```
; NES Side-Scrolling Shooter Game
; Assemble with ca65/ld65

.segment "HEADER"
   .byte "NES", $1A    ; iNES header identifier
   .byte 2             ; 2x 16KB PRG-ROM
   .byte 1             ; 1x 8KB CHR-ROM
   .byte $01           ; mapper 0, vertical mirroring
   .byte $00           ; mapper 0
   .byte $00           ; No PRG-RAM
   .byte $00           ; NTSC
   .byte $00
   .byte $00, $00, $00, $00, $00

.segment "ZEROPAGE"
player_x:    .res 1
player_y:    .res 1
bullets:     .res 16  ; 8 bullets * 2 bytes (x,y)
enemies:     .res 16  ; 8 enemies * 2 bytes (x,y)
scroll_x:    .res 1
score:       .res 2
frame_cnt:   .res 1
enemy_timer: .res 1
btn_state:   .res 1
lives:       .res 1
invuln_timer: .res 1  ; Invulnerability frames after hit
temp1:       .res 1   ; Temp storage for collision detection
temp2:       .res 1
temp3:       .res 1
temp4:       .res 1

.segment "CODE"

.proc RESET
   SEI          ; disable IRQs
  CLD         ; disable decimal mode
   LDX #$40
   STX $4017    ; disable APU frame IRQ
   LDX #$FF
   TXS          ; Set up stack
   INX          ; now X = 0
   STX $2000    ; disable NMI
```

```
   STX $2001    ; disable rendering
   STX $4010    ; disable DMC IRQs

vblankwait1:
  BIT $2002
   BPL vblankwait1

clrmem:
   LDA #$00
   STA $0000, x
   STA $0100, x
   STA $0300, x
   STA $0400, x
   STA $0500, x
   STA $0600, x
   STA $0700, x
   LDA #$FE
   STA $0200, x
   INX
   BNE clrmem

vblankwait2:
  BIT $2002
   BPL vblankwait2

   ; Initialize variables
   LDA #$80
   STA player_x
   LDA #$70
   STA player_y

   LDA #$00
   STA scroll_x
   STA frame_cnt
   STA enemy_timer
   STA invuln_timer

   LDA #$03    ; Start with 3 lives
   STA lives

   ; Initialize bullets offscreen
   LDX #$00
```

```
init_bullets:
   LDA #$FF
   STA bullets, x
   INX
   CPX #$10
   BNE init_bullets

   ; Initialize enemies offscreen
   LDX #$00
init_enemies:
   LDA #$FF
   STA enemies, x
   INX
   CPX #$10
   BNE init_enemies

   ; Load palette
   LDA $2002
   LDA #$3F
   STA $2006
   LDA #$00
   STA $2006
   LDX #$00
LoadPalettes:
   LDA palette, x
   STA $2007
   INX
   CPX #$20
   BNE LoadPalettes

   ; Enable NMI and sprites
   LDA #%10000000
   STA $2000
   LDA #%00011110
   STA $2001

; Main game loop
GameLoop:
   ; Wait for NMI
   LDA frame_cnt
@wait:
  CMP frame_cnt
```

```
    BEQ @wait

    ; Check if game over
    LDA lives
    BNE @continue
   JMP GameOver
@continue:

    ; Decrease invulnerability timer
    LDA invuln_timer
    BEQ @no_invuln
   DEC invuln_timer
@no_invuln:

    ; Read controller
    JSR ReadController

    ; Update player
    JSR UpdatePlayer

    ; Update bullets
    JSR UpdateBullets

    ; Update enemies
    JSR UpdateEnemies

    ; Check collisions
    JSR CheckCollisions

    ; Draw lives
    JSR DrawLives

    ; Update scroll
   INC scroll_x

   JMP GameLoop
.endproc

; Game Over state
.proc GameOver
    ; Flash the screen or wait
    LDA frame_cnt
```

```
@wait:
  CMP frame_cnt
   BEQ @wait
  JMP GameOver
.endproc

; NMI - runs once per frame
.proc NMI
   PHA
   TXA
   PHA
   TYA
   PHA

   ; Sprite DMA
   LDA #$00
   STA $2003
   LDA #$02
   STA $4014

   ; Update scroll
   LDA #$00
   STA $2006
   STA $2006
   LDA scroll_x
   STA $2005
   LDA #$00
   STA $2005

  INC frame_cnt

   PLA
   TAY
   PLA
   TAX
   PLA
   RTI
.endproc

.proc ReadController
   LDA #$01
   STA $4016
```

```
  LDA #$00
  STA $4016

  LDA $4016   ; A
  AND #$01
  BEQ @skip_fire
  JSR FireBullet
@skip_fire:

  LDA $4016   ; B
  LDA $4016   ; Select
  LDA $4016   ; Start

  LDA $4016   ; Up
  AND #$01
  BEQ @skip_up
  LDA player_y
  CMP #$10
  BCC @skip_up
  DEC player_y
@skip_up:

  LDA $4016   ; Down
  AND #$01
  BEQ @skip_down
  LDA player_y
  CMP #$D0
  BCS @skip_down
  INC player_y
@skip_down:

  LDA $4016   ; Left
  AND #$01
  BEQ @skip_left
  LDA player_x
  CMP #$08
  BCC @skip_left
  DEC player_x
@skip_left:

  LDA $4016   ; Right
  AND #$01
```

```
    BEQ @skip_right
    LDA player_x
   CMP #$E8
    BCS @skip_right
   INC player_x
@skip_right:

    RTS
.endproc

.proc UpdatePlayer
    ; Check if invulnerable (flicker sprite)
    LDA invuln_timer
    BEQ @draw_normal
   AND #$04
    BEQ @hide_sprite

@draw_normal:
    ; Draw player sprite at OAM offset $00 (sprite 0)
    LDA player_y
    STA $0200
    LDA #$00    ; Tile
    STA $0201
    LDA #$00    ; Attributes
    STA $0202
    LDA player_x
    STA $0203
    RTS

@hide_sprite:
    ; Hide sprite during invulnerability flicker
    LDA #$FF
    STA $0200
    RTS
.endproc

.proc FireBullet
    ; Find free bullet slot
    LDX #$00
@find_slot:
    LDA bullets, x
   CMP #$FF
```

```
   BEQ @found_slot
   INX
   INX
   CPX #$10
   BNE @find_slot
   RTS  ; No free slots
@found_slot:
   LDA player_x
  CLC
  ADC #$08
   STA bullets, x
   INX
   LDA player_y
   STA bullets, x
   RTS
.endproc

.proc UpdateBullets
   LDX #$00
@loop:
   LDA bullets, x
  CMP #$FF
   BEQ @next

   ; Move bullet right
  CLC
  ADC #$03  ; Move 3 pixels per frame (faster bullets)
   STA bullets, x

   ; Check if off-screen right
  CMP #$F0
   BCC @draw
   LDA #$FF
   STA bullets, x
   INX
   LDA #$FF
   STA bullets, x
   DEX
  JMP @next

@draw:
   ; Draw bullet sprite
```

```
   ; X = 0,2,4,6,8,10,12,14
   ; Sprite slots 1-8: offsets $04,$08,$0C,$10,$14,$18,$1C,$20
   TXA
   PHA       ; Save X
   LSR       ; Divide by 2: 0,1,2,3,4,5,6,7
  CLC
  ADC #$01  ; Add 1 to get sprite number 1-8
   ASL
   ASL       ; Multiply by 4 for byte offset
   TAY       ; Y = $04,$08,$0C,$10,$14,$18,$1C,$20

   PLA
   TAX       ; Restore X
   INX
   LDA bullets, x  ; Get Y position
   STA $0200, y
   DEX

   INY
   LDA #$01    ; Tile
   STA $0200, y

   INY
   LDA #$01    ; Attributes
   STA $0200, y

   INY
   LDA bullets, x  ; Get X position
   STA $0200, y

@next:
   INX
   INX
   CPX #$10
   BNE @loop
   RTS
.endproc

.proc UpdateEnemies
   ; Spawn enemies
  INC enemy_timer
   LDA enemy_timer
```

```
  CMP #$50  ; Spawn every ~80 frames
   BCC @no_spawn
   LDA #$00
   STA enemy_timer
   JSR SpawnEnemy
@no_spawn:

   ; Update enemy positions
   LDX #$00
@loop:
   LDA enemies, x
  CMP #$FF
   BEQ @next_enemy  ; Skip inactive enemies

   ; Move enemy left
   SEC
  SBC #$02  ; Move 2 pixels left per frame
   STA enemies, x

   ; Check if off-screen
  CMP #$04
   BCS @draw_this_enemy

   ; Remove off-screen enemy
   LDA #$FF
   STA enemies, x
   INX
   STA enemies, x
   DEX
  JMP @next_enemy  ; Don't draw, just move to next

@draw_this_enemy:
   ; Draw enemy sprite
   TXA
   PHA       ; Save X
   LSR       ; Divide by 2
  CLC
  ADC #$09  ; Sprite number 9-16
   ASL
   ASL       ; Multiply by 4
   TAY
```

```
   PLA
   TAX       ; Restore X

   ; Y position
   INX
   LDA enemies, x
   STA $0200, y
   DEX

   ; Tile
   INY
   LDA #$02
   STA $0200, y

   ; Attributes
   INY
   LDA #$02
   STA $0200, y

   ; X position
   INY
   LDA enemies, x
   STA $0200, y

@next_enemy:
   INX
   INX
   CPX #$10
   BNE @loop
   RTS
.endproc

.proc SpawnEnemy
   ; Find free enemy slot
   LDX #$00
@find_slot:
   LDA enemies, x
  CMP #$FF
   BEQ @found_slot
   INX
   INX
   CPX #$10
```

```
    BNE @find_slot
    RTS
@found_slot:
    ; Spawn at right side of screen
    LDA #$E8  ; X position (232 - visible on screen)
    STA enemies, x
    INX
    ; Random Y position
    LDA frame_cnt
   AND #$7F
   CLC
   ADC #$20
   CMP #$C0  ; Keep within reasonable bounds
    BCC @y_ok
    LDA #$80
@y_ok:
    STA enemies, x
    RTS
.endproc

.proc CheckCollisions
    ; Check bullet-enemy collisions
    LDX #$00
@bullet_loop:
    LDA bullets, x
   CMP #$FF
    BEQ @next_bullet

    ; Store bullet X in temp1
    STA temp1
    INX
    LDA bullets, x
    STA temp2  ; Store bullet Y in temp2
    DEX

    LDY #$00
@enemy_loop:
    LDA enemies, y
   CMP #$FF
    BEQ @next_enemy

    ; Store enemy X in temp3
```

```
    STA temp3
    INY
    LDA enemies, y
    STA temp4  ; Store enemy Y in temp4
    DEY

    ; Check X collision (within 12 pixels)
    LDA temp1  ; Bullet X
    SEC
   SBC temp3  ; Enemy X
    BCS @x_positive
    ; Negative difference
    EOR #$FF
   CLC
   ADC #$01
@x_positive:
   CMP #$0C
    BCS @next_enemy

    ; Check Y collision (within 12 pixels)
    LDA temp2  ; Bullet Y
    SEC
   SBC temp4  ; Enemy Y
    BCS @y_positive
    ; Negative difference
    EOR #$FF
   CLC
   ADC #$01
@y_positive:
   CMP #$0C
    BCS @next_enemy

    ; Collision detected!
    LDA #$FF
    STA bullets, x
    STA enemies, y
   JMP @next_bullet

@next_enemy:
    INY
    INY
    CPY #$10
```

```
   BNE @enemy_loop

@next_bullet:
   INX
   INX
   CPX #$10
   BNE @bullet_loop

   ; Check player-enemy collisions
   LDA invuln_timer
   BNE @skip_player_collision  ; Skip if invulnerable

   LDY #$00
@player_enemy_loop:
   LDA enemies, y
  CMP #$FF
   BEQ @next_player_enemy

   ; Store enemy X in temp3
   STA temp3
   INY
   LDA enemies, y
   STA temp4  ; Store enemy Y in temp4
   DEY

   ; Check X collision
   LDA player_x
   SEC
  SBC temp3
   BCS @px_positive
   EOR #$FF
  CLC
  ADC #$01
@px_positive:
  CMP #$10
   BCS @next_player_enemy

   ; Check Y collision
   LDA player_y
   SEC
  SBC temp4
   BCS @py_positive
```

```
    EOR #$FF
    CLC
    ADC #$01
@py_positive:
    CMP #$10
    BCS @next_player_enemy

    ; Player hit!
    LDA #$FF
    STA enemies, y
    DEC lives
    LDA #$78  ; 2 seconds of invulnerability (120 frames)
    STA invuln_timer
    JMP @skip_player_collision

@next_player_enemy:
    INY
    INY
    CPY #$10
    BNE @player_enemy_loop

@skip_player_collision:
    RTS
.endproc

.proc DrawLives
    ; Draw life indicators in top-left corner (sprites 17-19, offsets $44-$4F)
    LDX #$00
@loop:
    CPX lives
    BCS @empty_heart

    ; Draw filled heart
    TXA
    ASL
    ASL       ; Multiply by 4 (each sprite is 4 bytes)
    CLC
    ADC #$44  ; Start at sprite 17
    TAY

    LDA #$08
    STA $0200, y
```

```
    INY
    LDA #$00  ; Player tile as heart
    STA $0200, y
    INY
    LDA #$00
    STA $0200, y
    INY
    TXA
    ASL
    ASL
    ASL
   CLC
   ADC #$08
    STA $0200, y
   JMP @next_life

@empty_heart:
    ; Hide sprite
    TXA
    ASL
    ASL
   CLC
   ADC #$44
    TAY
    LDA #$FF
    STA $0200, y

@next_life:
    INX
    CPX #$03  ; Max 3 lives to display
    BNE @loop
    RTS
.endproc

palette:
    .byte $0F,$00,$10,$30  ; Background palette 0
    .byte $0F,$14,$24,$34  ; Background palette 1
    .byte $0F,$1B,$2B,$3B  ; Background palette 2
    .byte $0F,$12,$22,$32  ; Background palette 3
    .byte $0F,$30,$10,$00  ; Sprite palette 0 (player - white)
    .byte $0F,$27,$17,$07  ; Sprite palette 1 (bullets - yellow)
    .byte $0F,$16,$26,$36  ; Sprite palette 2 (enemies - red)
```

```
    .byte $0F,$1A,$2A,$3A  ; Sprite palette 3

.segment "VECTORS"
    .word NMI
    .word RESET
    .word 0

; CHR ROM
.segment "CHARS"
; Tile $00 - Player ship
    .byte %00011000
    .byte %00111100
    .byte %01111110
    .byte %11111111
    .byte %01111110
    .byte %00111100
    .byte %00011000
    .byte %00000000

    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000

; Tile $01 - Bullet
    .byte %00000000
    .byte %00000000
    .byte %00011000
    .byte %00111100
    .byte %00111100
    .byte %00011000
    .byte %00000000
    .byte %00000000


    .byte %00000000
    .byte %00000000
    .byte %00000000
```

```
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000

; Tile $02 - Enemy
    .byte %01111110
    .byte %11111111
    .byte %11011011
    .byte %11111111
    .byte %01111110
    .byte %00111100
    .byte %00011000
    .byte %00000000

    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000
    .byte %00000000

; Fill rest of CHR with blank tiles
    .res $2000 - 48, $00
```