3A6060    LD      A,(NumObstaclesJumped)      ; load A with counter of how many barrels/fires jumped
3EF6  3C        INC     A               ; increase it
3EF7  326060    LD      (NumObstaclesJumped),A      ; store

3EFA  DD19      ADD     IX,DE           ; add offset for next barrel or fire
3EFC  10C5      DJNZ    #3EC3           ; Next B

3EFE  C9        RET                     ; return

; ... overwrites the message from game creators...

3EFF:  00
3F00:  5C 76 49 4A 01 09 08 01 3F 7D 77 1E 19 1E 24 15  .(C)1981...NINTE
3F10:  1E 14 1F 10 1F 16 10 11 1D 15 22 19 13 11 10 19  NDO.OF.AMERICA.I
3F20:  1E 13 2B 3F                                      NC..
3F00:  5C 77 49 4A 10 01 09 08 01 2C 02 00 02 02 10 1E  .(C).1981-2022 N
3F10:  19 1E 24 15 1E 14 1F 3F 8E 76 10 22 1E 14 1D 2A  INTENDO....RNDMZ
3F20:  22 10 3F                                         R..

; called from #081C : patch to draw the TM logo on attract screen

3F24  21AF74    LD      HL,$74AF        ; load HL with screen VRAM address
3F27  11E0FF    LD      DE,$FFE0        ; load offset
3F2A  369F      LD      (HL),$9F        ; draw first part of TM logo to screen
3F2C  19        ADD     HL,DE           ; next screen location
3F2D  369E      LD      (HL),$9E        ; draw second part of TM logo to screen
3F2F  C9        RET                     ; return

3F30:  50 52 4F 47 52 41 4D 2C 57 45 20 57 4F 55 4C 44  PROGRAM,WE WOULD
3F40:  20 54 45 41 43 48 20 59 4F 55 2E 2A 2A 2A 2A 2A   TEACH YOU.*****
3F50:  54 45 4C 2E 54 4F 4B 59 4F 2D 4A 41 50 41 4E 20  TEL.TOKYO-JAPAN
3F60:  30 34 34 28 32 34 34 29 32 31 35 31 20 20 20 20  044(244)2151
3F70:  45 58 54 45 4E 54 49 4F 4E 20 33 30 34 20 20 20  EXTENTION 304
3F80:  53 59 53 54 45 4D 20 44 45 53 49 47 4E 20 20 20  SYSTEM DESIGN
3F90:  49 4B 45 47 41 4D 49 20 43 4F 2E 20 4C 49 4D 2E  IKEGAMI CO. LIM.

; code to display the romhack version

3F23  11 01 03   LD      DE,#0301        ; load task data for tekst "V1.00" - task 1
3F26  CD 9F 30   CALL    #309F           ; insert task to draw text
3F29  C3 1F 08   JP      #081F           ; jump back

3F2C  00000000                          ; no operations - free space

; code to set game mode 2 to 6 so game starts with title screen
```

```
3F30  3E06      LD      A,#06           ; load A with #06
3F32  320A60    LD      (#600A),A       ; store into game mode 2
3F35  C3F101    JP      #01F1           ; jump back

; code to fix for short broken ladders that appear as normal because top and bottom part overlap (jump from #0E3B)
; this part of the fix draws the bottom of the ladder with a shorter ladder graphic character (ladder < 32 pixels)

3F38  1A        LD      A,(DE)          ; load A with y-value bottom of the ladder
3F39  47        LD      B,A             ; store y-value bottom of ladder to B
3F3A  D5        PUSH    DE              ; save DE to stack for later
3F3B  1B        DEC     DE              ; decrement DE, DE points to x-value bottom of the ladder
3F3C  1B        DEC     DE              ; decrement DE, DE points to y-value top of the ladder
3F3D  1A        LD      A,(DE)          ; load A with y-value top of the ladder
3F3E  D1        POP     DE              ; restore DE from the stack
3F3F  4F        LD      C,A             ; store y-value top of the ladder to C
3F40  78        LD      A,B             ; load A with y-value bottom of the ladder from B
3F41  91        SUB     C               ; subtract y-value top of the ladder from y-value bottom of the ladder
3F42  FE20      CP      #20             ; is this a short broken ladder (shorter than 32 pixels)
3F44  DA3F0E    JP      C,#0E3F         ; yes, jump back, do not draw bottom of the broken ladder
3F47  2D        DEC     L               ; decrease HL (previous screen position, one position up)
3F48  36C0      LD      (HL),#C0        ; draw character #C0 to screen position (shorter ladder bottom piece)
3F4A  2C        INC     L               ; increase HL (original screen position)
3F4B  C33F0E    JP      #0E3F           ; jump back

; code to fix for short broken ladders that appear as normal because top and bottom part overlap (jump from #0E2F)
; this part of the fix skips drawing the bottom part of the ladder (ladder < 22 pixels)

3F4E  F5        PUSH    AF              ; save AF to the stack for later
3F4F  2AAD63    LD      HL,(#63AD)      ; initialize HL (original code from #0E2F)
3F52  3AB363    LD      A,(#63B3)       ; load A with original data item (original code from #0E33)
3F55  FE01      CP      #01             ; == #01? (is this a broken ladder?) (original code from #0E36)
3F57  C26D3F    JP      NZ,#3F6D        ; no, jump forward
3F5A  1A        LD      A,(DE)          ; load A with y-value bottom of the ladder
3F5B  47        LD      B,A             ; store y-value bottom ladder to B
3F5C  D5        PUSH    DE              ; save DE to stack for later
3F5D  1B        DEC     DE              ; decrement DE, DE points to x-value bottom of the ladder
3F5E  1B        DEC     DE              ; decrement DE, DE points to y-value top of the ladder
3F5F  1A        LD      A,(DE)          ; load A with y-value top of the ladder
3F60  D1        POP     DE              ; restore D from the stack
3F61  4F        LD      C,A             ; store y-value top of the ladder to C
3F62  78        LD      A,B             ; load A with y-value bottom of the ladder from B
3F63  91        SUB     C               ; subtract y-value top of the ladder from y-value bottom of the ladder
3F64  FE16      CP      #16             ; is this a short broken ladder (shorter than 22 pixels)?
3F66  D26D3F    JP      NC,#3F6D        ; no, jump forward
3F69  F1        POP     AF              ; restore AF from the stack
3F6A  C3330E    JP      #0E33           ; jump back, do not draw bottom of the broken ladder
3F6D  F1        POP     AF              ; restore AF from the stack
3F6E  C3320E    JP      #0E32           ; jump back, draw bottom of the broken ladder

; code to initialize a cycling pointer into the code (#0100-#3FFF)

3F71  213262    LD      HL,#6232        ; load HL with the memory that stores the MSB of the pointer into the code
3F74  3A1860    LD      A,(#6018)       ; load A with a random value based on RngTimer1
3F77  E63F      AND     #3F             ; A is a random number between #00 and #3F
3F79  77        LD      (HL),A          ; store the MSB of the pointer into the code
3F7A  23        INC     HL              ; load HL with the memory that stores the LSB of the pointer into the code
3F7B  3A1A60    LD      A,(#601A)       ; load A with a random value based on FrameCounter
3F7E  77        LD      (HL),A          ; store the MSB of the pointer into the code
3F7F  210960    LD      HL,#6009        ; load HL with timer (original code from #0A67)
3F82  C9        RET                     ; return

; continued code to display the sub-level in the high score list (jump from #3CE5)

3F83  77        LD      (HL),A          ; store board # into memory at (HL)
3F84  23        INC     HL              ; next position in high score row
3F85  0608      LD      B,#08           ; for B = #01 to #08
3F87  3610      LD      (HL),#10        ; draw a space at this position
3F89  23        INC     HL              ; next position in high score row
3F8A  10FB      DJNZ    #3F87           ; next B
3F8C  C3F213    JP      #13F2           ; jump back

3F8F  00000000000000000000000000000000  ; no operations - free space
3F9F  00                                ; no operations - free space

; jump here from #0CD1
; a patch ?

3FA0  CDA63F    CALL    #3FA6           ; call sub below
3FA3  C35F0D    JP      #0D5F           ; return to program [this was original line wiped by patch ?]

; called from #3FA0 above

3FA6  3E02      LD      A,#02           ; A := 2
3FA8  F7        RST     #30             ; check to see if the level is pie factory.  If not, RET to #3FA3 [then jump to
#0D5F]

3FA9  0602      LD      B,#02           ; for B = 1 to 2
3FAB  216C77    LD      HL,#776C        ; load HL with video RAM address for top rectractable ladder

3FAE  3610      LD      (HL),#10        ; clear the top of the ladder
3FB0  23        INC     HL
3FB1  23        INC     HL              ; next address
3FB2  36C0      LD      (HL),#C0        ; draw a ladder 2 rows down
3FB4  218C74    LD      HL,#748C        ; set HL for next loop - does the other side of the screen ; [sloppy?  this
instruction not needed on 2nd loop]
```

```
3FB7  10F5      DJNZ    #3FAE           ; Next B

3FB9  C9        RET                     ; return [to #3FA3, then jump to #0D5F]

3FBA:  00 00 00 00 00 00                ; unused

; called from #2285
; [seems like a patch ? - resets mario sprite when ladder descends]

3FC0  214D69    LD      HL,#694D        ; load HL with mario sprite value
3FC3  3603      LD      (HL),#03        ; store 3 = mario on ladder with left hand up
3FC5  2C        INC     L
3FC6  2C        INC     L               ; HL := #694F = mario sprite Y value
3FC7  C9        RET                     ; return

3FC8:  00 00 41 7F 7F 41 00 00
3FD0:  00 7F 7F 18 3C 76 63 41
3FD8:  00 00 7F 7F 49 49 49 41
3FE0:  00 1C 3E 63 41 49 79 79
3FE8:  00 7C 7E 13 11 13 7E 7C
3FF0:  00 7F 7F 0E 1C 0E 7F 7F
3FF8:  00 00 41 7F 7F 41 00 00
3FC8:  00 00 00 00 00 00 00 00          ; no operation - free space
3FD0:  00 00 00 00 00 00 00 00          ; no operation - free space
3FD8:  00 00 00 00 00 00 00 00          ; no operation - free space
3FE0:  00 00 00 00 00 00 00 00          ; no operation - free space
3FE8:  00 00 00 00 00 00 00 00          ; no operation - free space
3FF0:  00 00 00 00 00 00 00 00          ; no operation - free space
3FF8:  00 00 00 00 00 00 00 00          ; no operation - free space
```