The Beautiful Things of

Nintendo GAME BOY


NINTENDO
SWITCH


NINTENDO DS


GAME BOY
Nintendo
GAME BOY ADVANCE

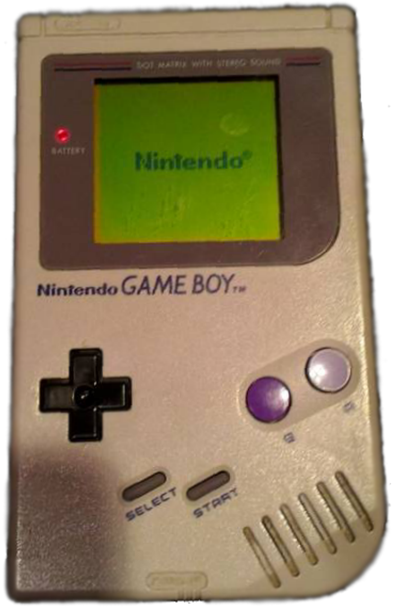

GAME BOY
Nintendo
GAME BOY COLOR

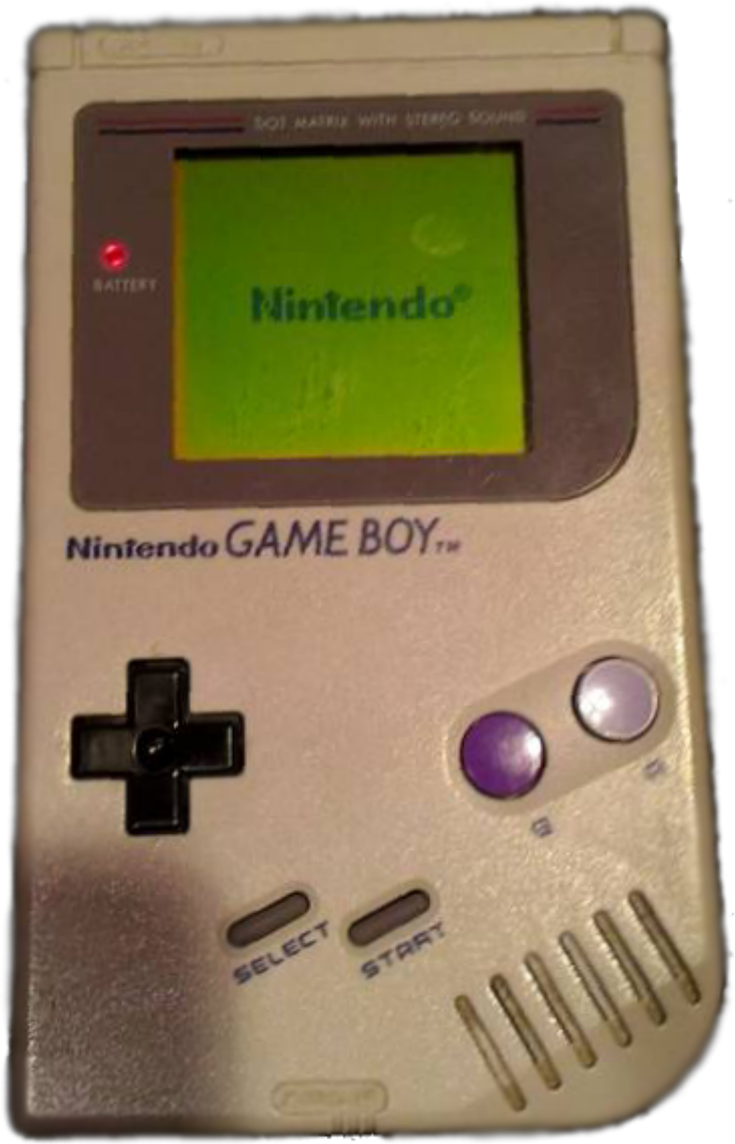

NINTENDO
SWITCH


NINTENDO DS


GAME BOY
Nintendo
GAME BOY ADVANCE


GAME BOY
Nintendo
GAME BOY COLOR


Nintendo
Nintendo GAME BOY
SELECT
START



Screen

LCD 160 x 144 pixel

4 Colors

Screen

LCD 160 x 144 pixel

4 Colors

Sound

3.5mm Jack

1x Speaker

Buttons

D-PAD, A, B, START, SELECT, VOLUME

PSU

4 x AA Batteries

3.5mm DC Jack

Screen

LCD 160 x 144 pixel

4 Colors

Sound

3.5mm Jack

1x Speaker

Buttons

D-PAD, A, B, START, SELECT, VOLUME

PSU

4 x AA Batteries

3.5mm DC Jack

Screen

LCD 160 x 144 pixel

4 Colors

Sound

3.5mm Jack

1x Speaker

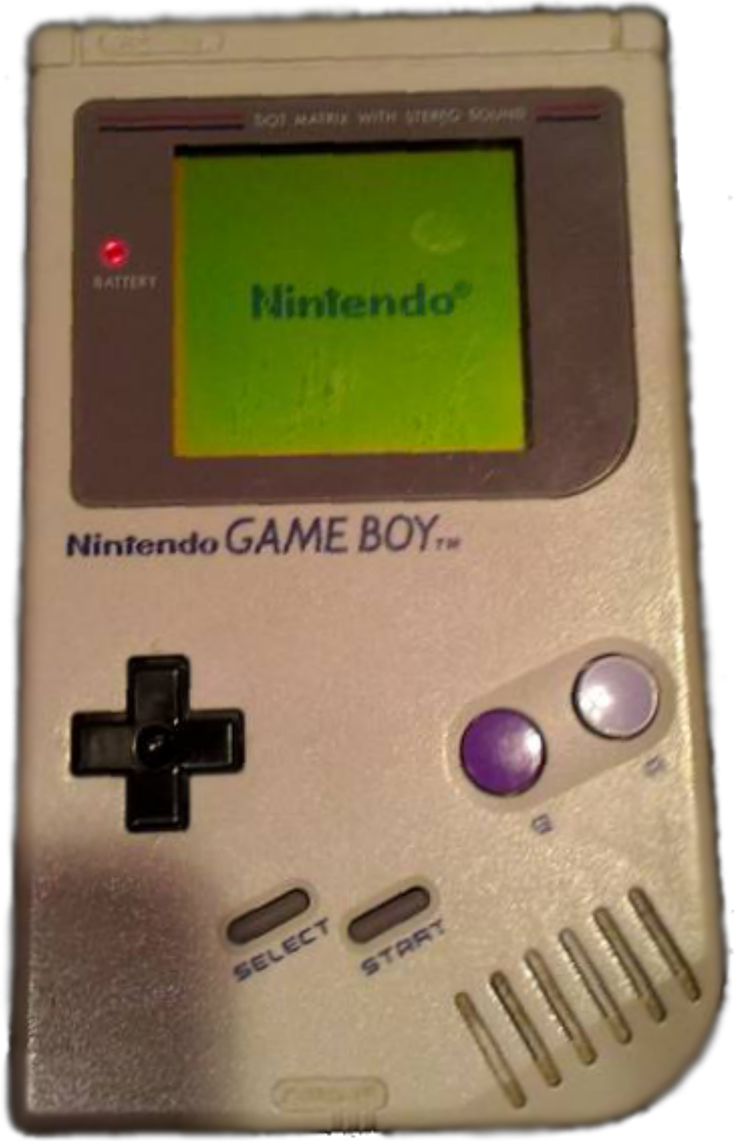
Buttons

D-PAD, A, B, START, SELECT, VOLUME

PSU

4 x AA Batteries

3.5mm DC Jack



Screen

LCD 160 x 144 pixel

4 Colors

Sound

3.5mm Jack

1x Speaker

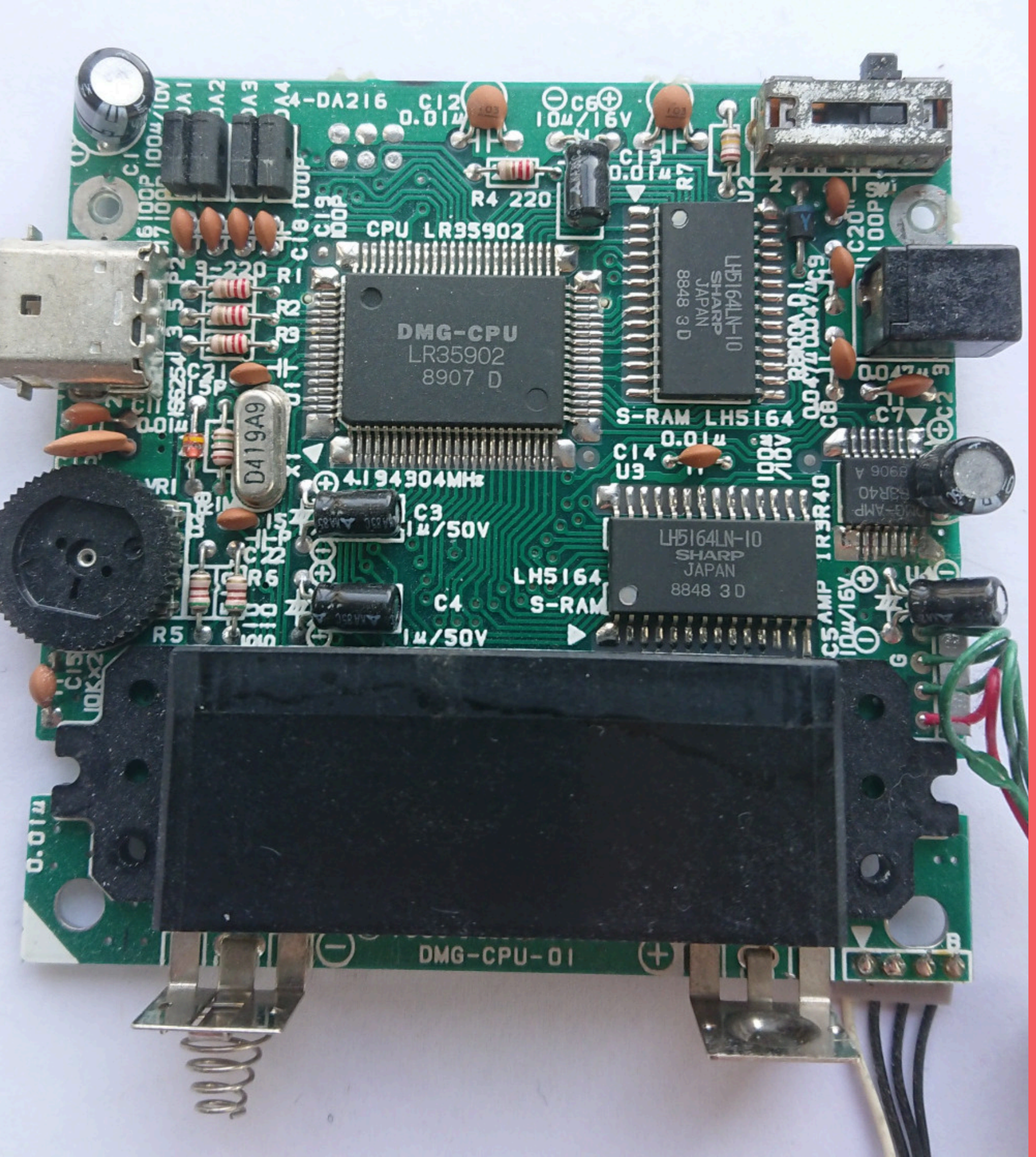
Buttons

D-PAD, A, B, START, SELECT, VOL DIAL

PSU

4 x AA Batteries

3.5mm DC Jack



Processor

Sharp LR35902

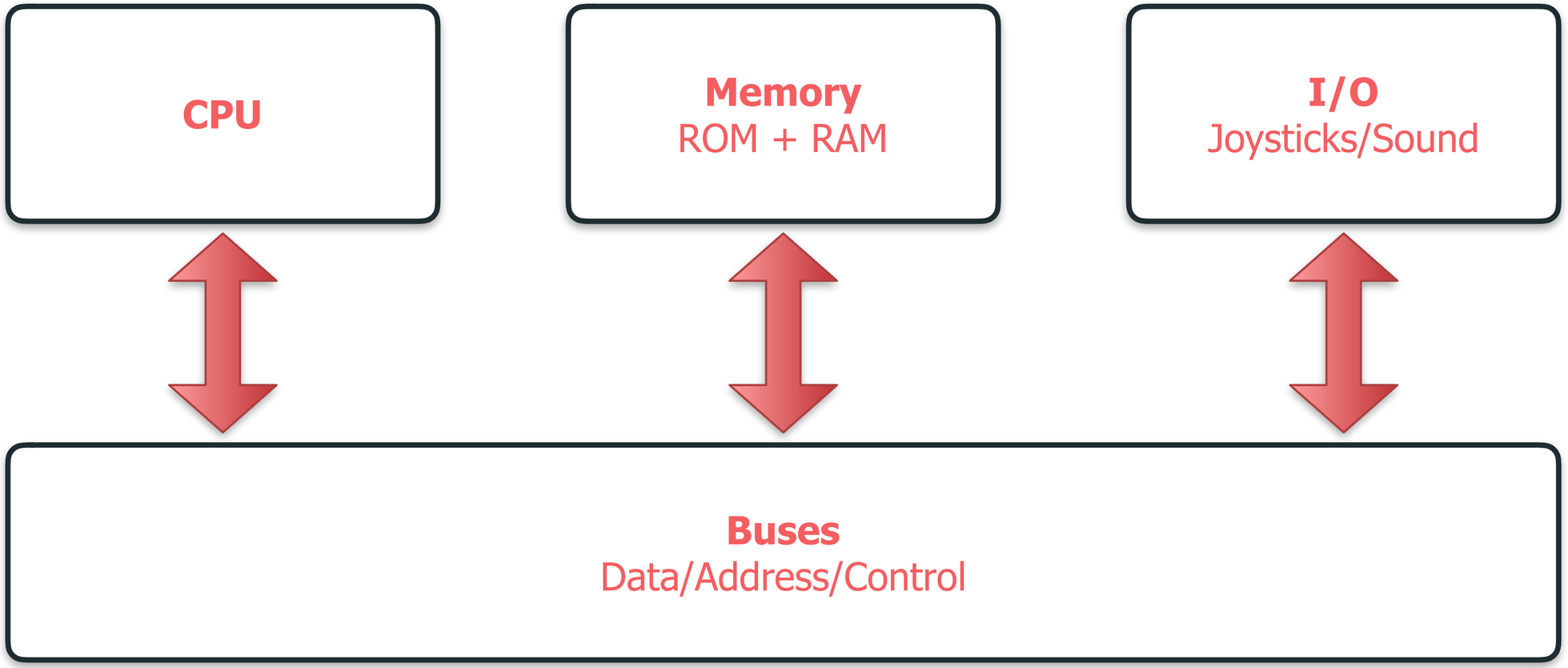
RAM

8 kB + 8kB (video)

ROM

0x100 bootstrap + Cartridge

Computer Architecture



Sharp **LR35902**

SHARP

Sharp **LR35902**

Sharp **LR35902**

4.19 MHz

Sharp **LR35902**

4.19 MHz

8 Bits

255

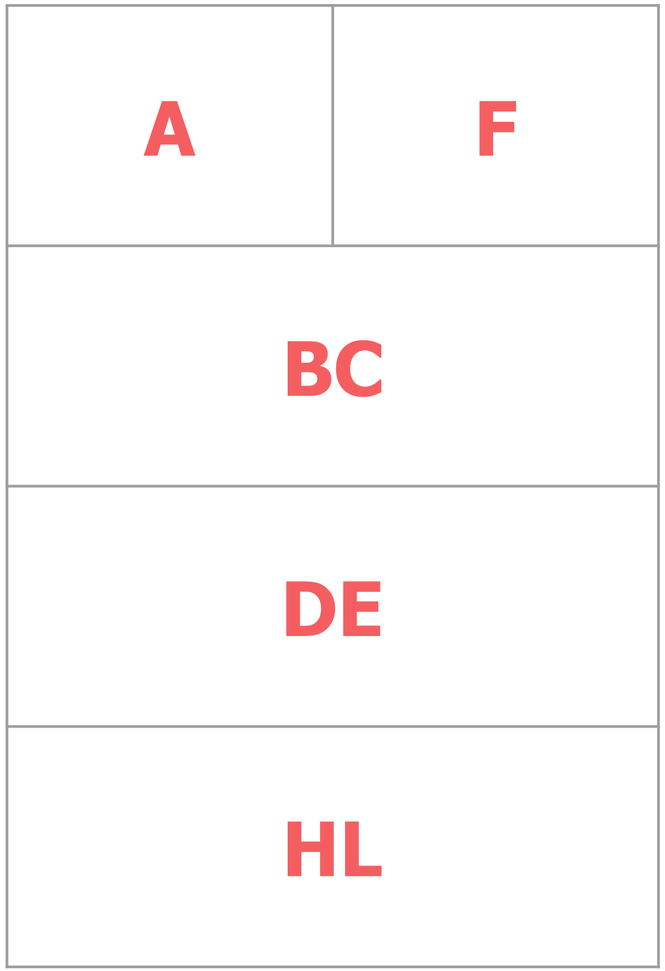
0xFF

0b1111 1111

General Registers

A	F
B	C
D	E
H	L

General Registers



Boot Loader

Boot Loader

256 bytes

Boot Loader

256 bytes

First piece of code

Boot Loader

256 bytes

First piece of code

Sets **everything** up

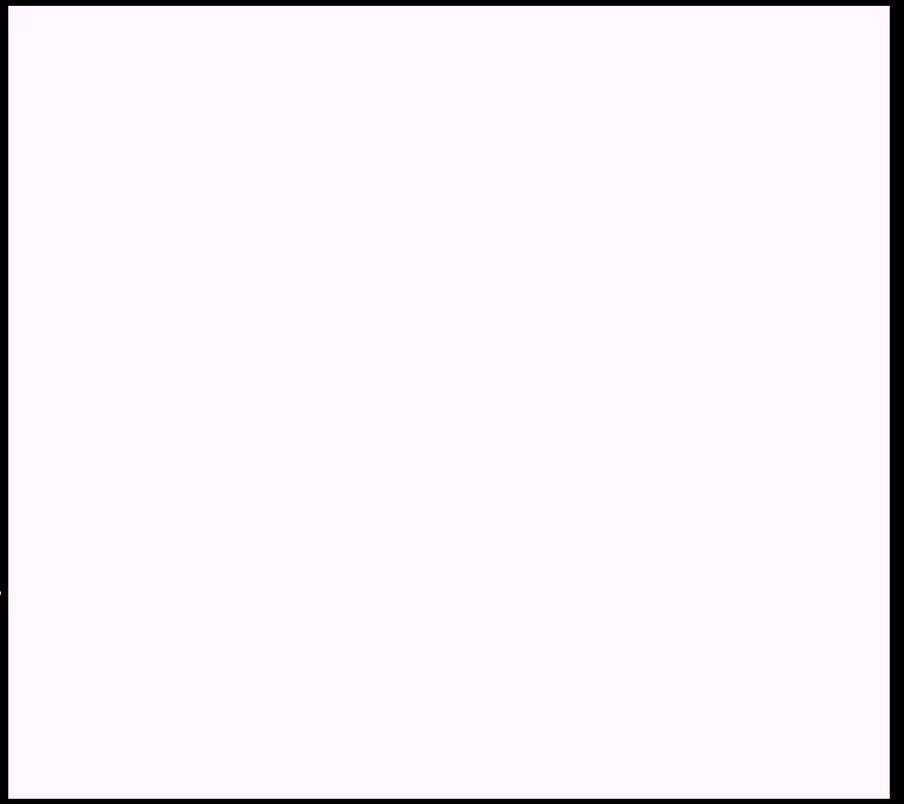
```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f
```

```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053
```

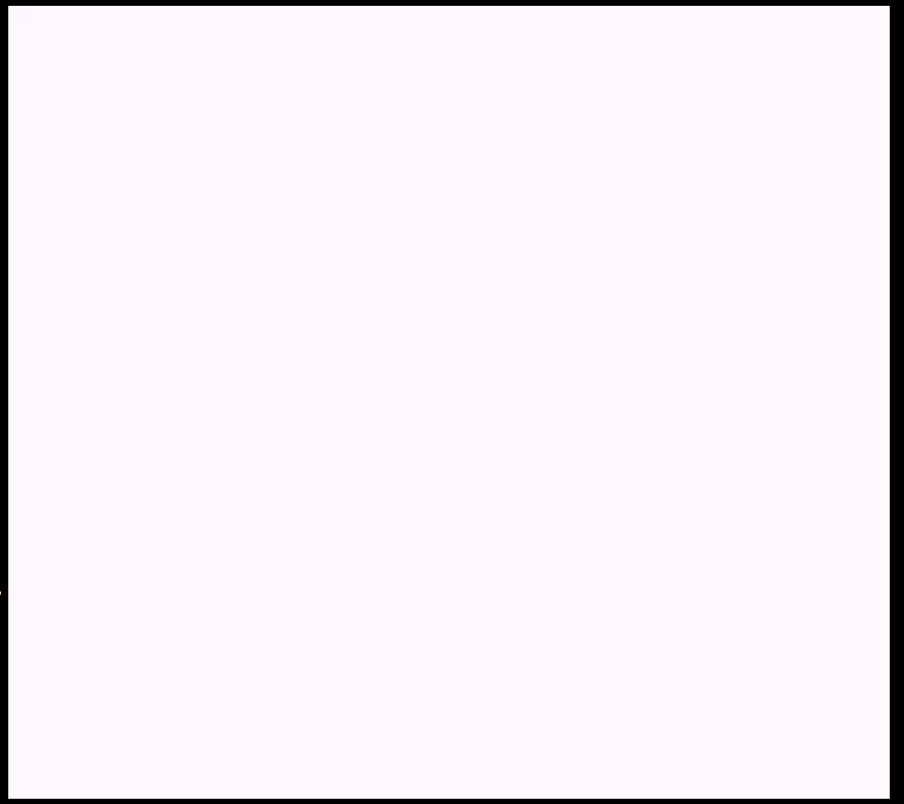

MODEL NO. MGB-001
Pat. Issued and Pend.
BATTERY: LR03(AAA)x2
MADE IN CHINA


MODEL NO. MGB-001
Pat. Issued and Pend.
BATTERY: LR03(AAA)x2
MADE IN CHINA

```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053

     ; === Scroll logo on screen, and play logo sound===

Addr_0055:
     LD H,A              ; $0055  Initialize scroll count, H=0
     LD A,$64            ; $0056
     LD D,A              ; $0058  set loop count, D=$64
     LD ($FF00+$42),A; $0059  Set vertical scroll register
     LD A,$91            ; $005b
     LD ($FF00+$40),A; $005d  Turn on LCD, showing Background
     INC B               ; $005f  Set B=1
Addr_0060:
     LD E,$02            ; $0060
Addr_0062:
     LD C,$0c            ; $0062
Addr_0064:
     LD A,($FF00+$44); $0064  wait for screen frame
     CP $90              ; $0066
     JR NZ, Addr_0064; $0068
     DEC C               ; $006a
     JR NZ, Addr_0064; $006b
     DEC E               ; $006d
     JR NZ, Addr_0062; $006e

     LD C,$13            ; $0070
     INC H               ; $0072  increment scroll count
     LD A,H              ; $0073
     LD E,$83            ; $0074
     CP $62              ; $0076  $62 counts in, play sound #1
     JR Z, Addr_0080 ; $0078
     LD E,$c1            ; $007a
     CP $64              ; $007c
     JR NZ, Addr_0086; $007e  $64 counts in, play sound #2
Addr_0080:
     LD A,E              ; $0080  play sound
     LD ($FF00+C),A  ; $0081
     INC C               ; $0082
     LD A,$87            ; $0083
     LD ($FF00+C),A  ; $0085
Addr_0086:
     LD A,($FF00+$42); $0086
     SUB B               ; $0088
     LD ($FF00+$42),A; $0089  scroll logo up if B=1
     DEC D               ; $008b
     JR NZ, Addr_0060; $008c

     DEC B               ; $008e  set B=0 first time
     JR NZ, Addr_00E0; $008f    ... next time, cause jump to "Nintendo Logo check"

     LD D,$20            ; $0091  use scrolling loop to pause
     JR Addr_0060        ; $0093

     ; ==== Graphic routine ====

     LD C,A              ; $0095  "Double up" all the bits of the graphics data
     LD B,$04            ; $0096     and store in Video RAM
Addr_0098:
     PUSH BC             ; $0098
     RL C                ; $0099
     RLA                 ; $009b
     POP BC              ; $009c
     RL C                ; $009d
     RLA                 ; $009f
     DEC B               ; $00a0
     JR NZ, Addr_0098; $00a1
     LD (HL+),A          ; $00a3
     INC HL              ; $00a4
     LD (HL+),A          ; $00a5
     INC HL              ; $00a6
     RET                 ; $00a7
```



```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053

     ; === Scroll logo on screen, and play logo sound===

Addr_0055:
     LD H,A              ; $0055  Initialize scroll count, H=0
     LD A,$64            ; $0056
     LD D,A              ; $0058  set loop count, D=$64
     LD ($FF00+$42),A; $0059  Set vertical scroll register
     LD A,$91            ; $005b
     LD ($FF00+$40),A; $005d  Turn on LCD, showing Background
     INC B               ; $005f  Set B=1
Addr_0060:
     LD E,$02            ; $0060
Addr_0062:
     LD C,$0c            ; $0062
Addr_0064:
     LD A,($FF00+$44); $0064  wait for screen frame
     CP $90              ; $0066
     JR NZ, Addr_0064; $0068
     DEC C               ; $006a
     JR NZ, Addr_0064; $006b
     DEC E               ; $006d
     JR NZ, Addr_0062; $006e

     LD C,$13            ; $0070
     INC H               ; $0072  increment scroll count
     LD A,H              ; $0073
     LD E,$83            ; $0074
     CP $62              ; $0076  $62 counts in, play sound #1
     JR Z, Addr_0080 ; $0078
     LD E,$c1            ; $007a
     CP $64              ; $007c
     JR NZ, Addr_0086; $007e  $64 counts in, play sound #2
Addr_0080:
     LD A,E              ; $0080  play sound
     LD ($FF00+C),A  ; $0081
     INC C               ; $0082
     LD A,$87            ; $0083
     LD ($FF00+C),A  ; $0085
Addr_0086:
     LD A,($FF00+$42); $0086
     SUB B               ; $0088
     LD ($FF00+$42),A; $0089  scroll logo up if B=1
     DEC D               ; $008b
     JR NZ, Addr_0060; $008c

     DEC B               ; $008e  set B=0 first time
     JR NZ, Addr_00E0; $008f    ... next time, cause jump to "Nintendo Logo check"

     LD D,$20            ; $0091  use scrolling loop to pause
     JR Addr_0060        ; $0093

     ; ==== Graphic routine ====

     LD C,A              ; $0095  "Double up" all the bits of the graphics data
     LD B,$04            ; $0096     and store in Video RAM
Addr_0098:
     PUSH BC             ; $0098
     RL C                ; $0099
     RLA                 ; $009b
     POP BC              ; $009c
     RL C                ; $009d
     RLA                 ; $009f
     DEC B               ; $00a0
     JR NZ, Addr_0098; $00a1
     LD (HL+),A          ; $00a3
     INC HL              ; $00a4
     LD (HL+),A          ; $00a5
     INC HL              ; $00a6
     RET                 ; $00a7
```



```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053

     ; === Scroll logo on screen, and play logo sound===

Addr_0055:
     LD H,A              ; $0055  Initialize scroll count, H=0
     LD A,$64            ; $0056
     LD D,A              ; $0058  set loop count, D=$64
     LD ($FF00+$42),A; $0059  Set vertical scroll register
     LD A,$91            ; $005b
     LD ($FF00+$40),A; $005d  Turn on LCD, showing Background
     INC B               ; $005f  Set B=1
Addr_0060:
     LD E,$02            ; $0060
Addr_0062:
     LD C,$0c            ; $0062
Addr_0064:
     LD A,($FF00+$44); $0064  wait for screen frame
     CP $90              ; $0066
     JR NZ, Addr_0064; $0068
     DEC C               ; $006a
     JR NZ, Addr_0064; $006b
     DEC E               ; $006d
     JR NZ, Addr_0062; $006e

     LD C,$13            ; $0070
     INC H               ; $0072  increment scroll count
     LD A,H              ; $0073
     LD E,$83            ; $0074
     CP $62              ; $0076  $62 counts in, play sound #1
     JR Z, Addr_0080 ; $0078
     LD E,$c1            ; $007a
     CP $64              ; $007c
     JR NZ, Addr_0086; $007e  $64 counts in, play sound #2
Addr_0080:
     LD A,E              ; $0080  play sound
     LD ($FF00+C),A  ; $0081
     INC C               ; $0082
     LD A,$87            ; $0083
     LD ($FF00+C),A  ; $0085
Addr_0086:
     LD A,($FF00+$42); $0086
     SUB B               ; $0088
     LD ($FF00+$42),A; $0089  scroll logo up if B=1
     DEC D               ; $008b
     JR NZ, Addr_0060; $008c

     DEC B               ; $008e  set B=0 first time
     JR NZ, Addr_00E0; $008f    ... next time, cause jump to "Nintendo Logo check"

     LD D,$20            ; $0091  use scrolling loop to pause
     JR Addr_0060    ; $0093

     ; ==== Graphic routine ====

     LD C,A              ; $0095  "Double up" all the bits of the graphics data
     LD B,$04            ; $0096     and store in Video RAM
Addr_0098:
     PUSH BC             ; $0098
     RL C                ; $0099
     RLA                 ; $009b
     POP BC              ; $009c
     RL C                ; $009d
     RLA                 ; $009f
     DEC B               ; $00a0
     JR NZ, Addr_0098; $00a1
     LD (HL+),A          ; $00a3
     INC HL              ; $00a4
     LD (HL+),A          ; $00a5
     INC HL              ; $00a6
     RET                 ; $00a7

Addr_00A8:
     ;Nintendo Logo
     .DB $CE,$ED,$66,$66,$CC,$0D,$00,$0B,$03,$73,$00,$83,$00,$0C,$00,$0D
     .DB $00,$08,$11,$1F,$88,$89,$00,$0E,$DC,$CC,$6E,$E6,$DD,$DD,$D9,$99
     .DB $BB,$BB,$67,$63,$6E,$0E,$EC,$CC,$DD,$DC,$99,$9F,$BB,$B9,$33,$3E

Addr_00D8:
     ;More video data
     .DB $3C,$42,$B9,$A5,$B9,$A5,$42,$3C
```

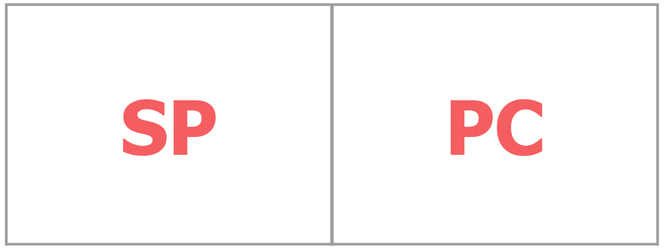
```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053

     ; === Scroll logo on screen, and play logo sound===

Addr_0055:
     LD H,A              ; $0055  Initialize scroll count, H=0
     LD A,$64            ; $0056
     LD D,A              ; $0058  set loop count, D=$64
     LD ($FF00+$42),A; $0059  Set vertical scroll register
     LD A,$91            ; $005b
     LD ($FF00+$40),A; $005d  Turn on LCD, showing Background
     INC B               ; $005f  Set B=1
Addr_0060:
     LD E,$02            ; $0060
Addr_0062:
     LD C,$0c            ; $0062
Addr_0064:
     LD A,($FF00+$44); $0064  wait for screen frame
     CP $90              ; $0066
     JR NZ, Addr_0064; $0068
     DEC C               ; $006a
     JR NZ, Addr_0064; $006b
     DEC E               ; $006d
     JR NZ, Addr_0062; $006e

     LD C,$13            ; $0070
     INC H               ; $0072  increment scroll count
     LD A,H              ; $0073
     LD E,$83            ; $0074
     CP $62              ; $0076  $62 counts in, play sound #1
     JR Z, Addr_0080 ; $0078
     LD E,$c1            ; $007a
     CP $64              ; $007c
     JR NZ, Addr_0086; $007e  $64 counts in, play sound #2
Addr_0080:
     LD A,E              ; $0080  play sound
     LD ($FF00+C),A  ; $0081
     INC C               ; $0082
     LD A,$87            ; $0083
     LD ($FF00+C),A  ; $0085
Addr_0086:
     LD A,($FF00+$42); $0086
     SUB B               ; $0088
     LD ($FF00+$42),A; $0089  scroll logo up if B=1
     DEC D               ; $008b
     JR NZ, Addr_0060; $008c

     DEC B               ; $008e  set B=0 first time
     JR NZ, Addr_00E0; $008f    ... next time, cause jump to "Nintendo Logo check"

     LD D,$20            ; $0091  use scrolling loop to pause
     JR Addr_0060    ; $0093

     ; ==== Graphic routine ====

     LD C,A              ; $0095  "Double up" all the bits of the graphics data
     LD B,$04            ; $0096     and store in Video RAM
Addr_0098:
     PUSH BC             ; $0098
     RL C                ; $0099
     RLA                 ; $009b
     POP BC              ; $009c
     RL C                ; $009d
     RLA                 ; $009f
     DEC B               ; $00a0
     JR NZ, Addr_0098; $00a1
     LD (HL+),A          ; $00a3
     INC HL              ; $00a4
     LD (HL+),A          ; $00a5
     INC HL              ; $00a6
     RET                 ; $00a7

Addr_00A8:
     ;Nintendo Logo
     .DB $CE,$ED,$66,$66,$CC,$0D,$00,$0B,$03,$73,$00,$83,$00,$0C,$00,$0D
     .DB $00,$08,$11,$1F,$88,$89,$00,$0E,$DC,$CC,$6E,$E6,$DD,$DD,$D9,$99
     .DB $BB,$BB,$67,$63,$6E,$0E,$EC,$CC,$DD,$DC,$99,$9F,$BB,$B9,$33,$3E

Addr_00D8:
     ;More video data
     .DB $3C,$42,$B9,$A5,$B9,$A5,$42,$3C

     ; ===== Nintendo logo comparison routine =====

Addr_00E0:
     LD HL,$0104         ; $00e0   ; point HL to Nintendo logo in cart
     LD DE,$00a8         ; $00e3   ; point DE to Nintendo logo in DMG rom

Addr_00E6:
     LD A,(DE)      ; $00e6
     INC DE         ; $00e7
     CP (HL)        ; $00e8   ;compare logo data in cart to DMG rom
     JR NZ,$fe      ; $00e9   ;if not a match, lock up here
     INC HL         ; $00eb
     LD A,L         ; $00ec
     CP $34         ; $00ed   ;do this for $30 bytes
     JR NZ, Addr_00E6    ; $00ef

     LD B,$19       ; $00f1
     LD A,B         ; $00f3
Addr_00F4:
     ADD (HL)       ; $00f4
     INC HL         ; $00f5
     DEC B               ; $00f6
     JR NZ, Addr_00F4    ; $00f7
     ADD (HL)       ; $00f9
     JR NZ,$fe      ; $00fa   ; if $19 + bytes from $0134-$014D  don't add to $00
                              ;  ... lock up

     LD A,$01       ; $00fc
     LD ($FF00+$50),A    ; $00fe   ;turn off DMG rom
```

```
     LD SP,$fffe         ; $0000  Setup Stack

     XOR A               ; $0003  Zero the memory from $8000-$9FFF (VRAM)
     LD HL,$9fff         ; $0004
Addr_0007:
     LD (HL-),A          ; $0007
     BIT 7,H        ; $0008
     JR NZ, Addr_0007    ; $000a

     LD HL,$ff26         ; $000c  Setup Audio
     LD C,$11       ; $000f
     LD A,$80       ; $0011
     LD (HL-),A          ; $0013
     LD ($FF00+C),A ; $0014
     INC C               ; $0015
     LD A,$f3       ; $0016
     LD ($FF00+C),A ; $0018
     LD (HL-),A          ; $0019
     LD A,$77       ; $001a
     LD (HL),A      ; $001c

     LD A,$fc       ; $001d  Setup BG palette
     LD ($FF00+$47),A    ; $001f

     LD DE,$0104         ; $0021  Convert and load logo data from cart into Video
RAM
     LD HL,$8010         ; $0024
Addr_0027:
     LD A,(DE)      ; $0027
     CALL $0095          ; $0028
     CALL $0096          ; $002b
     INC DE         ; $002e
     LD A,E         ; $002f
     CP $34         ; $0030
     JR NZ, Addr_0027    ; $0032

     LD DE,$00d8         ; $0034  Load 8 additional bytes into Video RAM
     LD B,$08       ; $0037
Addr_0039:
     LD A,(DE)      ; $0039
     INC DE         ; $003a
     LD (HL+),A          ; $003b
     INC HL         ; $003c
     DEC B               ; $003d
     JR NZ, Addr_0039    ; $003e

     LD A,$19       ; $0040  Setup background tilemap
     LD ($9910),A   ; $0042
     LD HL,$992f         ; $0045
Addr_0048:
     LD C,$0c       ; $0048
Addr_004A:
     DEC A               ; $004a
     JR Z, Addr_0055     ; $004b
     LD (HL-),A          ; $004d
     DEC C               ; $004e
     JR NZ, Addr_004A    ; $004f
     LD L,$0f       ; $0051
     JR Addr_0048   ; $0053
```

```
     ; === Scroll logo on screen, and play logo sound===

Addr_0055:
     LD H,A         ; $0055  Initialize scroll count, H=0
     LD A,$64       ; $0056
     LD D,A         ; $0058  set loop count, D=$64
     LD ($FF00+$42),A; $0059  Set vertical scroll register
     LD A,$91       ; $005b
     LD ($FF00+$40),A; $005d  Turn on LCD, showing Background
     INC B          ; $005f  Set B=1
Addr_0060:
     LD E,$02       ; $0060
Addr_0062:
     LD C,$0c       ; $0062
Addr_0064:
     LD A,($FF00+$44); $0064  wait for screen frame
     CP $90         ; $0066
     JR NZ, Addr_0064; $0068
     DEC C          ; $006a
     JR NZ, Addr_0064; $006b
     DEC E          ; $006d
     JR NZ, Addr_0062; $006e

     LD C,$13       ; $0070
     INC H          ; $0072  increment scroll count
     LD A,H         ; $0073
     LD E,$83       ; $0074
     CP $62         ; $0076  $62 counts in, play sound #1
     JR Z, Addr_0080 ; $0078
     LD E,$c1       ; $007a
     CP $64         ; $007c
     JR NZ, Addr_0086; $007e  $64 counts in, play sound #2
Addr_0080:
     LD A,E         ; $0080  play sound
     LD ($FF00+C),A  ; $0081
     INC C          ; $0082
     LD A,$87       ; $0083
     LD ($FF00+C),A  ; $0085
Addr_0086:
     LD A,($FF00+$42); $0086
     SUB B          ; $0088
     LD ($FF00+$42),A; $0089  scroll logo up if B=1
     DEC D          ; $008b
     JR NZ, Addr_0060; $008c

     DEC B          ; $008e  set B=0 first time
     JR NZ, Addr_00E0; $008f    ... next time, cause jump to "Nintendo Logo check"

     LD D,$20       ; $0091  use scrolling loop to pause
     JR Addr_0060   ; $0093

     ; ==== Graphic routine ====

     LD C,A         ; $0095  "Double up" all the bits of the graphics data
     LD B,$04       ; $0096     and store in Video RAM
Addr_0098:
     PUSH BC        ; $0098
     RL C           ; $0099
     RLA            ; $009b
     POP BC         ; $009c
     RL C           ; $009d
     RLA            ; $009f
     DEC B          ; $00a0
     JR NZ, Addr_0098; $00a1
     LD (HL+),A     ; $00a3
     INC HL         ; $00a4
     LD (HL+),A     ; $00a5
     INC HL         ; $00a6
     RET            ; $00a7
```

```
Addr_00A8:
     ;Nintendo Logo
     .DB $CE,$ED,$66,$66,$CC,$0D,$00,$0B,$03,$73,$00,$83,$00,$0C,$00,$0D
     .DB $00,$08,$11,$1F,$88,$89,$00,$0E,$DC,$CC,$6E,$E6,$DD,$DD,$D9,$99
     .DB $BB,$BB,$67,$63,$6E,$0E,$EC,$CC,$DD,$DC,$99,$9F,$BB,$B9,$33,$3E

Addr_00D8:
     ;More video data
     .DB $3C,$42,$B9,$A5,$B9,$A5,$42,$3C

     ; ===== Nintendo logo comparison routine =====

Addr_00E0:
     LD HL,$0104         ; $00e0   ; point HL to Nintendo logo in cart
     LD DE,$00a8         ; $00e3   ; point DE to Nintendo logo in DMG rom

Addr_00E6:
     LD A,(DE)      ; $00e6
     INC DE         ; $00e7
     CP (HL)        ; $00e8   ;compare logo data in cart to DMG rom
     JR NZ,$fe      ; $00e9   ;if not a match, lock up here
     INC HL         ; $00eb
     LD A,L         ; $00ec
     CP $34         ; $00ed   ;do this for $30 bytes
     JR NZ, Addr_00E6    ; $00ef

     LD B,$19       ; $00f1
     LD A,B         ; $00f3
Addr_00F4:
     ADD (HL)       ; $00f4
     INC HL         ; $00f5
     DEC B               ; $00f6
     JR NZ, Addr_00F4    ; $00f7
     ADD (HL)       ; $00f9
     JR NZ,$fe      ; $00fa   ; if $19 + bytes from $0134-$014D  don't add to $00
                              ;  ... lock up

     LD A,$01       ; $00fc
     LD ($FF00+$50),A    ; $00fe   ;turn off DMG rom
```

Special Registers



Special Registers

SP

PC

Special Registers

[S]tack [Pointer]

16 bits

0xFFFF

Stores Register Data

Special Registers

[S]tack [Pointer]

16 bits

0xFFFF

Stores Register Data

[P]rogram [C]ounter

16 bits

0xFFFF

Next Instruction

PC

16 bits

0xFFFF

Next Instruction

Popular Game **ROMs**

Popular Game **ROMs**

Pokemon Blue

1024Kb

Popular Game **ROMs**

Pokemon Blue

1024Kb

Tetris

32Kb

Popular Game **ROMs**

Pokemon Blue

1024Kb

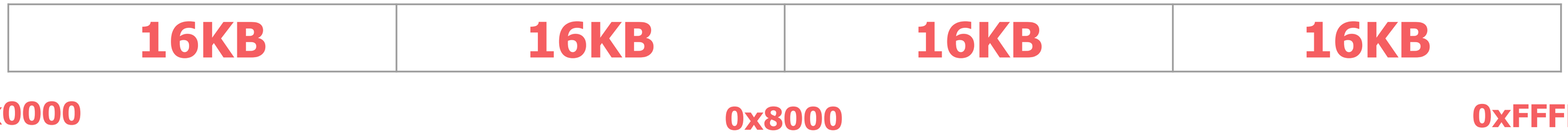
Tetris

32Kb

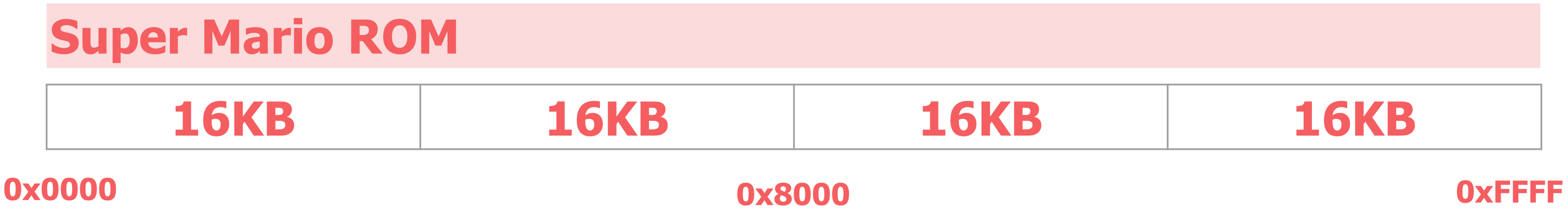
Super Mario

64Kb

Memory **Access**



Memory **Access**



Memory **Access**

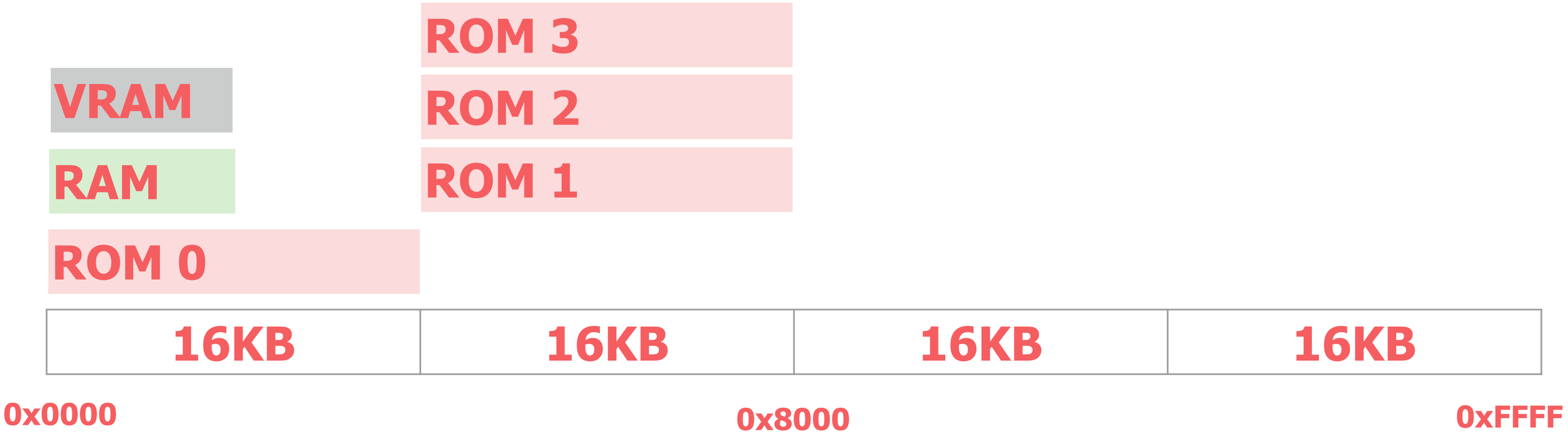
VRAM

RAM

Super Mario ROM

0x0000 **0x8000** **0xFFFF**

Memory **Access**




Nintendo
DMG-APED-NOE MADE IN JAPAN
DMG-APED-0
LH538KUU
JAPAN F1
LH52256CN-10LL
SHARP


U4 MM1134A Batt CR1616 DMG -A02-UT
U3 256K-BRAM
MBC-5
VCC
Nintendo GAME BOY
Nintendo
MADE IN JAPAN


Batt CR1616 DMG -DECN -02
95-03
DMG MBC1B1 Nintendo
3 PRG U1
VDD
GND
Nintendo GAME BOY
DONKEY KONG LAND
DMG-YT-FAH
MADE IN JAPAN
FABRIQUE AU JAPON
THIS SIDE OUT / CE COTE A L'EXTERIEUR

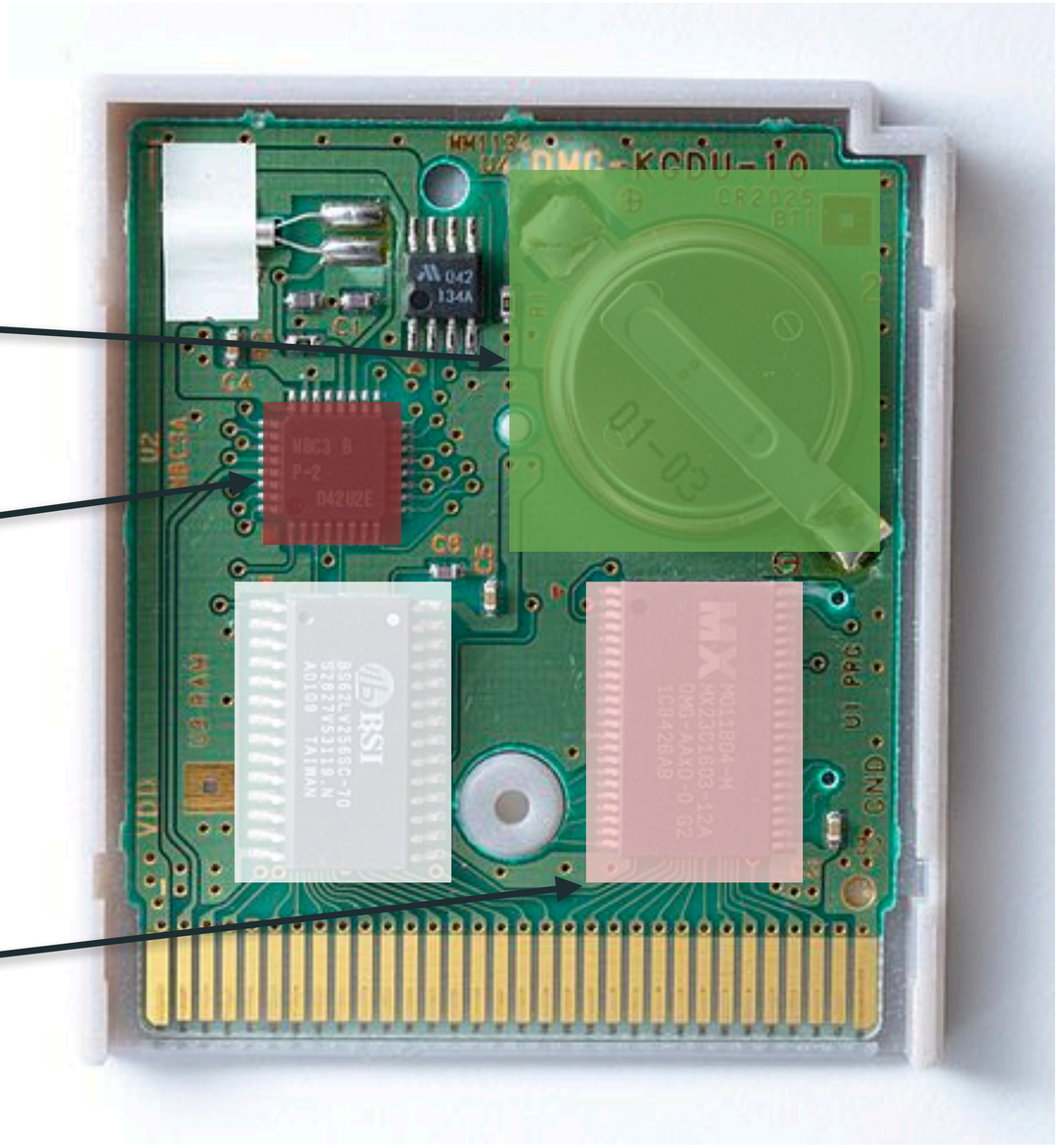

DMG-AQ7-01
U1 · 2M/4M/8M · ROM
U2
MBC5
C1
C2
C3
C4
C5
VDD
GND
DMG-AKQP-0
LH5308NG
JAPAN F1
9844 D
MBC5
P-1
840U6L
Nintendo GAME BOY
DMG-AKQP-EUR
MADE IN JAPAN
FABRIQUÉ AU JAPON
THIS SIDE OUT / CE CÔTÉ À L'EXTÉRIEUR
HERAUSGEGEBEN SEITE
Nintendo


C2
10µ
C1
103
DMG-
AAA-
03
U1 PRG
DMG-TRA-1
SHARP
JAPAN A0
9116 D
GND
Nintendo GAME BOY
TETRIS
DMG-TR-FAH
MADE IN JAPAN
FAIT AU JAPON
THIS SIDE OUT / CE CÔTÉ À L'EXTÉRIEUR
Nintendo

On-board Battery

MBC

ROM



Game Boy Cartridge Repair

Repair guides and support for read-only memory cartridges used in Game Boy, Game Boy Advance, and other handheld nintendo consoles.

Create a Guide
I Have This

Replacement Guides

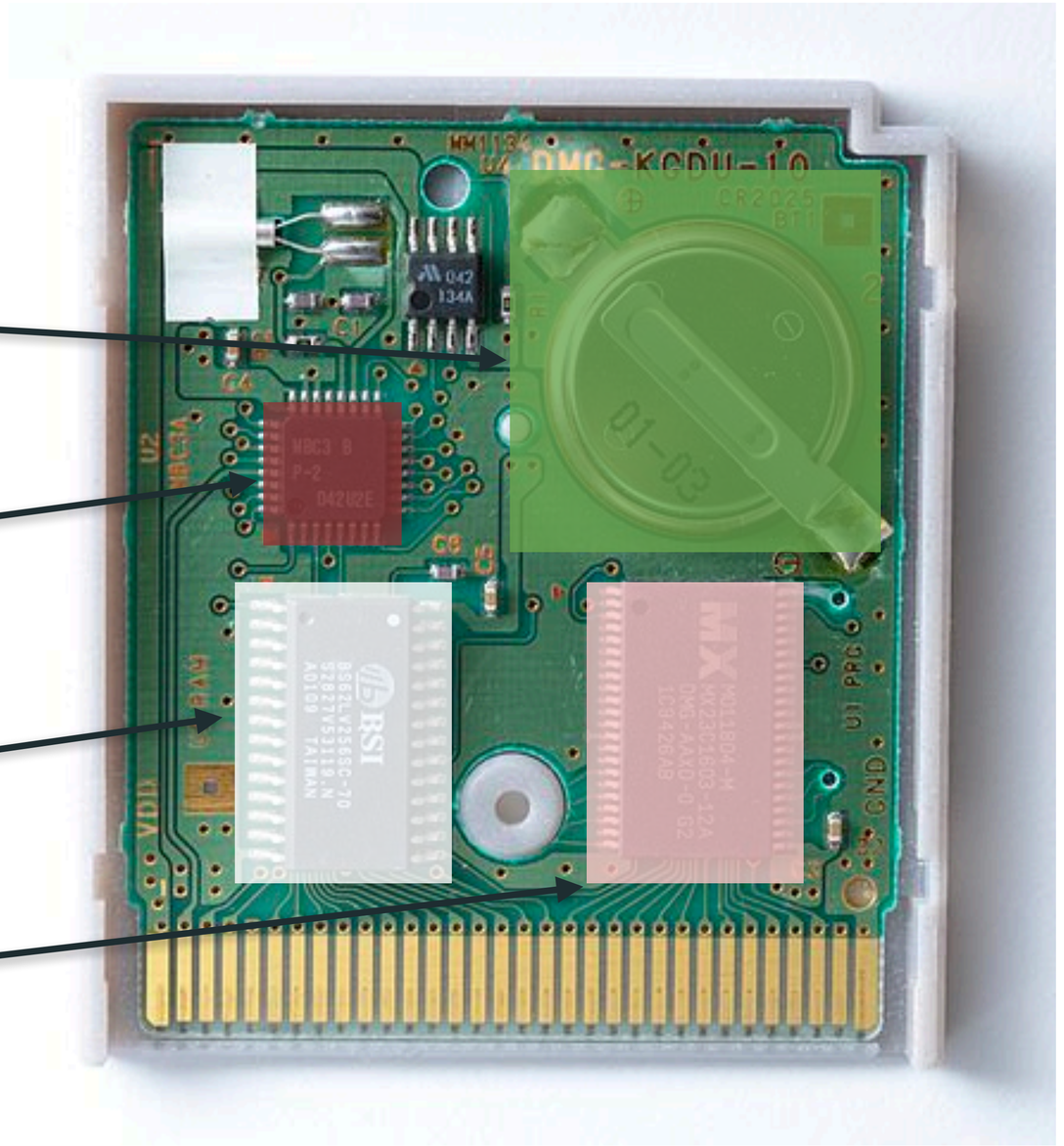
Battery

On-board Battery

MBC

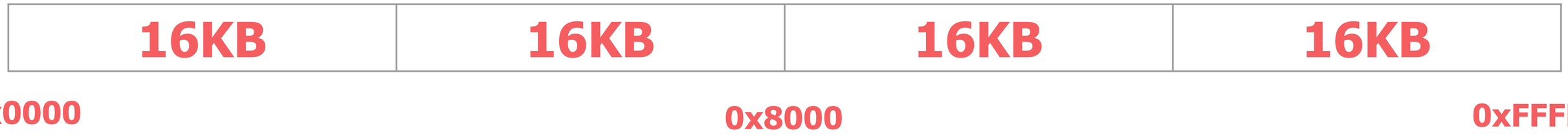
External RAM

ROM

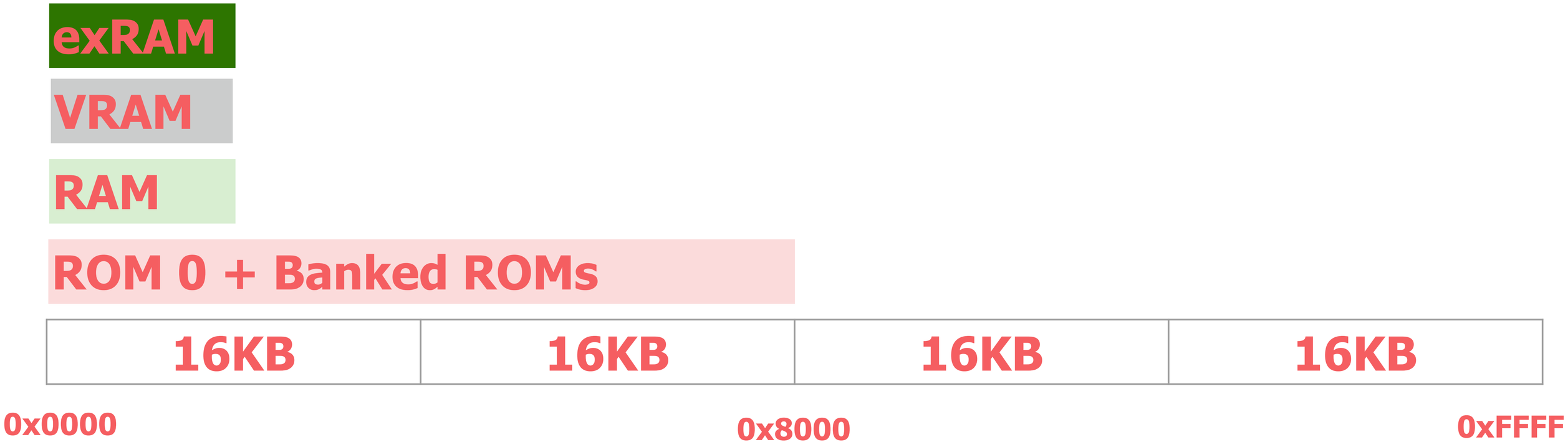


Memory **Access**

ROM 0 + Banked ROMs

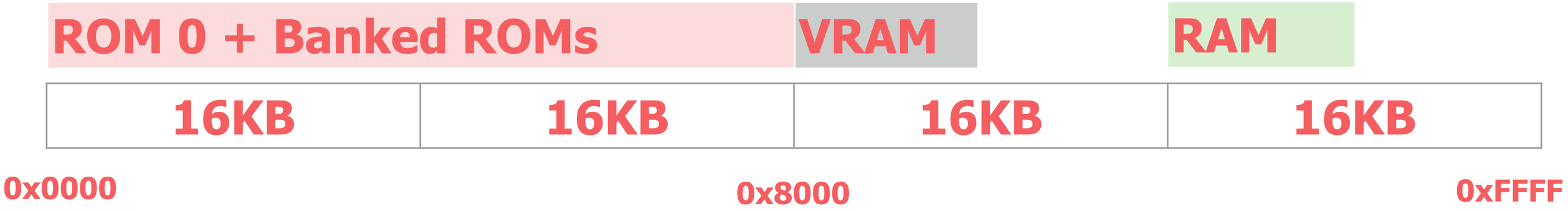


Memory **Access**

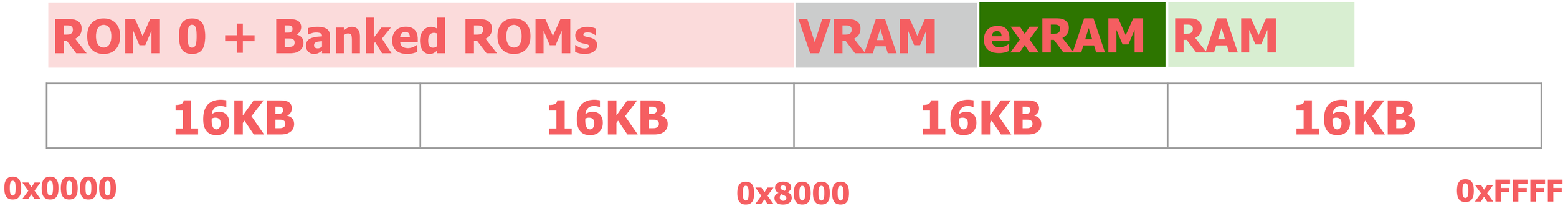


Memory **Access**

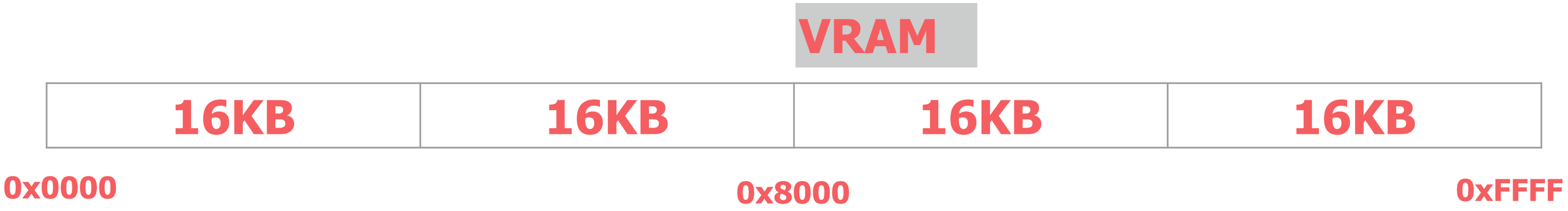
exRAM



Memory **Access**



Pixel Processing Unit (PPU)



LCD

160 x 144 pixel

4 shades of green

Tile-based rendering

0x8000 - 0x9FFF (8kB)

LCD

160 x 144 pixel

4 shades of green

Tile-based rendering

0x8000 - 0x9FFF (8kB)

Pixels

0x62

VRAM

Pixels

0b01100010

VRAM

Pixels

0b01100010

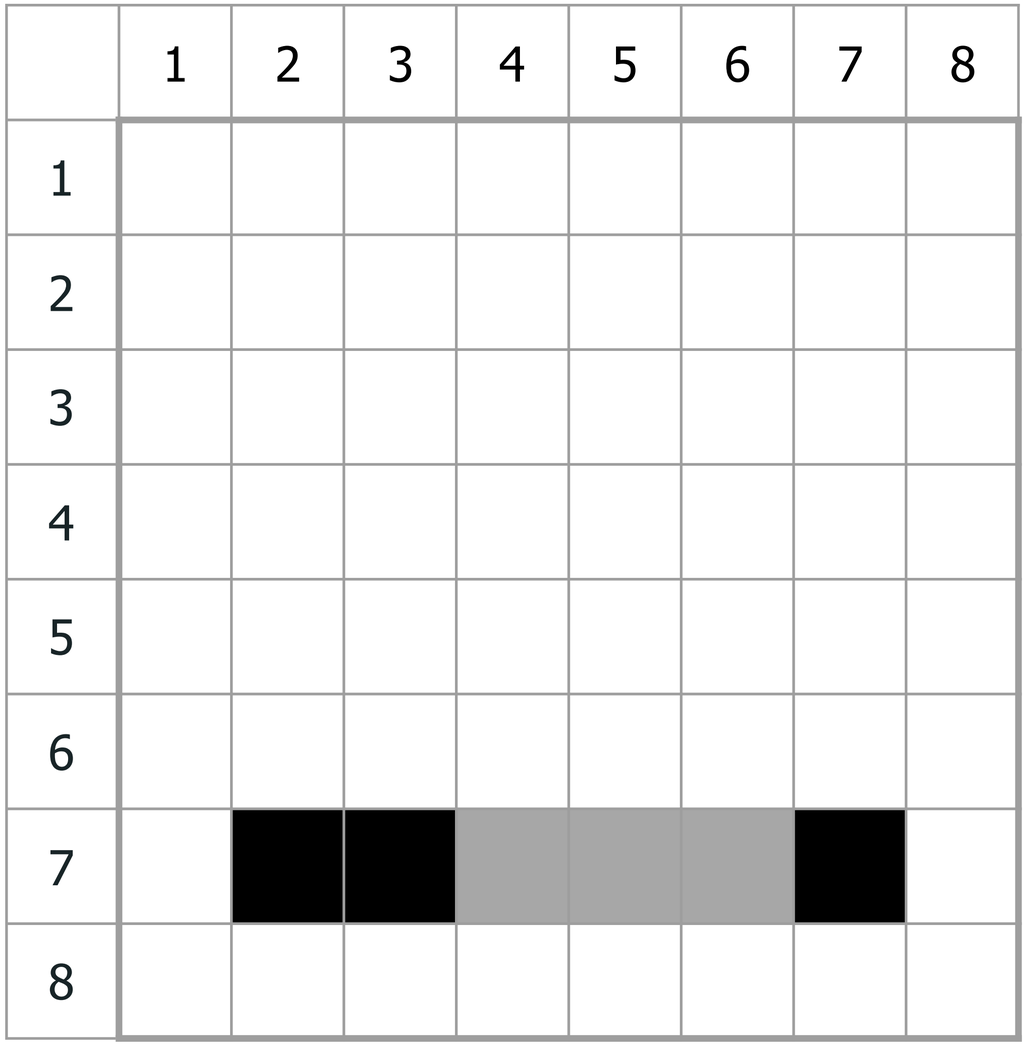
VRAM

Pixels

0b01100010
0b01111110

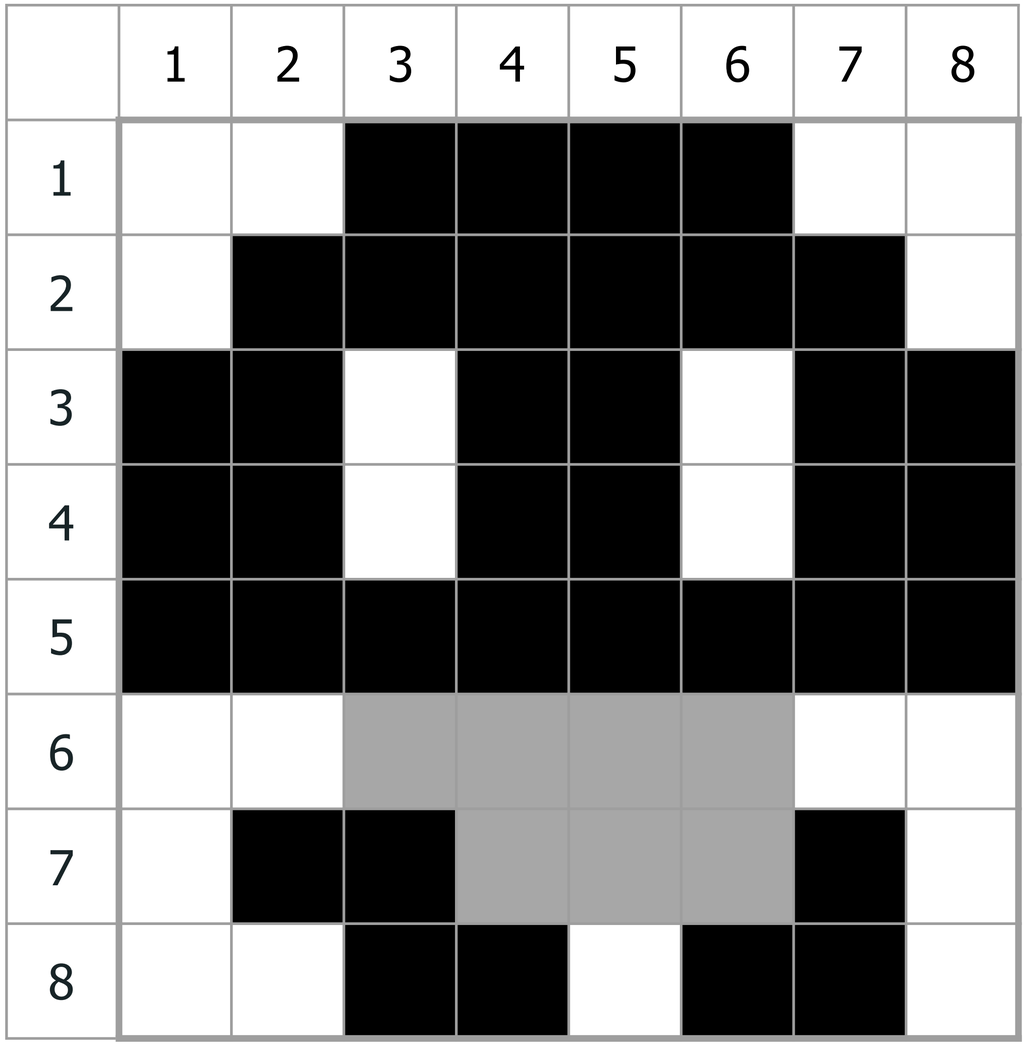
VRAM

Pixels



VRAM

Pixels



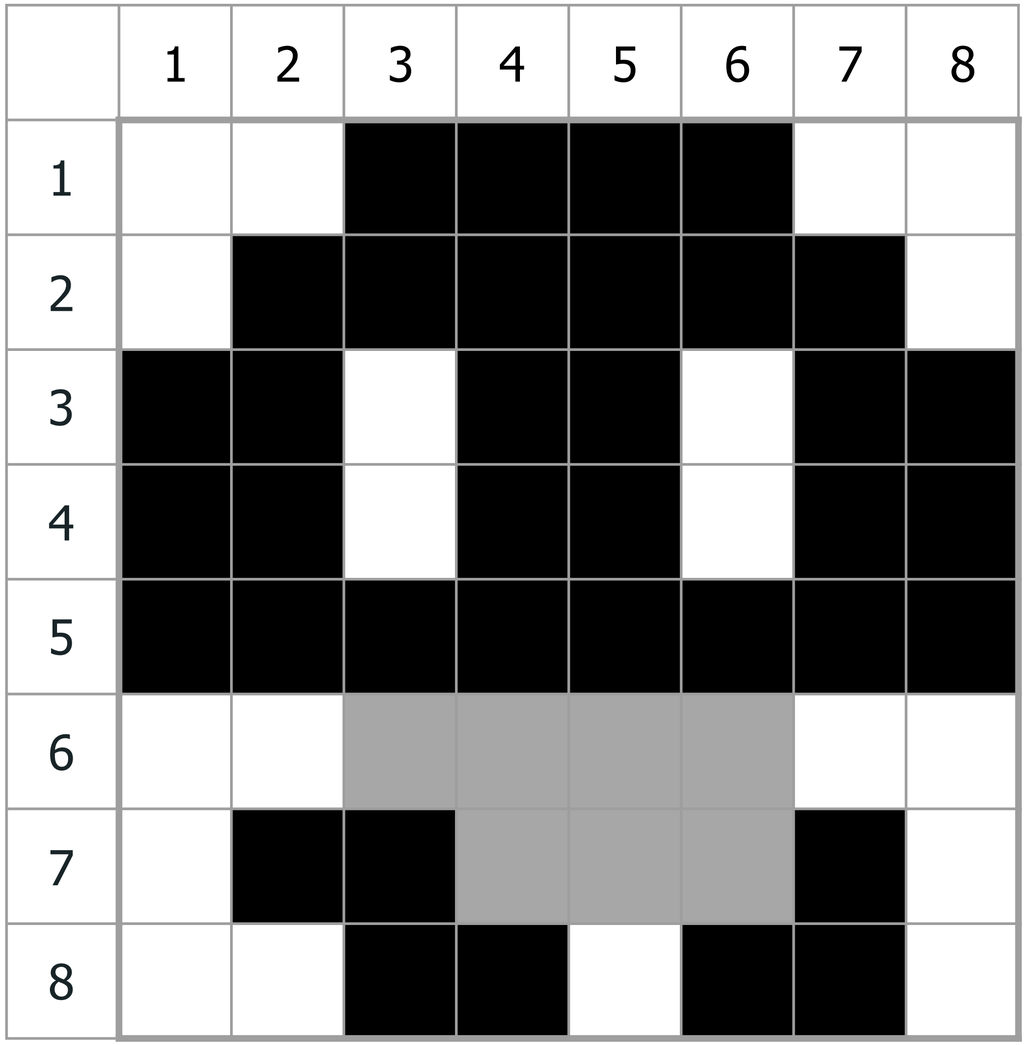
VRAM

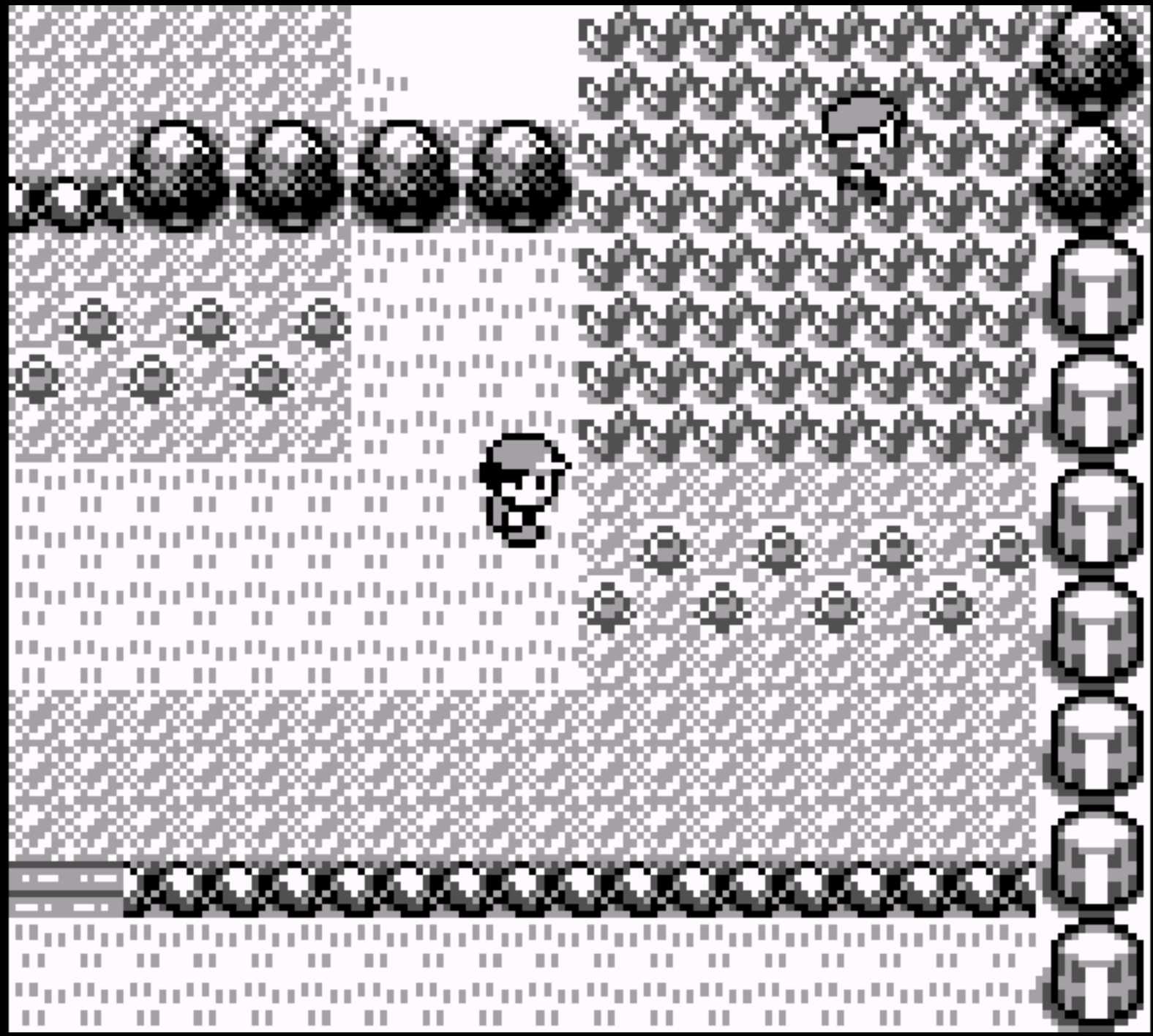
Pixels

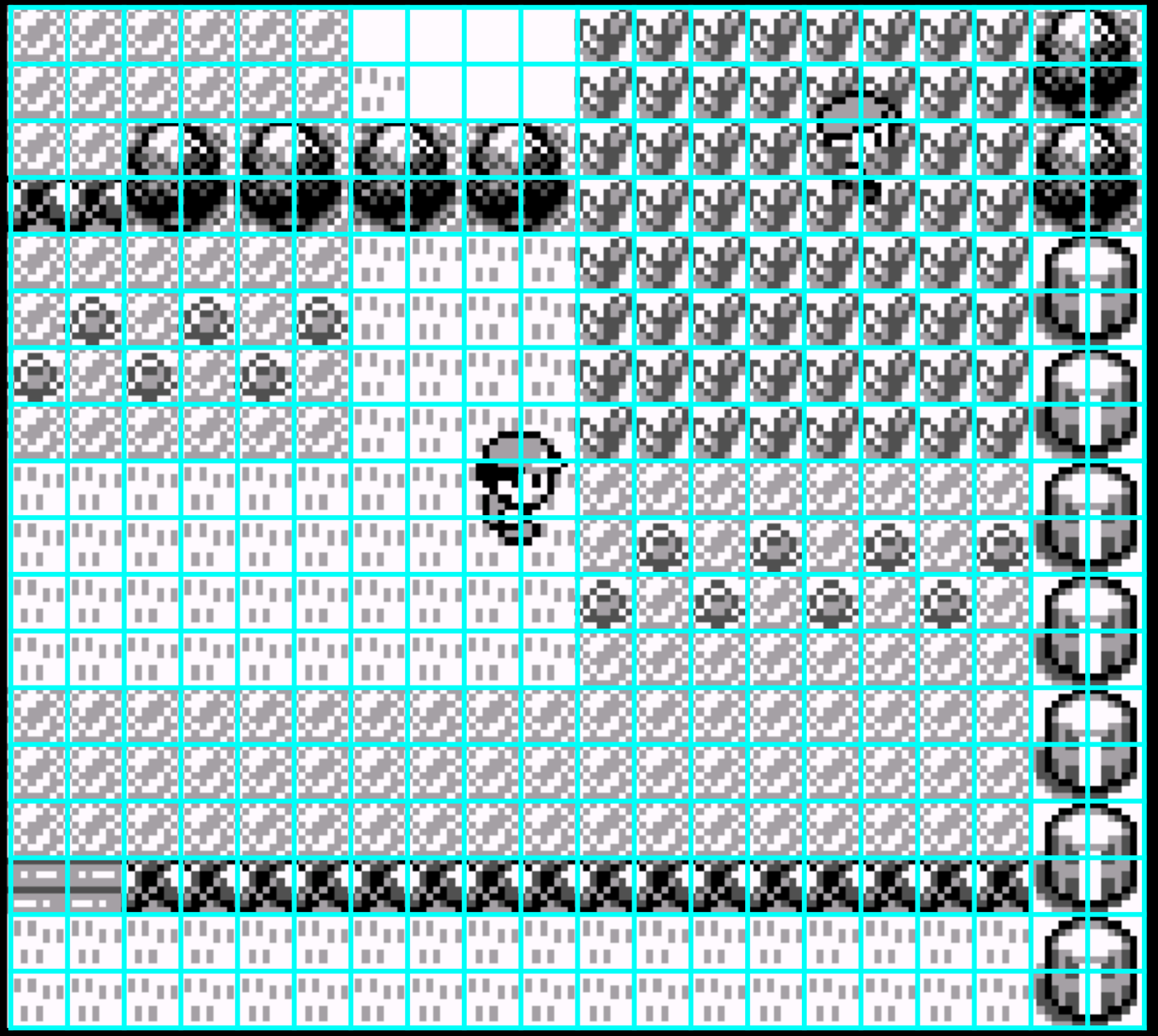
160 x 144 pixel

20 x 18 tiles

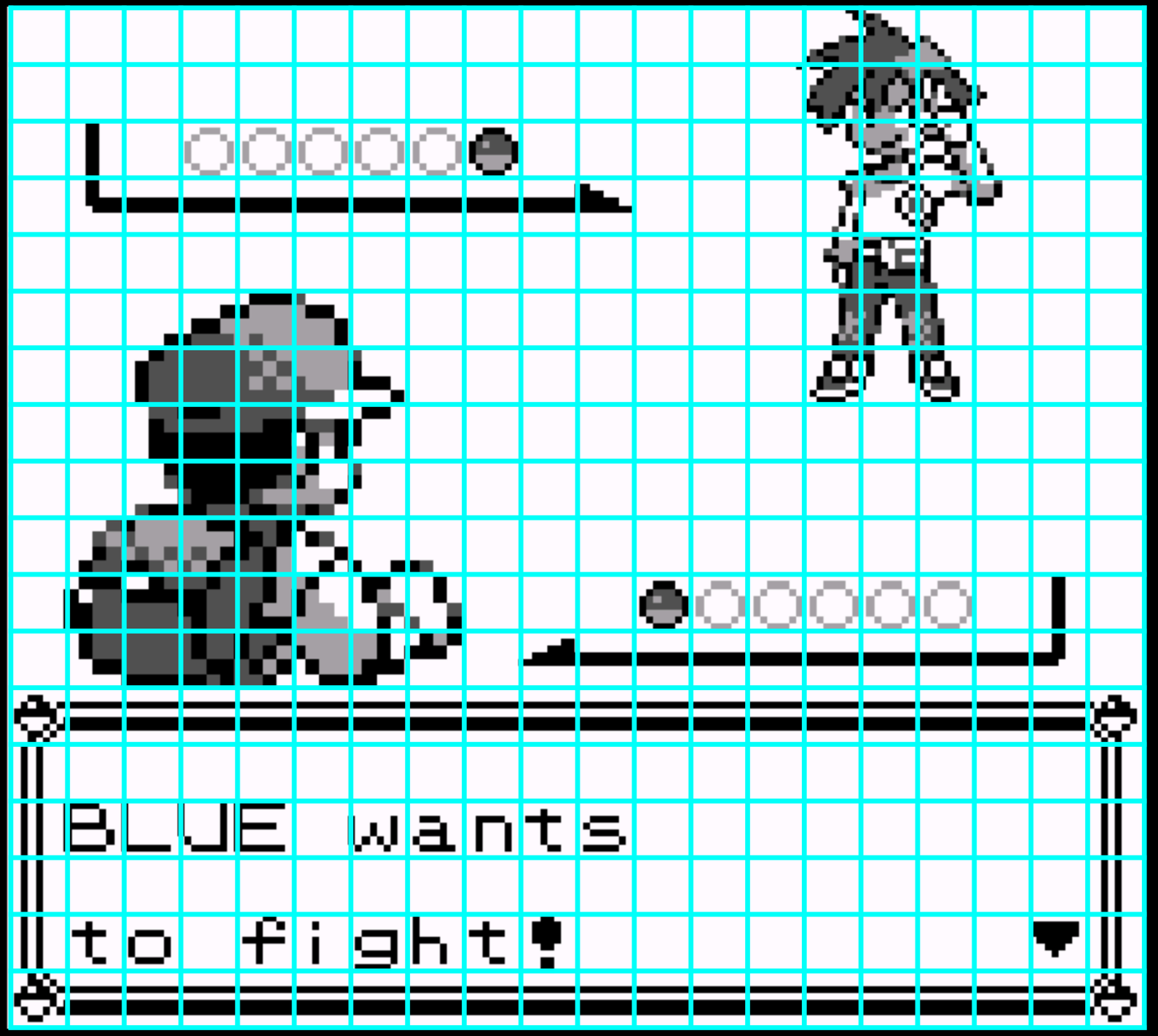
VRAM



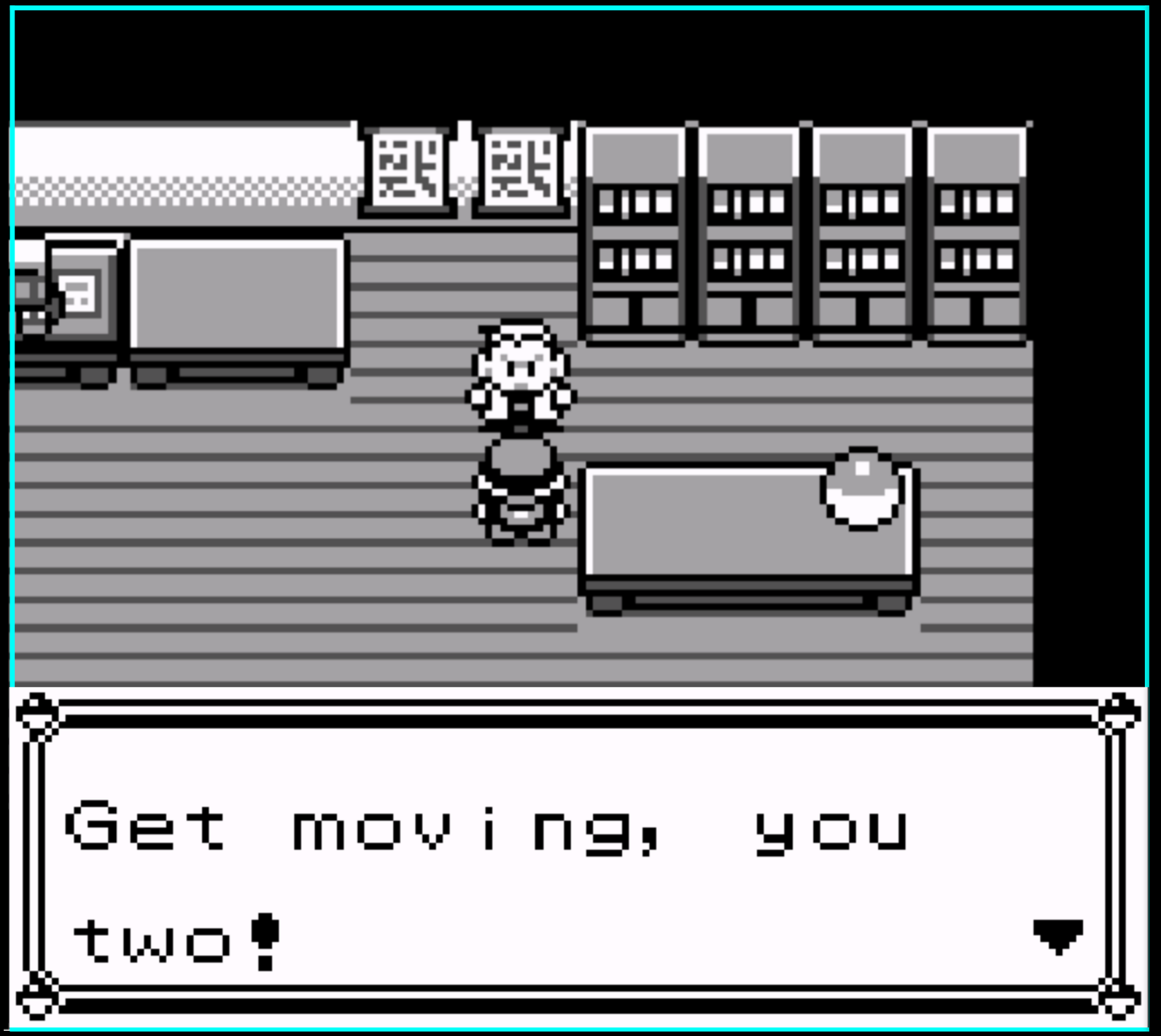




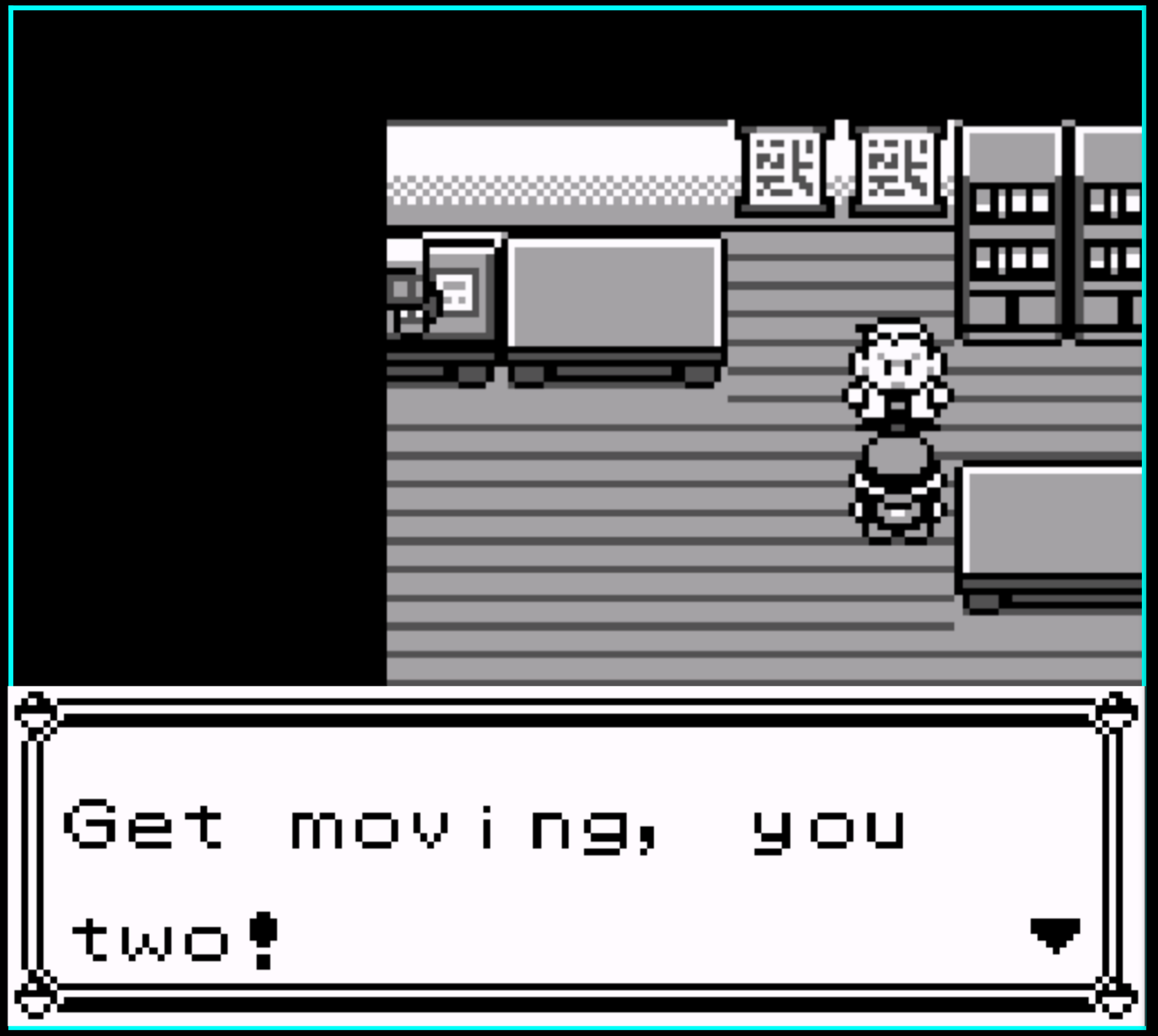

BLUE wants
to fight!

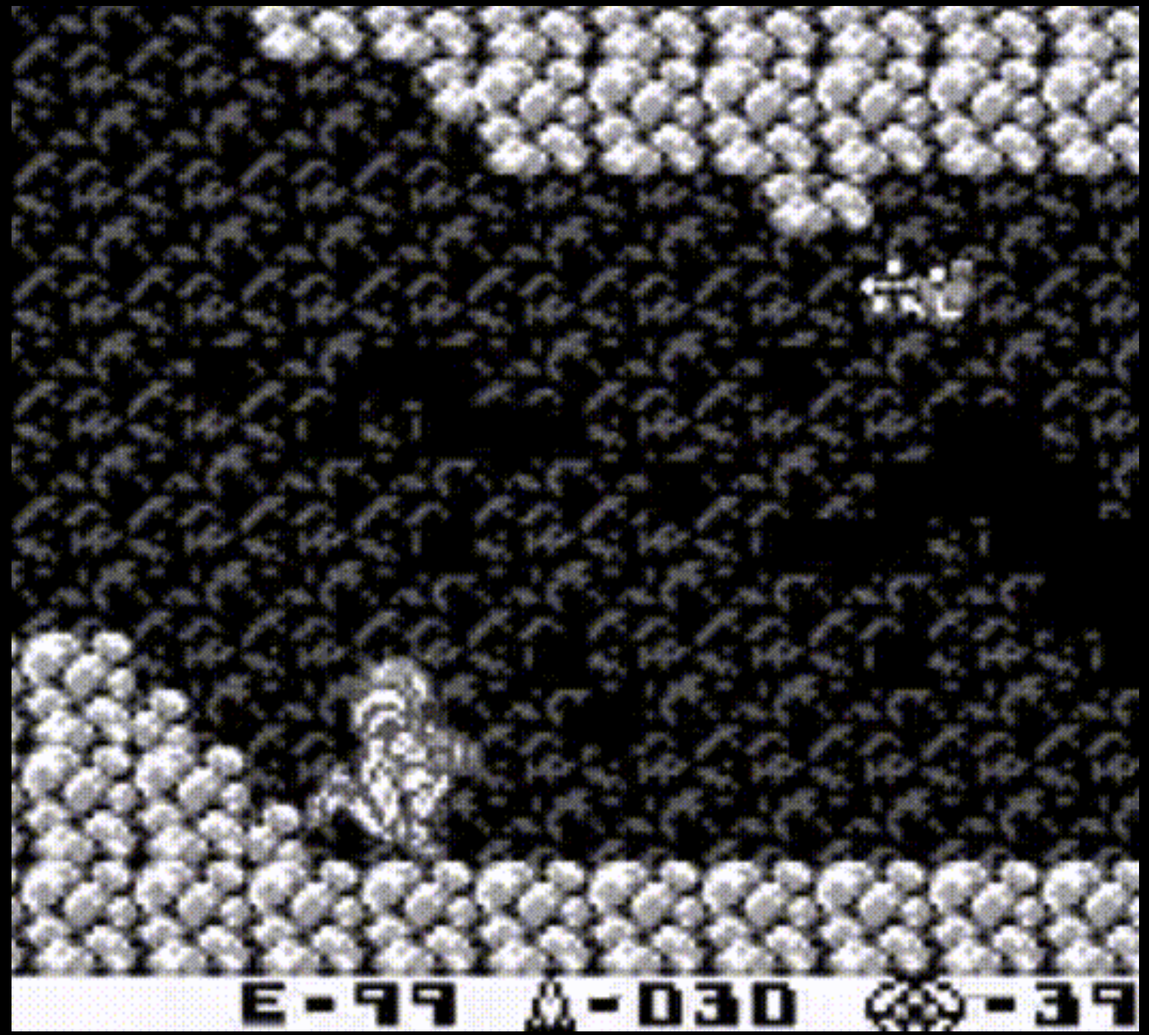

BLUE wants
to fight!

```
.screen {
   overflow: ‘hidden’;
}
```

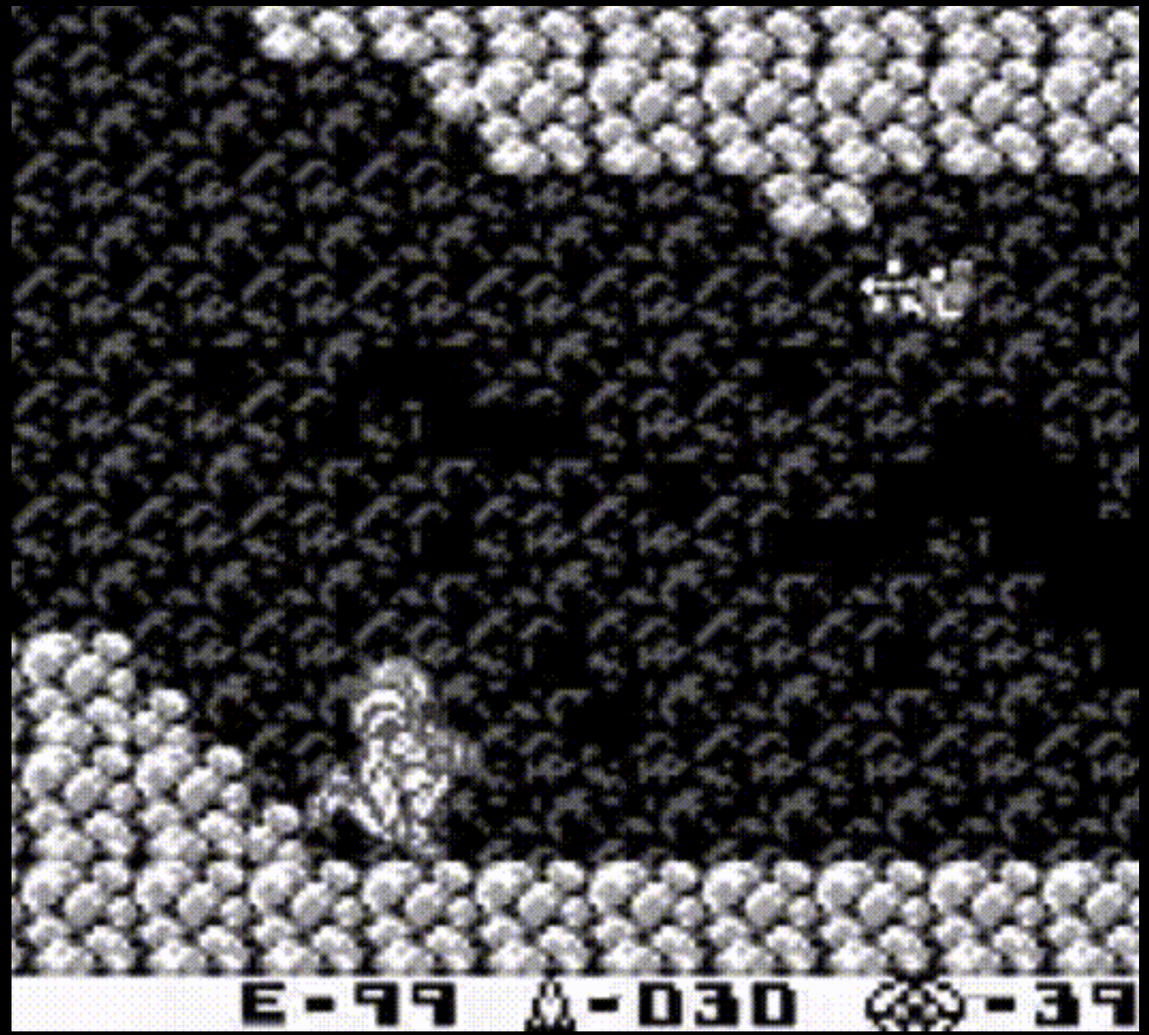


```
.screen {
   overflow: ‘hidden’;
   left: 60;
}
```



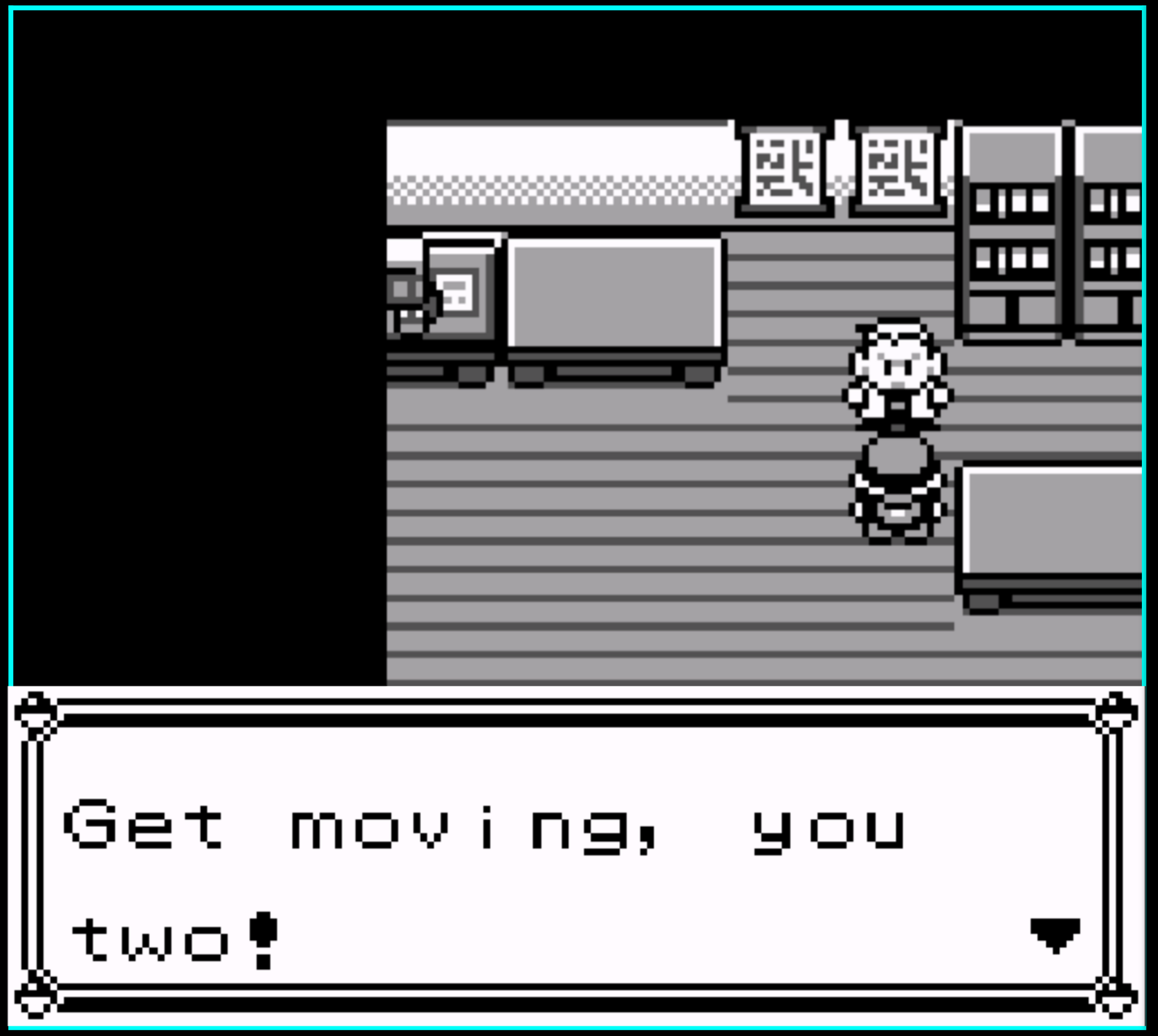

E-99
-030
-39


MARIO×02
0
×00
WORLD
1-1
TIME
400

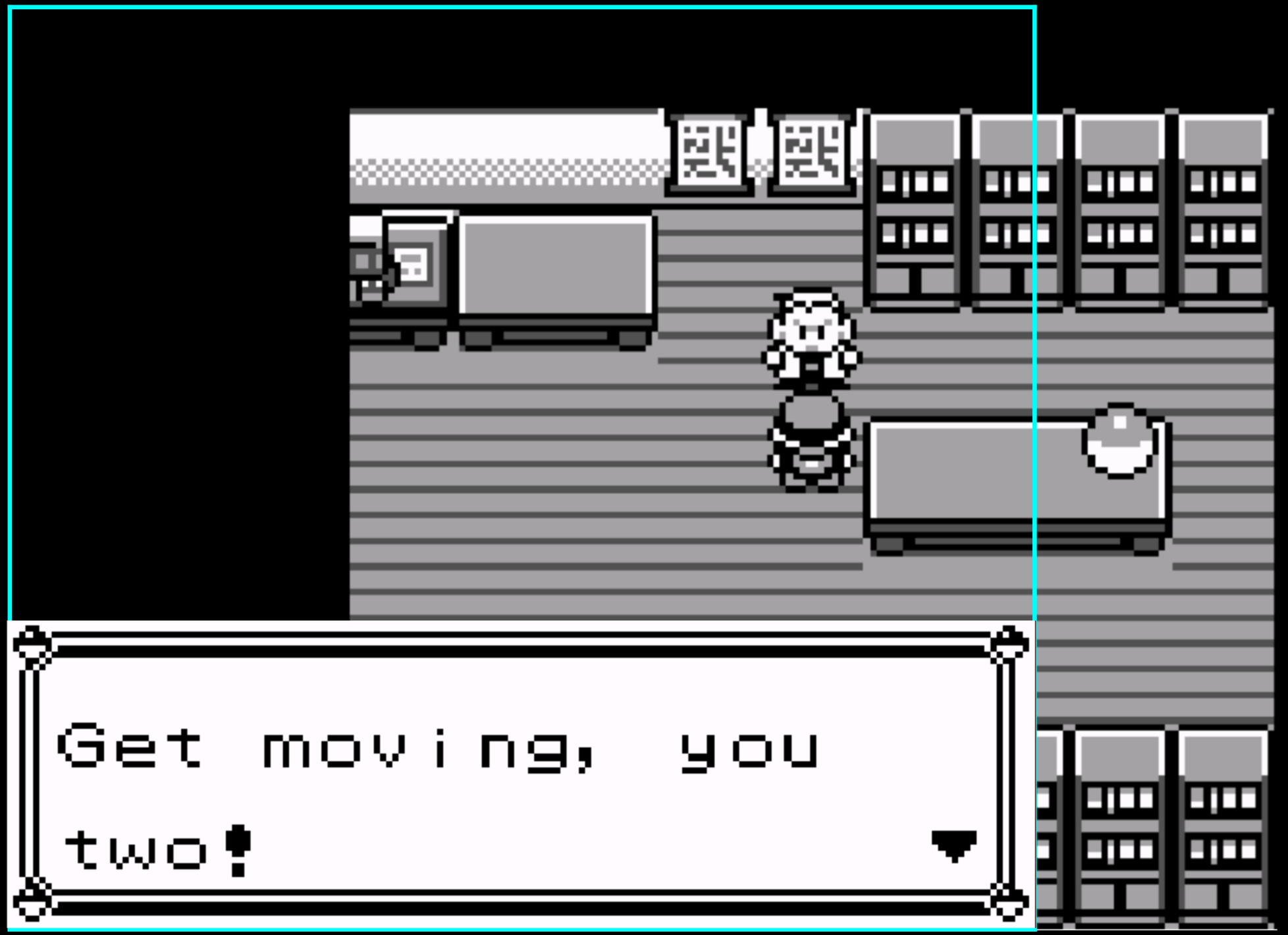

E-99
-030
-39


MARIO×02
0
×00
WORLD
1-1
TIME
400

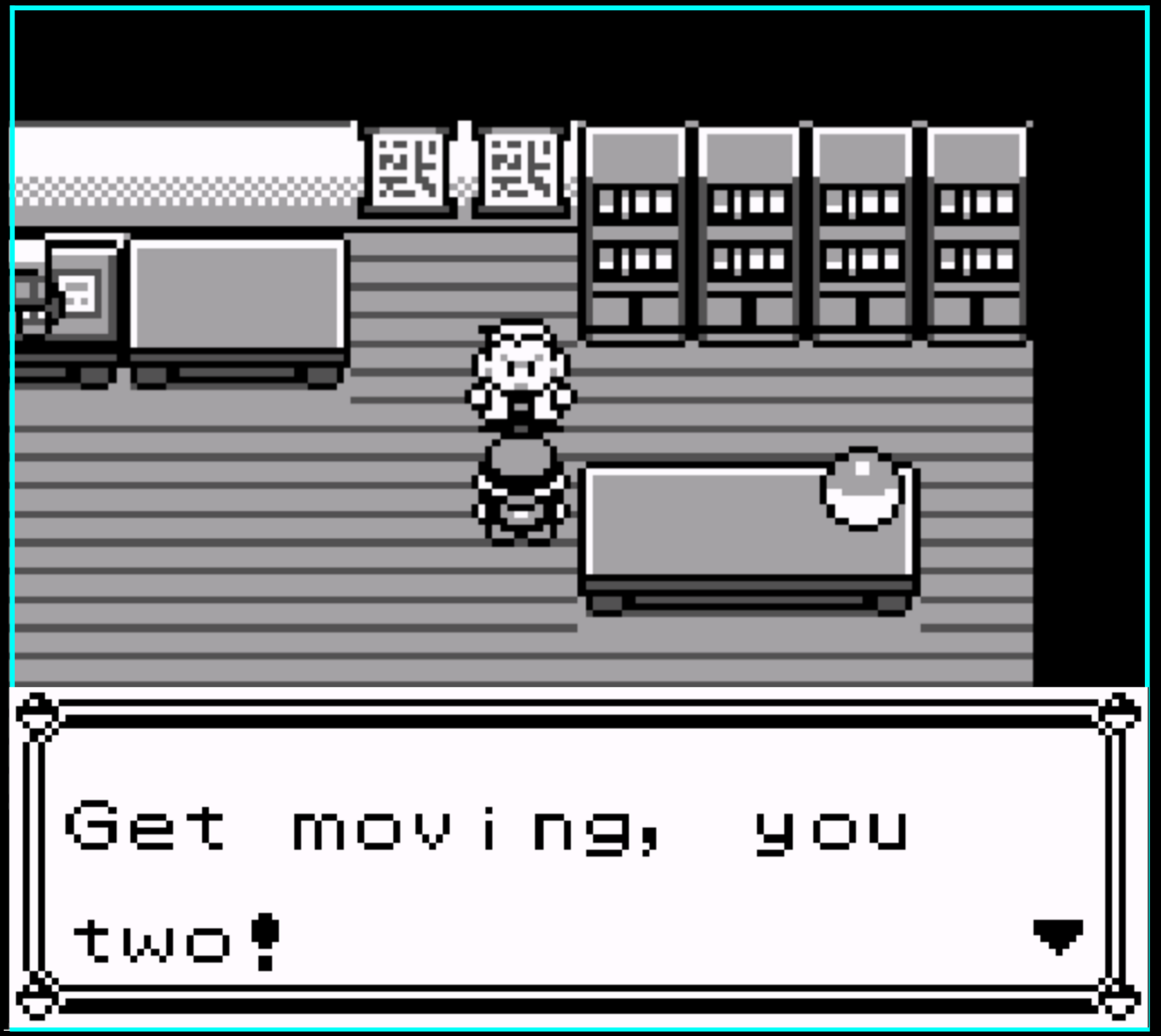
```
.screen {
   overflow: ‘hidden’;
   left: 60;
}
```



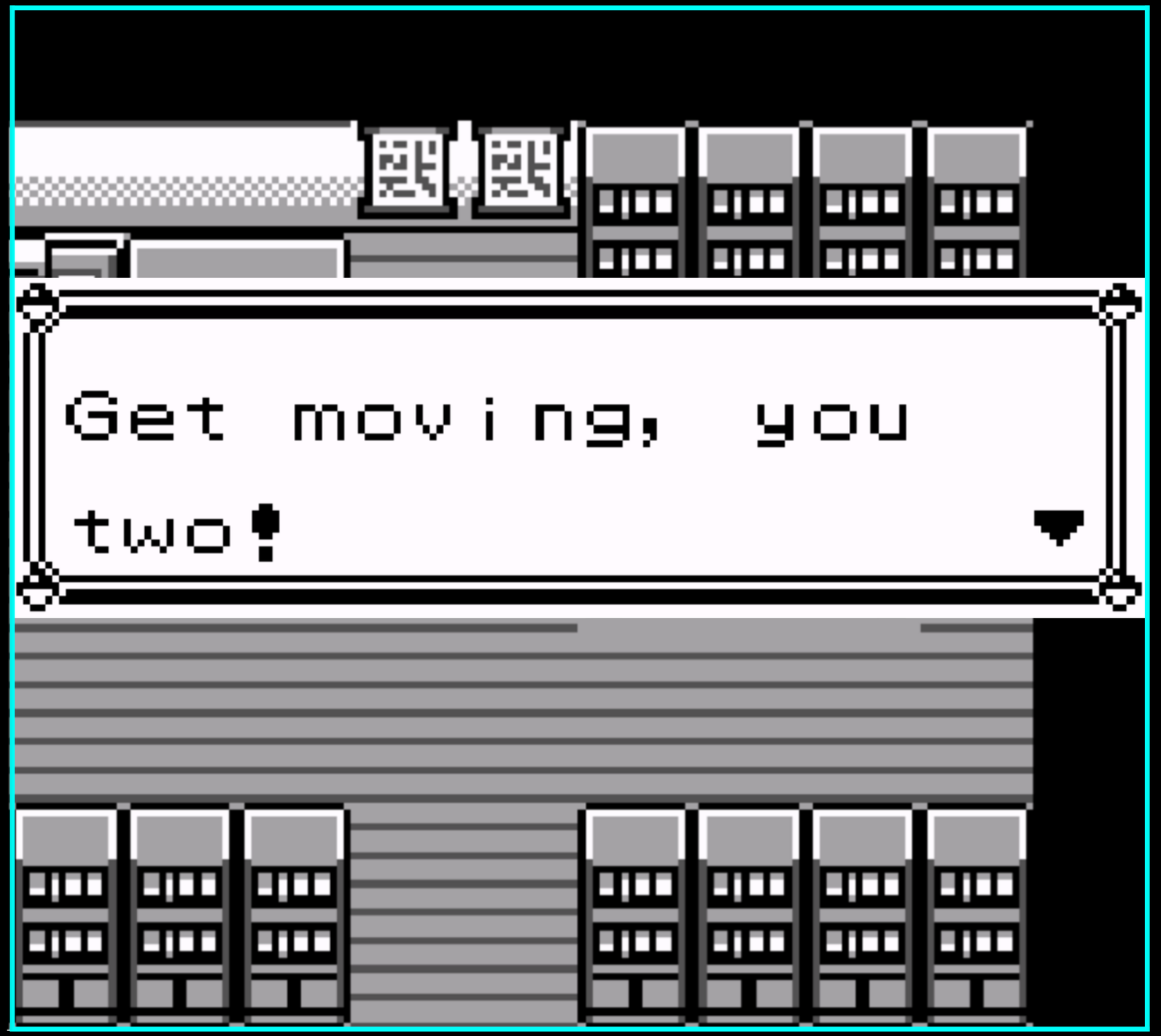
```
.screen {
   overflow: ‘hidden’;
   left: 60;
}
```



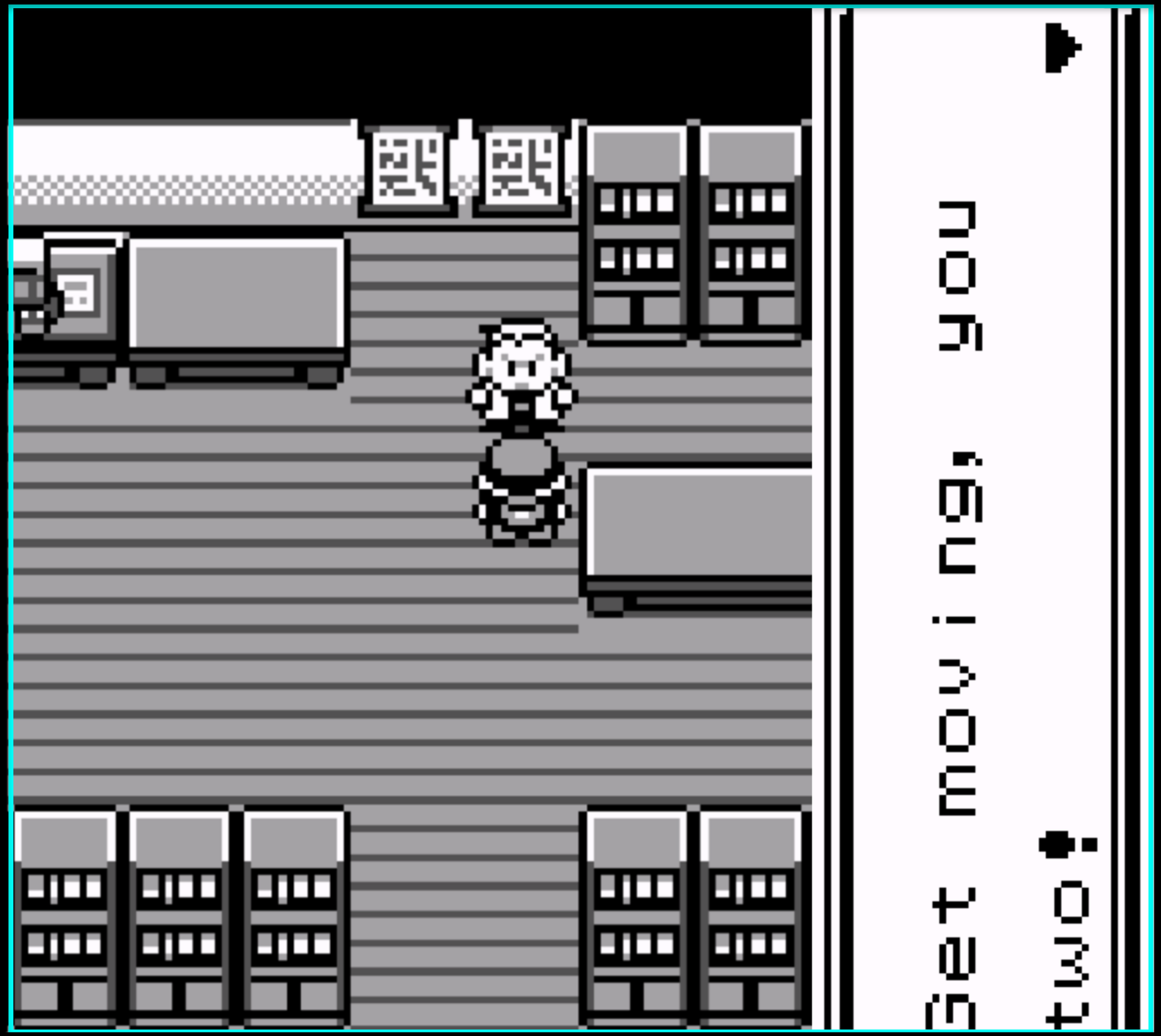
```
.fixed-window {
  position: ‘absolute’;
  z-index: 10;
}
```

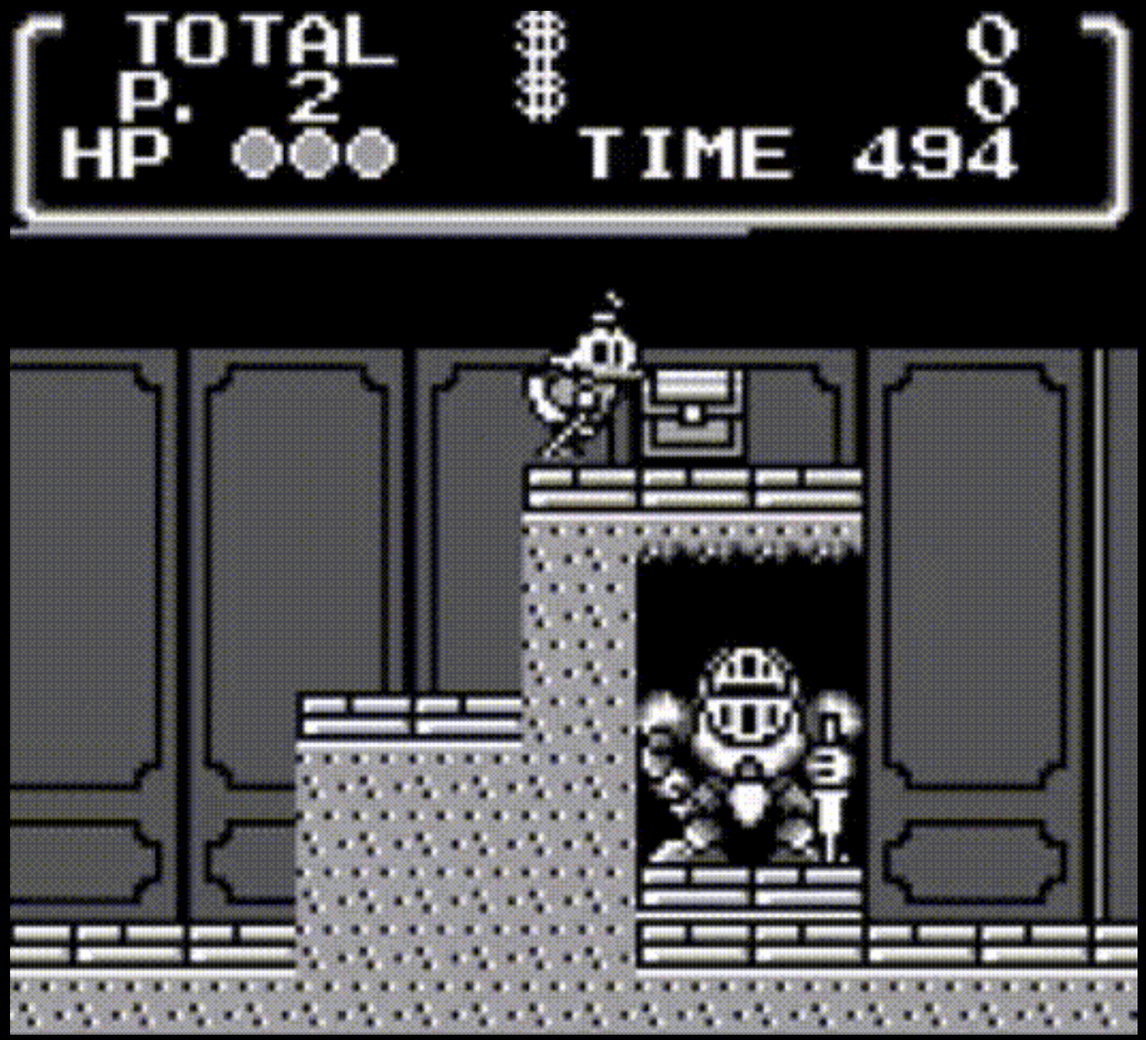


```
.fixed-window {
  position: ‘absolute’;
  z-index: 10;
  top: -80;
}
```

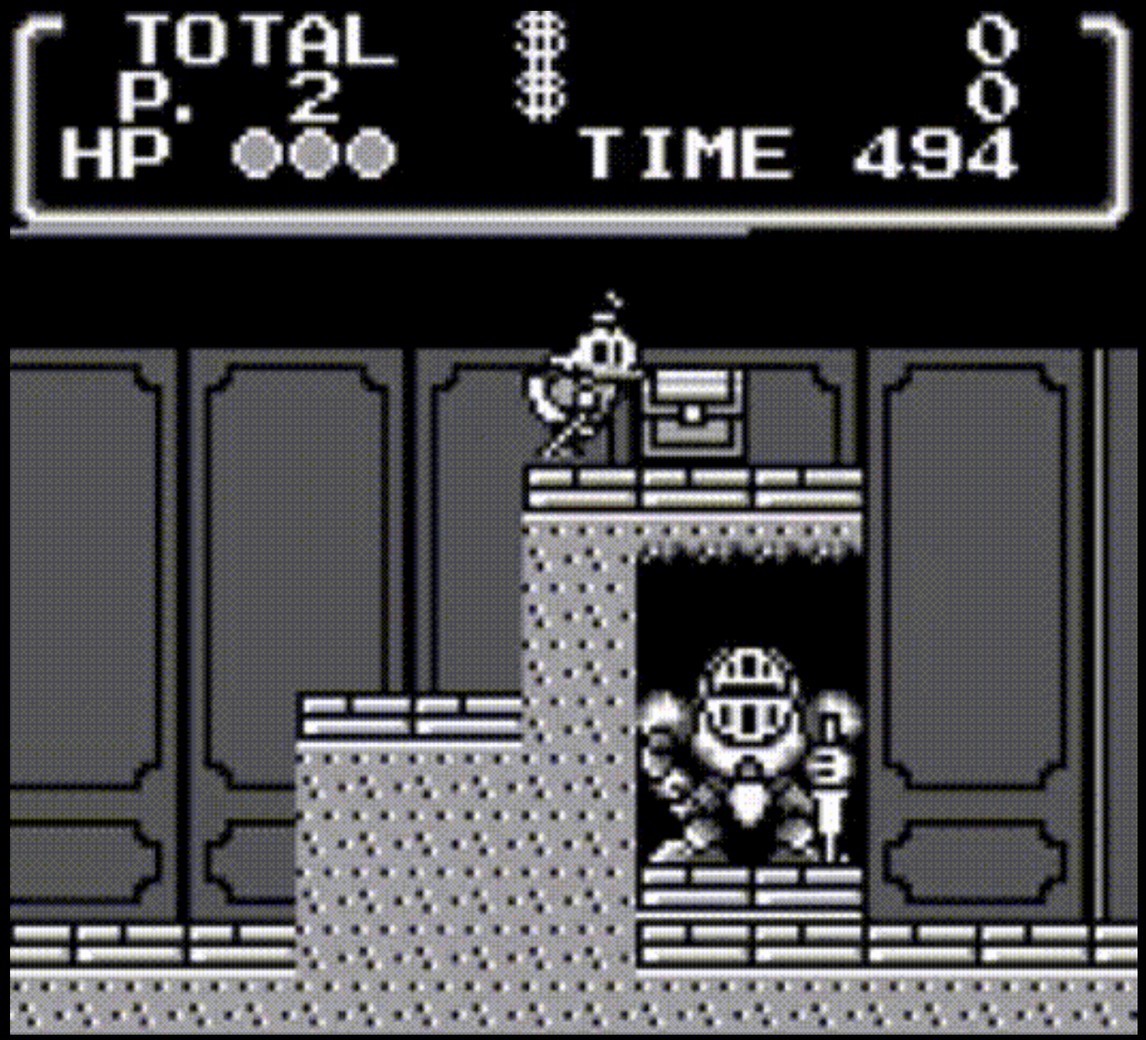


```
.fixed-window {
  position: ‘absolute’;
  z-index: 10;
  top: -80;
}
```

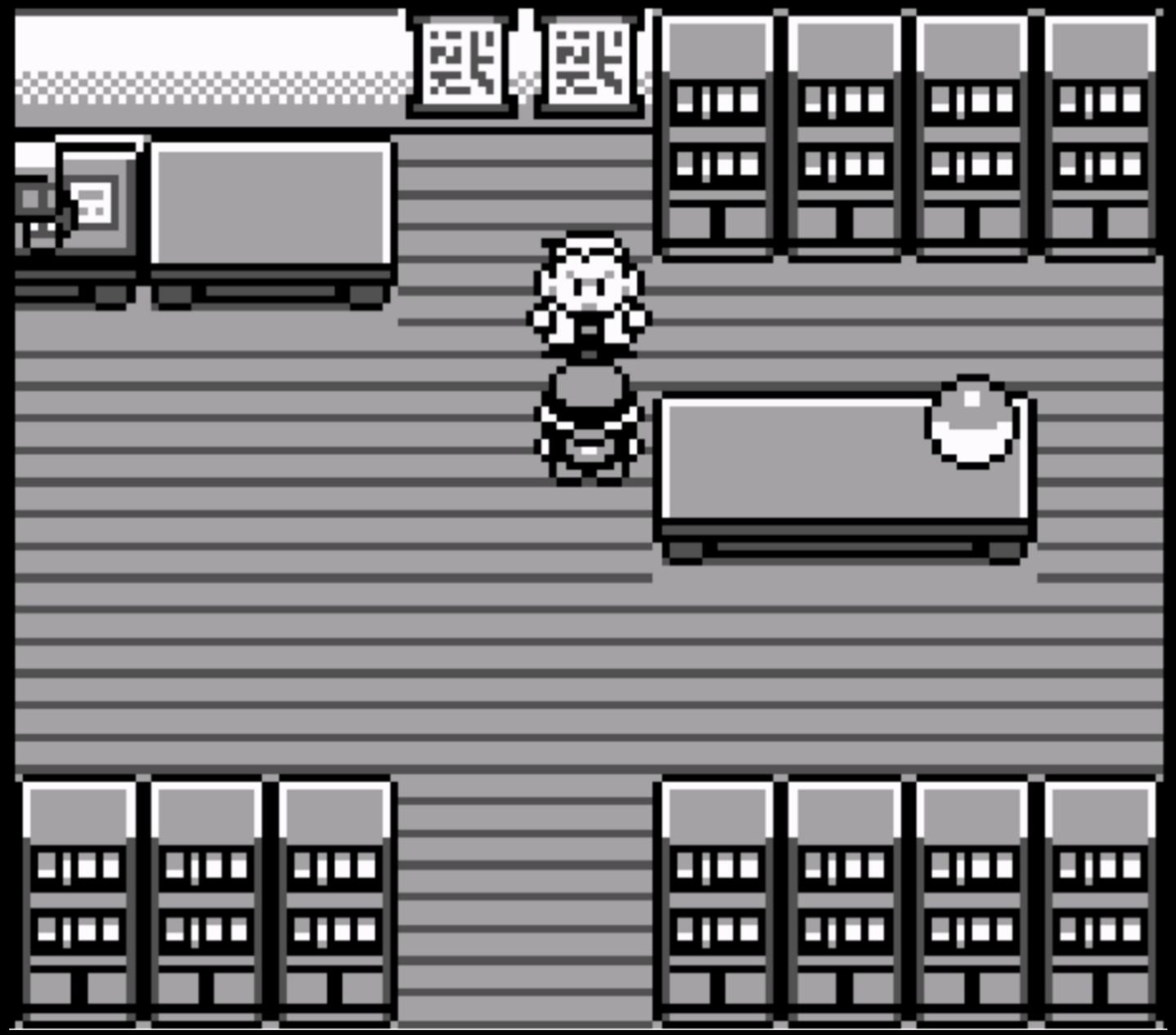



TOTAL
P. 2
HP
TIME 494


MARIO×02
0
×00
WORLD
1-1
TIME
400

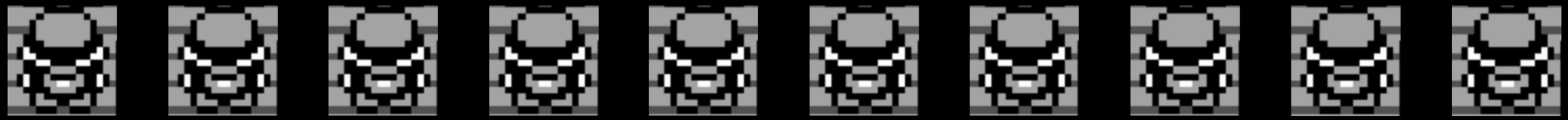

TOTAL
P. 2
HP
TIME 494


MARIO×02
0
×00
WORLD
1-1
TIME
400



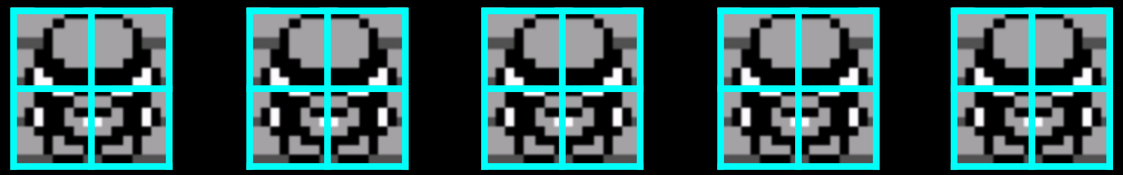
```
.sprite1 {
  position: ‘absolute’;


}
```

```
.sprite1 {
  position: ‘absolute’;
  top: 80;
  left: 40;
}
```

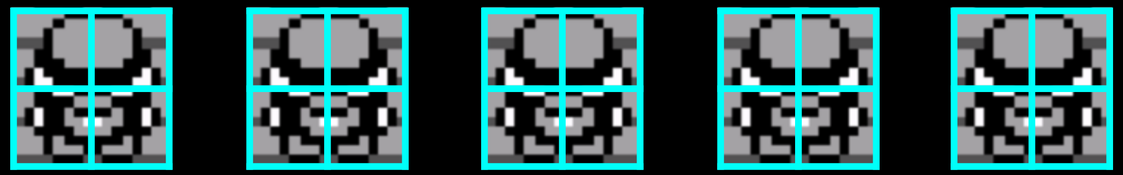


```
.sprite1 {
  position: ‘absolute’;
  top: 80;
  left: 40;
}
```

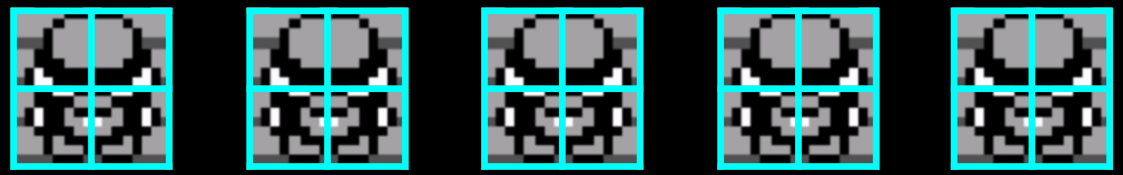
```
.sprite1 {
  position: ‘absolute’;
  top: 80;
  left: 40;
}
```

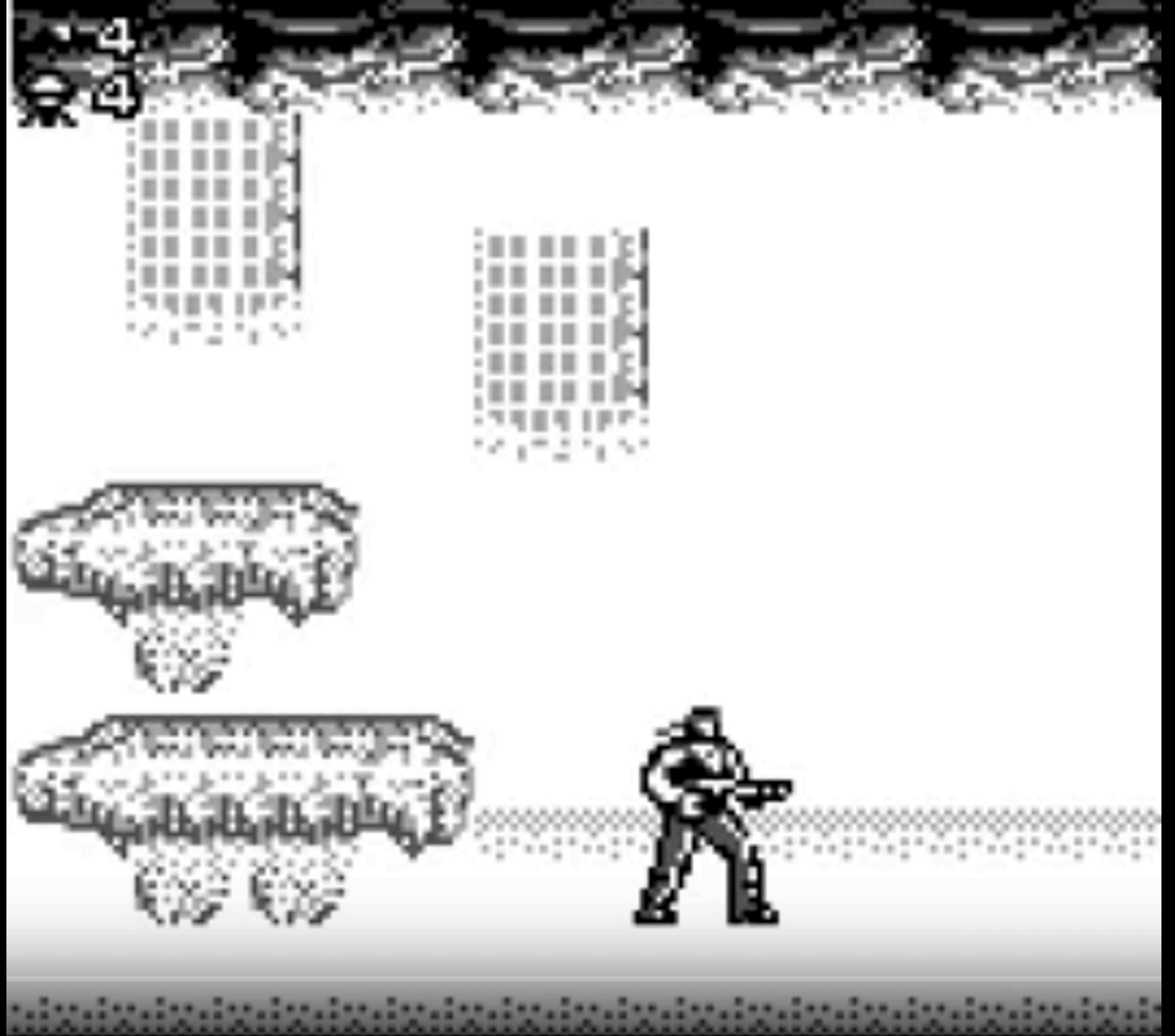


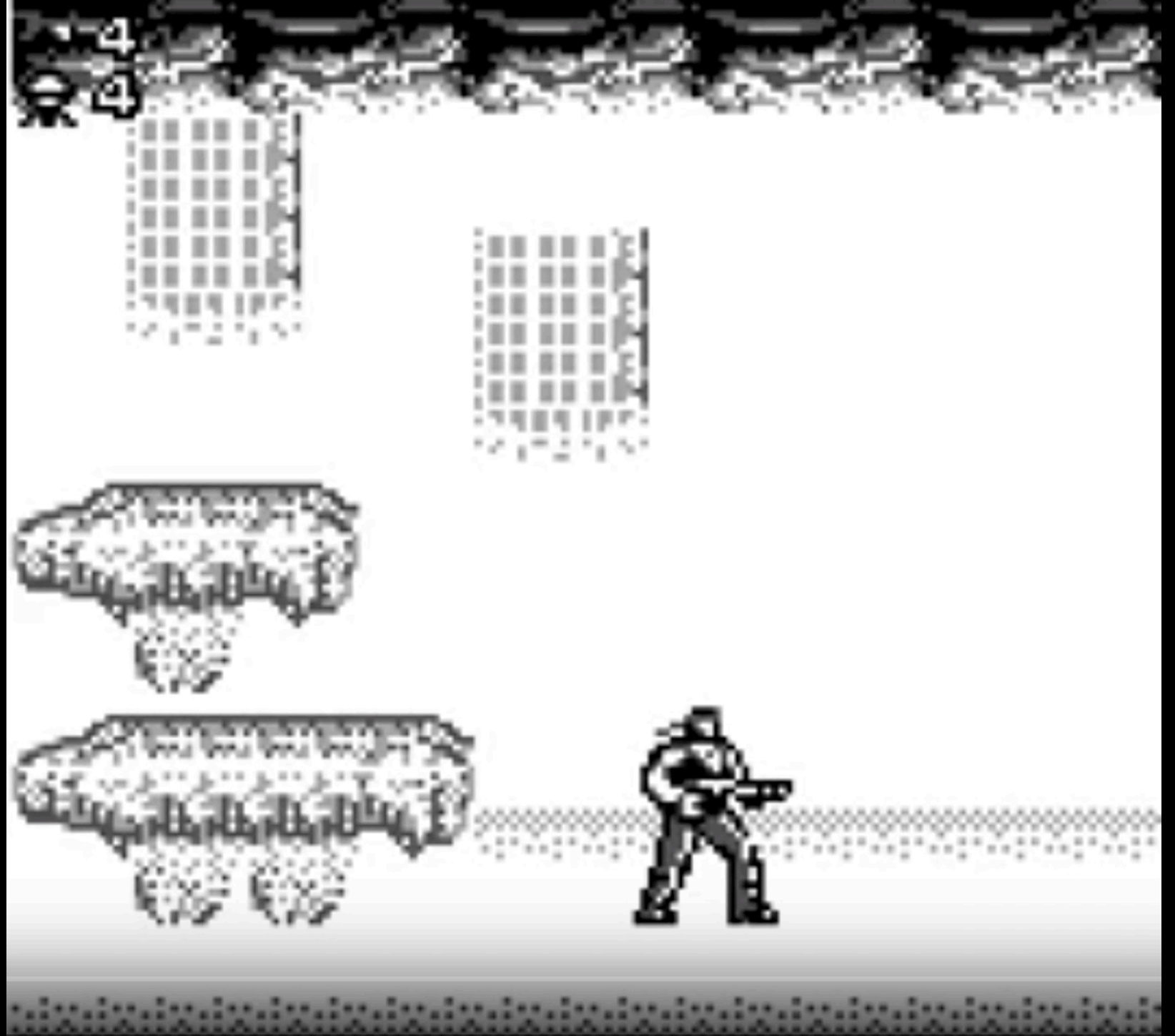
```
.sprite1 {
  position: ‘absolute’;
  top: 80;
  left: 40;
}
```



```
.sprite1 {
  position: ‘absolute’;
  top: 80;
  left: 40;
}
```







References

Mother of all DMG Technical references - https://github.com/gbdev/awesome-gbdev

DMG Technical references I - http://gbdev.gg8.se/

DMG Technical references II - https://cturt.github.io/cinoop.html

Technical Bootstrap Walkthrough - https://realboyemulator.wordpress.com/

[IMG] Cartridge Analysis - https://gekkio.fi

[VID] What it looks like when you turn on a game boy without a game - https://www.youtube.com/watch?v=cyzD8AdpCeE

[VID] World of Long Plays - https://www.youtube.com/user/cubex55

Failed emulator that goes only to the logo - https://github.com/KenLSM/gb-emu

Instruction examples

WARNING: Proceed at your own risk.

01 FF 00

LD BC NN

01 FF 00

LD BC NN

Bytes	01	FF	00

01 FF 00

LD BC NN

Bytes	01	FF	00
Mnemonic	LD BC	N	N
English	Load the next two bytes into register BC		

01 FF 00

LD BC NN

Bytes	01	FF	00
Mnemonic	LD BC	N	N
English	Load the next two bytes into register BC		

B = 00

C = FF

Endianness

01 FF 00

LD BC NN

Endianness

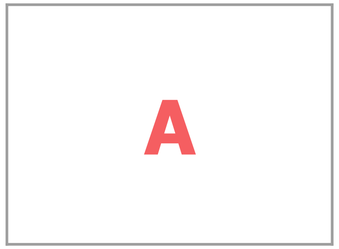
01 FF 00

LD BC NN

Addr.	Data	Little Endian	
x	01	Op Code	
x + 1	FF	LSB	C
x + 2	00	MSB	B

AF Registers

AF Registers



[A]ccumulator

Can perform Arithmetic and Logic

[F]lags

Should only be read from
Not written to

HL Registers

HL Registers

H	L

Special Op Codes available!

LDD A, (HL)
Load Data Decrement

HL Registers

H	L

Special Op Codes available!

LDD A, (HL)
Load Data Decrement

HL → A
HL--

HL Registers - LDD (HL), A

A→ (HL)
HL--

Registers		
H	L	A
FF	FF	0

Address	
0xFFFE	0xFFFF
7	75

HL Registers - LDD (HL), A

A→ (HL)
HL--

Registers		
H	L	A
FF	FF	0

Address	
0xFFFE	0xFFFF
7	0

HL Registers - LDD (HL), A

A→ (HL)
HL--

Registers		
H	L	A
FF	FE	0

Address	
0xFFFE	0xFFFF
7	0

How to write an emulator

[ ] Platform

Which platform are you running on: node.js

[ ] I/O

Visuals: Bash + blessed.js

Audios: N. Y. E.

[ ] Documents

You need them. A lot.