.segment "HEADER"

.byte $4E, $45, $53, $1A ; iNES header identifier

.byte 2 ; 2x 16KB PRG code

.byte 1 ; 1x 8KB CHR data

.byte $01, $00 ; Mapper 0, vertical mirroring

.segment "ZEROPAGE"

controller1: .res 1 ; Reserve 1 byte for controller state

color\_index: .res 1 ; Reserve 1 byte to store the current index of the color

.segment "VECTORS"

.addr nmi ; NMI vector

.addr reset ; Reset vector

.addr 0 ; IRQ vector (unused)

.segment "STARTUP"

.segment "CODE"

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: ; First wait for vblank to make sure PPU is ready

BIT $2002

BPL vblankwait1

clrmem:

LDA #$00

STA $0000, x

STA $0100, x

STA $0200, x

STA $0400, x

STA $0500, x

STA $0600, x

STA $0700, x

LDA #$FE

STA $0300, x

INX

BNE clrmem

vblankwait2: ; Second wait for vblank, PPU is ready after this

BIT $2002

BPL vblankwait2

LoadPalettes:

LDA $2002 ; read PPU status to reset the high/low latch

LDA #$3F

STA $2006 ; write the high byte of $3F00 address

LDA #$00

STA $2006 ; write the low byte of $3F00 address

LDX #$00 ; start out at 0

LoadPalettesLoop:

LDA palette, x ; load data from address (palette + the value in x)

; 1st time through loop it will load palette+0

; 2nd time through loop it will load palette+1

; 3rd time through loop it will load palette+2

; etc

STA $2007 ; write to PPU

INX ; X = X + 1

CPX #$20 ; Compare X to hex $10, decimal 16 - copying 16 bytes = 4 sprites

BNE LoadPalettesLoop ; Branch to LoadPalettesLoop if compare was Not Equal to zero

; if compare was equal to 32, keep going down

LoadSprites:

LDX #$00 ; start at 0

LoadSpritesLoop:

LDA sprites, x ; load data from address (sprites + x)

STA $0200, x ; store into RAM address ($0200 + x)

INX ; X = X + 1

CPX #$20 ; Compare X to hex $20, decimal 32

BNE LoadSpritesLoop ; Branch to LoadSpritesLoop if compare was Not Equal to zero

; if compare was equal to 32, keep going down

LDA #%10000000 ; enable NMI, sprites from Pattern Table 1

STA $2000

LDA #%00010000 ; enable sprites

STA $2001

Forever:

JMP Forever ;jump back to Forever, infinite loop

nmi:

LDA #$00

STA $2003 ; set the low byte (00) of the RAM address

LDA #$02

STA $4014 ; set the high byte (02) of the RAM address, start the transfer

hello1:

ldx #$00 ; Set SPR-RAM address to 0

stx $2003

@loop: lda hello, x ; Load the hello message into SPR-RAM

sta $2004

inx

cpx #$1c

bne @loop

LatchController:

LDA #$01

STA $4016

LDA #$00

STA $4016 ; tell both the controllers to latch buttons

ReadA:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadADone ; branch to ReadADone if button is NOT pressed (0)

jsr BassDrum

BassDrum:

; Enable the noise channel

lda #%00001000 ; Bit 3 set = enable noise channel

sta $4015

; Configure Noise Channel Envelope

lda #%00110100 ; Volume 4, Envelope disabled, decay rate fast

; Bit 7 = 0 (disable envelope)

; Bit 6 = 1 (constant volume)

; Bit 5-0 = 4 (volume)

sta $400C ; Write to Noise Envelope/Volume register

; Configure Noise Frequency

lda #%00111111 ; Frequency index = $23 (higher frequency for sharpness)

; Bit 7 = 0 (non-looping random noise)

; Bits 4-0 = $23 (frequency index)

sta $400E ; Write to Noise Period register

; Restart the length counter

lda #%00001000 ; Load length counter (short duration)

sta $400F ; Writing to $400F also resets envelope and length counter

DelayLoopX1:

ldx #$bb ; Outer loop for a longer delay

DelayLoopOuterX1:

ldy #$bb ; Inner loop

DelayLoopInnerX1:

dey

bne DelayLoopInnerX1 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX1 ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadADone: ; handling this button is done

ReadB:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadBDone ; branch to ReadADone if button is NOT pressed (0)

jsr SnareDrum

SnareDrum:

; Enable the noise channel

lda #%00001000 ; Bit 3 set = enable noise channel

sta $4015

; Configure Noise Channel Envelope

lda #%00110100 ; Volume 4, Envelope disabled, decay rate fast

; Bit 7 = 0 (disable envelope)

; Bit 6 = 1 (constant volume)

; Bit 5-0 = 4 (volume)

sta $400C ; Write to Noise Envelope/Volume register

; Configure Noise Frequency

lda #%00100011 ; Frequency index = $23 (higher frequency for sharpness)

; Bit 7 = 0 (non-looping random noise)

; Bits 4-0 = $23 (frequency index)

sta $400E ; Write to Noise Period register

; Restart the length counter

lda #%00001000 ; Load length counter (short duration)

sta $400F ; Writing to $400F also resets envelope and length counter

DelayLoopX2:

ldx #$bb ; Outer loop for a longer delay

DelayLoopOuterX2:

ldy #$bb ; Inner loop

DelayLoopInnerX2:

dey

bne DelayLoopInnerX1 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX1 ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadBDone: ; handling this button is done

ReadSelect:

LDA $4016 ; Read player 1 controller

AND #%00000001 ; Only look at bit 0 (Select button)

BEQ ReadSelectDone ; If Select is NOT pressed, skip

DelayLoopXcolor1:

ldx #$aa ; Outer loop for a longer delay

DelayLoopOuterXcolor1:

ldy #$aa ; Inner loop

DelayLoopInnerXcolor1:

dey

bne DelayLoopInnerXcolor1 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterXcolor1 ; Repeat outer loop until X = 0

; Increment color\_index to cycle through colors

lda color\_index ; Load current color index

clc ; Clear carry (safe addition)

adc #$01 ; Add 1 to move to the next color

cmp #$05 ; Check if it exceeds the number of colors (5)

bcc NoReset ; If below 5, continue

lda #$00 ; Reset to the first color

NoReset:

sta color\_index ; Store updated color index

; Load the color based on the current index

lda $2002 ; Reset PPU address latch

lda #$3F

sta $2006 ; Set VRAM address to palette

lda #$00

sta $2006

lda color\_index ; Load current color index

asl ; Multiply index by 2 (for table lookup)

tay ; Store in Y register for lookup

lda color\_table, y ; Load the color value from the table

sta $2007 ; Write it to the background color

DelayLoopXcolor:

ldx #$ff ; Outer loop for a longer delay

DelayLoopOuterXcolor:

ldy #$ff ; Inner loop

DelayLoopInnerXcolor:

dey

bne DelayLoopInnerXcolor ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterXcolor ; Repeat outer loop until X = 0

ReadSelectDone:

ReadStart:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadStartDone ; branch to ReadADone if button is NOT pressed (0)

jsr Bass2

Bass2:

lda #%00000111 ;enable Sq1, Sq2 and Tri channels

sta $4015

;Triangle

lda #%10000001 ;Triangle channel on

sta $4008

lda #$DF ;$042 is a G# in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopX6:

ldx #$bb ; Outer loop for a longer delay

DelayLoopOuterX6:

ldy #$bb ; Inner loop

DelayLoopInnerX6:

dey

bne DelayLoopInnerX6 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX6 ; Repeat outer loop until X = 0

lda #$00

sta $4015

ReadStartDone:

ReadUp:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadUpDone ; branch to ReadADone if button is NOT pressed (0)

jsr HiHat

HiHat:

; Enable the noise channel

lda #%00001000 ; Bit 3 set = enable noise channel

sta $4015

; Configure Noise Channel Envelope

lda #%00110100 ; Volume 4, Envelope disabled, decay rate fast

; Bit 7 = 0 (disable envelope)

; Bit 6 = 1 (constant volume)

; Bit 5-0 = 4 (volume)

sta $400C ; Write to Noise Envelope/Volume register

; Configure Noise Frequency

lda #%00100000 ; Frequency index = $23 (higher frequency for sharpness)

; Bit 7 = 0 (non-looping random noise)

; Bits 4-0 = $23 (frequency index)

sta $400E ; Write to Noise Period register

; Restart the length counter

lda #%00001000 ; Load length counter (short duration)

sta $400F ; Writing to $400F also resets envelope and length counter

DelayLoopX3:

ldx #$bb ; Outer loop for a longer delay

DelayLoopOuterX3:

ldy #$bb ; Inner loop

DelayLoopInnerX3:

dey

bne DelayLoopInnerX3 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX3 ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadUpDone:

ReadDown:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadDownDone ; branch to ReadADone if button is NOT pressed (0)

jsr Tom

Tom:

; Enable the noise channel

lda #%00001000 ; Bit 3 set = enable noise channel

sta $4015

; Configure Noise Channel Envelope

lda #%00110110 ; Volume 6, Envelope disabled, decay rate fast

; Bit 7 = 0 (disable envelope)

; Bit 6 = 1 (constant volume)

; Bit 5-0 = 6 (volume)

sta $400C ; Write to Noise Envelope/Volume register

; Configure Noise Frequency

lda #%00011110 ; Frequency index = $1E (low frequency, deep sound)

; Bit 7 = 0 (non-looping random noise)

; Bits 4-0 = $1E (frequency index)

sta $400E ; Write to Noise Period register

; Restart the length counter

lda #%00001000 ; Load length counter (short duration)

sta $400F ; Writing to $400F also resets envelope and length counter

DelayLoopX4:

ldx #$5E ; Outer loop for a longer delay

DelayLoopOuterX4:

ldy #$5F ; Inner loop

DelayLoopInnerX4:

dey

bne DelayLoopInnerX4 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX4 ; Repeat outer loop until X = 0

lda #$00

sta $4015

ReadDownDone:

ReadLeft:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadLeftDone ; branch to ReadADone if button is NOT pressed (0)

; add instructions here to do something when button IS pressed (1)

jsr Synth

Synth:

; Enable the noise channel

lda #%00001000 ; Bit 3 set = enable noise channel

sta $4015

; Configure Noise Channel Envelope

lda #%00110100 ; Volume 4, Envelope disabled, decay rate fast

; Bit 7 = 0 (disable envelope)

; Bit 6 = 1 (constant volume)

; Bit 5-0 = 4 (volume)

sta $400C ; Write to Noise Envelope/Volume register

; Configure Noise Frequency

lda #%00100000 ; Frequency index = $23 (higher frequency for sharpness)

; Bit 7 = 0 (non-looping random noise)

; Bits 4-0 = $23 (frequency index)

sta $400E ; Write to Noise Period register

; Restart the length counter

lda #%00001000 ; Load length counter (short duration)

sta $400F ; Writing to $400F also resets envelope and length counter

DelayLoopX5:

ldx #$10 ; Outer loop for a longer delay

DelayLoopOuterX5:

ldy #$10 ; Inner loop

DelayLoopInnerX5:

dey

bne DelayLoopInnerX5 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX5 ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadLeftDone:

ReadRight:

LDA $4016 ; player 1 - A

AND #%00000001 ; only look at bit 0

BEQ ReadRightDone ; branch to ReadADone if button is NOT pressed (0)

jsr Bass

Bass:

lda #%00000111 ;enable Sq1, Sq2 and Tri channels

sta $4015

;Triangle

lda #%10000001 ;Triangle channel on

sta $4008

lda #$FF ;$042 is a G# in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopX9:

ldx #$5b ; Outer loop for a longer delay

DelayLoopOuterX9:

ldy #$5b ; Inner loop

DelayLoopInnerX9:

dey

bne DelayLoopInnerX9 ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterX9 ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadRightDone:

ReadAPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadAPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr CSharpNote

CSharpNote:

;load sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Square 1 (C# note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$C9 ; $0C9 is a C# in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2A:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2A:

ldy #$BA ; Inner loop

DelayLoopInnerP2A:

dey

bne DelayLoopInnerP2A ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2A ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadAPlayer2Done: ; handling this button is done

ReadBPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadBPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr ENote

ENote:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Square 2 (E note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$A9 ; $0A9 is an E in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2B:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2B:

ldy #$BA ; Inner loop

DelayLoopInnerP2B:

dey

bne DelayLoopInnerP2B ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2B ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadBPlayer2Done:

ReadSelectPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadSelectPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr GSharpNote

GSharpNote:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Square 2 (E note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$84 ; $0A9 is an E in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2C:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2C:

ldy #$BA ; Inner loop

DelayLoopInnerP2C:

dey

bne DelayLoopInnerP2C ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2C ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadSelectPlayer2Done:

ReadStartPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadStartPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr BNote

BNote:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Triangle (B note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$6F ; B in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2D:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2D:

ldy #$BA ; Inner loop

DelayLoopInnerP2D:

dey

bne DelayLoopInnerP2D ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2D ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadStartPlayer2Done:

ReadUpPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadUpPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr CSharpNote2

CSharpNote2:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Triangle (C# note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$64 ; C# in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2E:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2E:

ldy #$BA ; Inner loop

DelayLoopInnerP2E:

dey

bne DelayLoopInnerP2E ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2E ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadUpPlayer2Done:

ReadDownPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadDownPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr DNote

DNote:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Triangle (D note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$5E ; D in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2F:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2F:

ldy #$BA ; Inner loop

DelayLoopInnerP2F:

dey

bne DelayLoopInnerP2F ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2F ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadDownPlayer2Done:

ReadLeftPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadLeftPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr ENote2

ENote2:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Triangle (E note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$54 ; E in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2G:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2G:

ldy #$BA ; Inner loop

DelayLoopInnerP2G:

dey

bne DelayLoopInnerP2G ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2G ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadLeftPlayer2Done:

ReadRightPlayer2:

LDA $4017 ; player 2 - A

AND #%00000001 ; only look at bit 0

BEQ ReadRightPlayer2Done ; branch to ReadADone if button is NOT pressed (0)

jsr FSharpNote

FSharpNote:

; Enable Sound channel

lda #%00000001

sta $4015 ; Enable Square 1 channel, disable others

lda #%00010110

sta $4015 ; Enable Square 2, Triangle, and DMC channels. Disable Square 1 and Noise.

lda #$0F

sta $4015 ; Enable Square 1, Square 2, Triangle, and Noise channels. Disable DMC.

lda #%00000111 ; Enable Square 1, Square 2, and Triangle channels

sta $4015

; Triangle (F# note)

lda #%10000001 ; Triangle channel on

sta $4008

lda #$4A ; F# in NTSC mode

sta $400A

lda #$00

sta $400B

DelayLoopP2H:

ldx #$BA ; Outer loop for a longer delay

DelayLoopOuterP2H:

ldy #$BA ; Inner loop

DelayLoopInnerP2H:

dey

bne DelayLoopInnerP2H ; Repeat inner loop until Y = 0

dex

bne DelayLoopOuterP2H ; Repeat outer loop until X = 0

lda #$00

sta $4015

rti

ReadRightPlayer2Done:

RTI ; return from interrupt

palette:

; Background Palette

.byte $0f, $22, $00, $00 ; $0f controls background color (grayish blue) and $22 controls flicker (light blue)

.byte $0f, $00, $00, $00

.byte $0f, $00, $00, $00

.byte $0f, $00, $00, $00

; Sprite Palette

.byte $0c, $20, $00, $00 ; second value controls the font color

.byte $0f, $00, $00, $00

.byte $0f, $00, $00, $00

.byte $0f, $00, $00, $00

.byte $0f, $00, $00, $00

hello:

.byte $00, $00, $00, $00 ; Why do I need these here?

.byte $00, $00, $00, $00

.byte $6c, $00, $00, $6c

.byte $6c, $01, $00, $76

.byte $ff, $ff, $ff, $ff

.byte $6c, $02, $00, $80

.byte $6c, $03, $00, $8b ; 1: y position of O 2:? 3: color of O 4: x position of O

.segment "SPRITES"

sprites:

; Sprite Palette

.byte $80, $32, $00, $80 ;sprite 0

.byte $80, $33, $00, $88 ;sprite 1

.byte $88, $34, $00, $80 ;sprite 2

.byte $88, $35, $00, $88 ;sprite 3

.segment "CHARS"

.byte %00001111 ; J (00)

.byte %00001111

.byte %00000110

.byte %00000110

.byte %00000110

.byte %11000110

.byte %11000110

.byte %11111110

.byte $00, $00, $00, $00, $00, $00, $00, $00

.byte %01111110 ; A (01)

.byte %11000011

.byte %11000011

.byte %11111111

.byte %11111111

.byte %11000011

.byte %11000011

.byte %11000011

.byte $00, $00, $00, $00, $00, $00, $00, $00

.byte %11000011 ; M (02)

.byte %11111111

.byte %11011011

.byte %11011011

.byte %11000011

.byte %11000011

.byte %11000011

.byte %11000011

.byte $00, $00, $00, $00, $00, $00, $00, $00

.byte %00011000 ; ! (03)

.byte %00011000

.byte %00011000

.byte %00011000

.byte %00011000

.byte %00000000

.byte %00011000

.byte %00011000

.byte $00, $00, $00, $00, $00, $00, $00, $00

.segment "RODATA"

color\_table:

.byte $17, $1C, $1c, $1B, $2a ; The cycle of colors