```
AppFilename         equ "NewFile"              ; What we're called (for file generation)

AppFirst            equ $8000                  ; First byte of code (uncontended memory)

                    zeusemulate "48K","ULA+"   ; Set the model and enable ULA+


; Start planting code here. (When generating a tape file we start saving from here)

                    org AppFirst               ; Start of application

; Definición de constantes y arrays
Array1              defb 1,2,3,4,5,6,7,8,9     ; Primer array
Array2              defb 2,2,1,1,1,9,1,1,1     ; Segundo array
Resultados          defs 9                     ; Tercer array para almacenar resultados

; Punto de entrada del programa
AppEntry:
                    ld hl, Array1              ; Cargar dirección del primer array en HL
                    ld de, Array2              ; Cargar dirección del segundo array en DE
                    ld bc, 9                   ; Longitud de los arrays (9 elementos)

; Bucle para comparar y procesar los arrays
ProcesarArrays:
                    ld a, (hl)                 ; Cargar el elemento del primer array en A
                    ld a, (de)                 ; Cargar el elemento del segundo array en E
                    cp e                       ; Comparar el contenido de A con E                    ; Comparar con
elemento del segundo array
                    jr z, Igual                ; Saltar si son iguales
                    jr c, Menor                ; Saltar si A < (DE)

                    ld a, (hl)                 ; Cargar elemento del primer array en A
                    ld a, (de)                 ; Cargar elemento del segundo array en E (usando LD, no ADD)
                    add a, e                   ; Sumar A + E
                    ld (hl), a                 ; Almacenar el resultado en el primer array
                    jr SiguienteElemento       ; Saltar al siguiente elemento


Menor:
                    ld a, (de)                 ; Cargar el valor de memoria en DE en A
                    sub (hl)                   ; Restar A - contenido de HL
                    ld (hl), a                 ; Almacenar el resultado en el primer array
                    jr SiguienteElemento       ; Saltar al siguiente elemento

Igual:
                    ; Si son iguales
                    ld a, 7                    ; Cargar valor 7 en A
                    ld (hl), a                 ; Almacenar 7 en el primer array

SiguienteElemento:
                    inc hl                     ; Avanzar al siguiente elemento de Array1
                    inc de                     ; Avanzar al siguiente elemento de Array2
                    djnz ProcesarArrays        ; Decrementar BC y repetir si no es cero

; Bucle para buscar el valor 7 en Resultados
```

```
Buscar7:
                ld hl, Resultados           ; Cargar dirección del array de resultados
                ld b, 9                     ; Longitud del array (9 elementos)

BuscarLoop:
                ld a, (hl)                  ; Cargar elemento del array de resultados en A
                cp 7                        ; Comparar con 7
                jr z, Encontrado7           ; Si es igual, saltar a Encontrado7
                inc hl                      ; Avanzar al siguiente elemento
                djnz BuscarLoop             ; Decrementar B y repetir si no es cero

                ; Si no se encontró el valor 7
                ld a, 0                     ; Cargar 0 en A (no encontrado)
                jp fin                      ; Saltar a terminar

Encontrado7:
                ; Si se encontró el valor 7
                ld a, 1                     ; Cargar 1 en A (encontrado)


fin             halt                        ;
                jp AppEntry                 ;


; Stop planting code after this. (When generating a tape file we save bytes below here)
AppLast         equ *-1                     ; The last used byte's address

; Generate some useful debugging commands

                profile AppFirst,AppLast-AppFirst+1 ; Enable profiling for all the code

; Setup the emulation registers, so Zeus can emulate this code correctly

Zeus_PC         equ AppEntry                ; Tell the emulator where to start
Zeus_SP         equ $FF40                   ; Tell the emulator where to put the stack

; These generate some output files

                ; Generate a SZX file
                output_szx AppFilename+".szx",$0000,AppEntry ; The szx file

                ; If we want a fancy loader we need to load a loading screen
;               import_bin AppFilename+".scr",$4000         ; Load a loading screen

                ; Now, also generate a tzx file using the loader
                output_tzx AppFilename+".tzx",AppFilename,"",AppFirst,AppLast-AppFirst,1,AppEntry ; A
tzx file using the loader
```