Emergency – invasion imminent

This month your favourite aunt brings you the second part of Auntie John and the invaders from outer space – so get blasting.

Greetings! If you missed last month's exciting machine-code extravaganza, then I'm afraid that what follows will be almost totally irrelevant. The assembler source code that follows is the second and final instalment of our DIY (Destroy it Yourself) Space Invaders game. If you don't have last month's ACU then rush out to that newsagent around the corner who always has lots of back issues and buy it immediately. If your newsagent is anything like mine, it is staffed by two old ladies who still refer to money as 'new pence' and who think a computer is someone who gets the train to work every morning. Technology seems to have passed by this particular part of the world. For goodness sake, you can even buy a copy of 'Your ZX81'!

Alternatively, have a word with the Back Issue Department; she has nothing better to do while she watches *Neighbours* during the lunchbreak.

The assembler listing that is taking up all that space over there is the second half of our Space Invaders program. If you are typing it all in, please bear in mind that it must be added immediately after the first section of code: It cannot be assembled separately, because the labels cross-reference between the listings. Load up the code you spent all last month entering and type in all this stuff directly after it. What fun! If you don't think that there will be enough space in your computer's memory to hold all the source listings as well as the assembled code, check out your assembler instructions for written object code to disc or tape. With the MAXAM assembler, the directive in question is 'WRITE <filename>'.

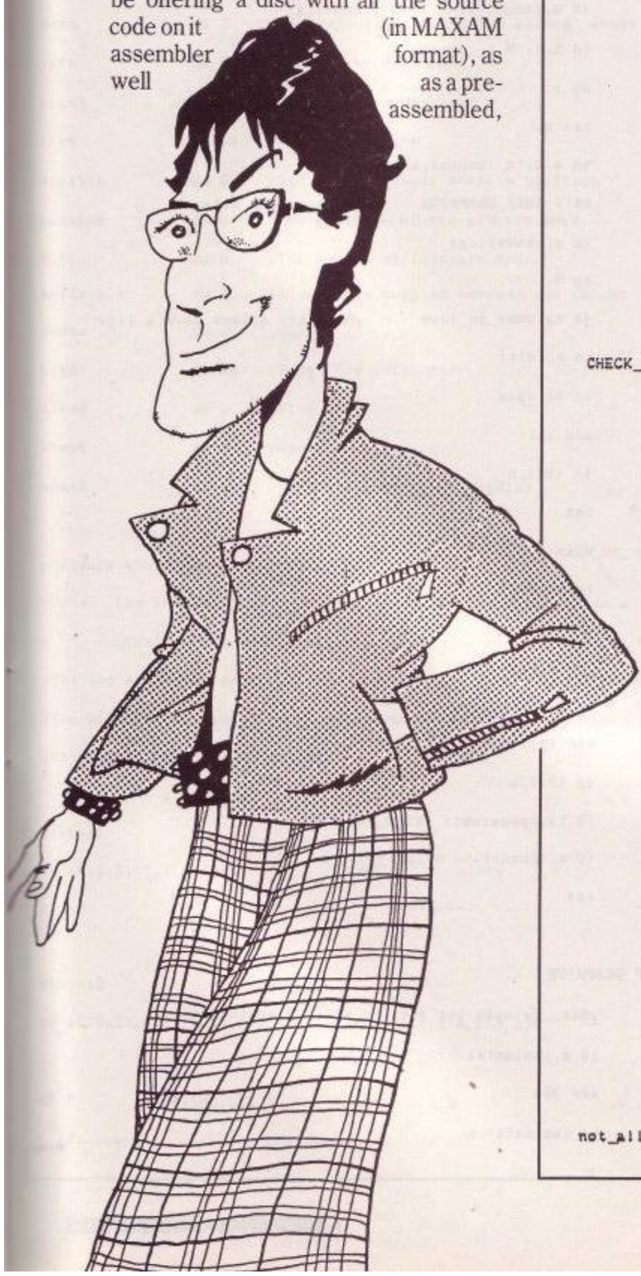
This month's assembler is the code needed to control the Invaders, move the bombs, and provide the various sub-routines needed, such as printing text and creating random numbers. (The random number sub-routine is a very useful one which you may want to steal, copy, borrow, rip-off and use it in your own programs). All the game

```
  ; --------------------------  INVADERS  ---------------------------
  ; - - - - - - - - - - - - -   Part Two   - - - - - - - - - - - - -


  INVADERS                                              ;Control the Aliens.
        push iy:pop ix:call PRINT_INVADERS              ;Business with IY/IX
        call MOVE_INVADERS                              ;is to animate aliens
        call CONTROL_BOMB                               ;see text.
        ret



PRINT_INVADERS
       ld a,0:ld (moveflag),a
       ld b,4                                          ;There are 4 rows.
       ld de,invadersdata                              ;DE -> life/death array.
       ld a,(ypos)                                     ;Aliens' height
       ld h,a
lpb    push bc
       ld b,7                                          ;There are 7 coloumns.
       ld a,(xpos)                                     ;How far across the
       ld l,a                                          ;screen they are.
lpc    push bc
       ld a,(de):cp 0:jr z,delay       ;If this invader is dead then slow down.
       push de:push hl
       call PRINT_ALIEN                                       ;Print a single Alien
       pop hl:pop de:push de:push hl
       call CHECK_INVADER                              ;Is it being shot?
       pop hl:pop de
         ld (de),a
  dead   inc de
         ld a,6:add l:ld l,a
         pop bc
         djnz lpc
         ld a,3:add h:ld h,a
         pop bc
         djnz lpb                                       ;Continue until all
         ret                                            ;Aliens looked at.


 delay   ld b,70                                        ;This delay is to
 lp3     djnz lp3                                       ;reduce the speed of the
```

variables are given their definitions in this section and, worst of all, the graphics data is defined. Graphics data is the bane of every programmer's life: it takes forever to define it all. I created it all with the Advanced Art Studio, and then wrote a program to make machine-code data statements out of the drawings. You'll just have to type in all the numbers I'm afraid.

The machine code itself does nothing startingly complicated. To animate the Invaders – in real terms swapping between two sets of graphics data – the IY and IX registers are used. The routine SWAPGRAPHICS decides the data to be printed, using a variable which toggles between the two values of zero and 255.

And so all there is left to say is get typing! For the faint hearted ACU will be offering a disc with all the source code on it (in MAXAM assembler format), as well as a pre-assembled,



```
        jp dead                                 ;game as more and more
                                                ;Aliens are killed.

PRINT_ALIEN                                     ;Print a single Alien
        push ix                                 ;at co-cords stored in
        push hl                                 ;H and L registers and
        call PRINT_CHAR:push de:pop ix          ;graphics pointed to in
        pop hl:push hl                          ;the IX register.
        inc l:inc l
        call PRINT_CHAR:push de:pop ix          ;Each alien is made up
        pop hl:push hl                          ;of several characters
        ld a,4:add l:ld l,a                     ;which must all be
        call PRINT_CHAR:push de:pop ix          ;printer one after
        pop hl:push hl:inc h                    ;the other.
        call PRINT_CHAR:push de:pop ix
        pop hl:push hl:inc h                    ;The graphics data
        inc l:inc l                             ;which is held in IX
        call PRINT_CHAR:push de:pop ix          ;is updated during the
        pop hl:push hl:inc h                    ;call to PRINT_CHAR
        ld a,4:add l:ld l,a
        call PRINT_CHAR
        pop hl
        pop ix
        ret

CHECK_INVADER
        ld a,(missile)
        cp 0
        jp z,not_hit                       ;Can't be shot if no missile is fired.
        sub 1
        cp 6
        jp nc,not_hit                                   ;Missile missed alien.
        ld a,(missile+1)
        sub h
        cp 2
        jp nc,not_hit                                   ;Missile missed alien.
        ;Bang! The missile has hit the Alien.
        push hl:ld ix,space:call PRINT_MISSILE:pop hl        ;Erase missile
        ld ix,banggr:call Print_Alien                        ;Draw explosion.
        call wait_frame:call wait_frame:call wait_frame      ;pause
        push hl
        ld a,0:ld (missile),a                                ;Reset missile
        ld a,(hits):inc a:ld (hits),a                  ;Increase hit counter
        cp 28:jr nz,not_all_dead                       ;Are all aliens dead?
        ld a,4:ld (flag),a
not_all_dead
```

```
        ld hl,(score):ld bc,5:add hl,bc:ld (score),hl   ;Increase score
        call Print_score                                ;and print it.
        ld hl,(score):ex de,hl:ld hl,(high)             ;Is it greater than
        scf:ccf:sbc hl,de                               ;the high score?
        jr nc,less_than
        ld hl,(score):ld (high).hl                      ;yes - change high
        call Print_high                                 ;and print it.
less_than
        pop hl
        ld ix,space:call Print_Alien:ld ix,aliengr2     ;Erase explosion
        ld a,0
        ret
not hit call DROP BOMB
        ld a,l:cp 0;call z,movedown                     ;Check for sides of
        ld a,l:cp 70:call z,movedown                    ;screen, and if aliens
        ld a,h:cp 22:jr nz,not_landed                   ;have reached bottom.
        ld a,2:ld (flag),a
not_landed
        ld a,1
        ret
 movedown
         ld a.1:ld (moveflag),a:ret

 DROP BOMB
        ;Given DE from Invadersdata decide (or not) to drop bomb
        ld a,(bomb):cp 0:ret nz
        push hl:push de
        ld hl,7
        add hl,de
        ld a,(hl)
        cp 0
        jr nz,no_bomb
        call random
        ld a,l:and 127
        cp 48
        jr nz,no_bomb
        pop de:pop hl
        ld a,h:inc a:inc a:ld (bomb+1),a
        ld a,l:inc a:inc a:ld (bomb).a
        ret
no_bomb pop de:pop hl:ret
CONTROL_BOMB
        ;Print invaders bomb and move it down;the screen.
        ld a,(bomb):cp 0:ret z
        ld hl,(bomb)
        ld ix,space:call PRINT_CHAR
        ld a,(bomb+1):inc a:ld (bomb+1),a
```

```
        cp 25:jr z,stopbomb
        ld hl,(bomb)
        ld ix,bombgr:call PRINT_CHAR
        ret
stopbomb
        ;Check to see if it has hit anything.
        ld a,(basepos):ld b,a
        ld a,(bomb)
        sub b
        cp 4
        jr nc,not_hit_base
        ld a,3:ld (flag),a
not_hit_base
        ld a,0:ld (bomb),a
MOVE_INVADERS
        ;Err.. move the invaders!
        ld a.(count):inc a:ld (count),a
        ld b,a:ld a,(speed)
        cp b
        ret nz
        ld a,0:ld (count),a
        call SWAP_GRAPHICS        ;Animate the aliens.
        ld a,(moveflag)
        cp 0
        jr nz,come_on_down        ;Move all aliens down a line.
        ld a.(dir)
        ld hl,xpos
        add (hl)
        ld (hl),a
        ret
come_on_down
        ld a,(dir)
        xor '254
        ld (dir),a
        ld hl,xpos
        add (hl)
        ld (hl),a
        ld ix.space:call PRINT_INVADERS
        ld a,(ypos):inc a:ld (ypos),a
        ret

SWAP_GRAPHICS
        ;Swap between two sets of alien graphics data.
        ld a.(animate)
        xor 255
        ld (animate),a
        ld iy,aliengr1
        cp 0
        ret z
        ld iy,aliengr2
        ret
;-----------------------*** Variables ***----------------------

flag            db 0    ;The Control flag
speed           db 0    ;How often the baddies move
count           db 0    ;A counter used with 'speed'
invadersdata    ds 26   ;Area where Invaders are stored
db 0,0,0,0,0,0,0        ;Seven zeros used by Drop-bomb routine, honest!
moveflag        db 0  ;A flag controlling Invaders movement down the screen
dir             db 0    ;The direction they move in
xpos            db 0    ;Their actual x co-ord
ypos            db 0    ;The highest Invader position
bomb            dw 0    ;The co-ordinates for the aliens' bombs
left            db 8    ;The key to move left
right           db 1    ;The key to move right
fire            db 23   ;The key to fire
missile         dw 0    ;Stores the player missile position
basepos         db 0    ;The position of the players base
hits            db 0    ;The number of Invaders shot
animate         db 0    ;A variable toggled between two values
score           dw 0    ;Guess!
high            dw 20   ;Tough one this, too!
lives           db 0    ;Give up.
sheet           db 0    ;Hmm.
space           ds 96.0 ;Blanks for erasing graphics

;Message and Space-shield strings.
;Note   The number 31 is a control character which acts like a
;       LOCATE statement. Similarly, 15 and 14 control PEN and PAPER.
;Get the spaces correct by comparing with the dashes in the comments line.
;The strings contain not only words (such as SCORE:) but block graphics
;to define the player's shields.

string1
db 15,1,31,1,1,"SCORE:             SHEET:       HIGH:      s"
;                 ------------    --------   -------

string2
db 15,3,14,0,31,6,20,214,143.215,"        ",214,143,215
;                                  --------
db "        ",214.143,215,"        ",214,143,215
; --------               --------
```

```
db 31,6,21,143,143,143,"        ",143,143,143
;                        --------
db "        ",143,143,143,"        ".143.143.143
; `--------                --------
db 31,6.22,212,32,213,"        ",212,32.213
;                       --------
db "        ",212,032,213,"        ",212,032,213.15,1,"s"
 --------                --------


string3
db 31,14,10,"GAME OVERs"


string4
db 31,14,10,"WELL DONEs"


string5
;String to erase the little bases drawn up by Print_lives routine.
db 31,13,1,32,32,32,31.13,1,"s"


string6
;Title screen
db 15,1,31,4,5,"Hyperventilating Space Invaders"
db 15,2,31,19,7, "by"
db 15,1,31,15,9,"Auntie John"
db 15,3,31,11.12."Press Space to Plays"

;---------------**** General Purpose SubRoutines Library ****--------

PRINT_CHAR
;This is a very fast Character-sized Object print routine.
;ix = address of a 16 byte table of graphics
;h = Y co-ordinate (0-24)
;l = X co-ordinate (0-78)
push ix:ld e,l:ld a,h:sla a:ld hl,chartable:ld b,0:ld c,a:add hl.bc
 ld a,(hl):ld c,a:inc hl:ld a,(hl):or &c0:ld b,a
 ld a,e:ld l,a:ld h,0:add hl,bc:pop de:ld bc,&7ff
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl).a:inc de:addhl,bc
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:addhl,bc
ld a,(de):ld (hl).a:inc hl:inc de:ld a,(de):ld (hl).a:inc de:addhl,bc
ld a,(de):ld (hl).a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:addhl,bc
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:addhl,bc
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:addhl,bc
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:addhl,bc
ld a,(de):ld (hl),a:inc hl:inc de:ld a,(de):ld (hl),a:inc de:ret
chartable
;data needed for above subroutine.
```

```
dw &C000,&C050,&C0A0,&C0F0,&C140,&C190,&C1E0,&C230,&C280

dw &C2D0,&C320,&C370,&C3C0,&C410,&C460,&C4B0,&C500,&C550

dw &C5A0,&C5F0,&C640,&C690,&C6E0,&C730,&C780

PRINT_STRING

;Print an ASCII string ending with a (non-printed) Dollar

ld a,(hl)

cp "$"

ret z

call txt_output

inc hl

jr PRINT_STRING


PRINTHL

;Print the contents of the HL register pair as a five-digit

;decimal number. You should know this code inside out by now!

ld de,10000:call pri:ld de,1000:call pri:ld de,100:call pri

ld de,10:call pri:ld de,1:pri:ld a,255:pr2:inc a:scf:ccf:sbc hl,de

jp nc,pr2:add hl,de:add 48:jp &bb5a


RANDOM

;Return a psuedo-random number in the HL pair

;Treat it as a magic formula. I do.

push af:push bc:push de:ld bc,(seed):ld hl,(seed)

sla l:rl h:add hl,bc:ld b,h:ld c,l:sla l:rl h

ld d,l:sla l:rl h:sla l:rl h:add hl,bc:ld b,h

ld c,l:ld h,d:ld l,&29:or a:sbc hl,bc:ld (seed),hl

pop de:pop bc:pop af:ret

seed dw 0



; -------------------*** Very Boring Graphics Data ***--------------------

;      --- Try bribing your younger brother to type this in! ---


bombgr

;The Alien's bomb.

db 0,12,3,192,48,12,0,195,0,60,3,192,48,12,0,192


missilegr

;The Player's missile

db 1,8,1,8,1,8,0,0,16,128,16,128,0,0,17,136


aliengr1

;The first Alien position

db 0,0,0,0,0,51,0,116,0,248,0,248,0,248,0,234

db 119,238,248,241,240,240,148,148,150,150,240,240,240,240,240,240

DB 0,0,0,0,204,0,226,0,241,0,241,0,241,0,49,0

db 0,234,0,234,0,251,0,249,0,252,0,50,0,0,0,0

db 86,100,64,32,32,64,136,0,196,17,136,0,0,0,0,0

db 117,0,117,0,117,0,249,0,226,0,200,0,0,0,0,0


aliengr2

;The second Alien position

db 0,0,0,51,0,116,0,248,0,248,0,248,0,234,0,234

db 248,241,240,240,148,148,150,150,240,240,240,240,240,240,234,117

db 0,0,204,0,226,0,241,0,241,0,241,0,117,0,117,0

db 0,234,0,234,0,234,0,224,0,224,0,226,0,238,0,0

db 100,86,34,68,0,0,0,0,0,0,0,0,0,0,0,0

db 117,0,117,0,117,0,112,0,112,0,114,0,119,0,0,0


basegr

;The data for the Player's base

db 0,0,0,0,0,0,0,0,0,7,0,120,0,120,0,120

db 1,8,16,128,19,132,54,194,120,225,240,240,240,240,240,240

db 0,0,0,0,0,0,0,0,14,0,225,0,225,0,225,0


banggr

;The explosion data

db 0,0,0,16,0,84,0,102,0,51,0,145,0,240,0,128

db 0,136,17,40,145,104,209,128,241,59,141,78,251,12,75,93

db 0,0,0,0,64,0,255,0,135,0,0,0,17,0,238,0

db 0,248,0,51,0,69,0,113,0,17,0,204,0,68,0,136

db 135,120,47,140,240,107,20,33,84,118,220,68,4,34,48,102

db 224,0,0,0,136,0,238,0,51,0,128,0,128,0,0,0



;----------------------------- THE END --------------------------


;Possible things you might like to add

;1. Sound effects          *

;2. Interrupt driven Music     *

;3. High score table           *

;4. Different ink/papercolours  *

;5. Useful shields

;6. More varied aliens

;7. Joystick or user defined controls

;8. Anything else you can think of...


;(* = covered in previous AJ Machine Code articles)
```