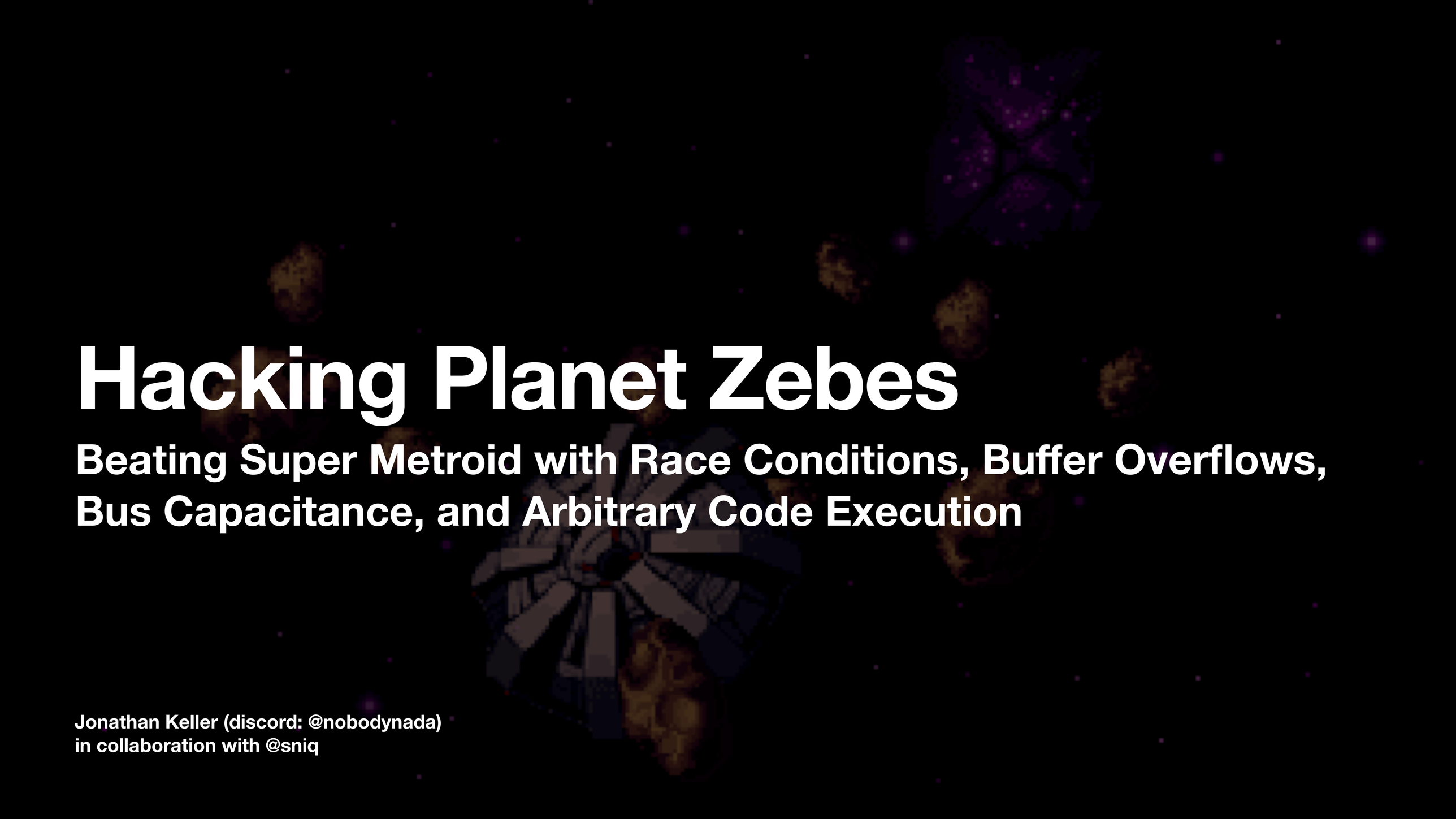


Hacking Planet Zebes

Beating Super Metroid with Race Conditions, Buffer Overflows, Bus Capacitance, and Arbitrary Code Execution

Jonathan Keller (discord: @nobodynada)
in collaboration with @sniq

What's a speedrun?

What counts as "beating the game?"

Category	Description	World Record
any%	Reach the credits	40:36 by Oatsngoats
100%	Collect all items, reach the credits	1:12:40 by ShinyZeni

What's a speedrun?

What counts as "beating the game?"

Category	Description	World Record	Restrictions
any%	Reach the credits	40:36 by Oatsngoats	No Major Glitches
100%	Collect all items, reach the credits	1:12:30 by ShinyZeni	No Major Glitches
any% glitched	Reach the credits	11:00 by Ruya	

What's a speedrun?

What counts as "beating the game?"

Category	Description	World Record	Restrictions
any%	Reach the credits	40:36 by Oatsngoats	No Major Glitches**; a human must be playing the game**
100%	Collect all items, reach the credits	1:12:30 by ShinyZeni	No Major Glitches**; a human must be playing the game**
any% glitched	Reach the credits	11:00 by Ruya	**a human must be playing the game**

What's a speedrun?

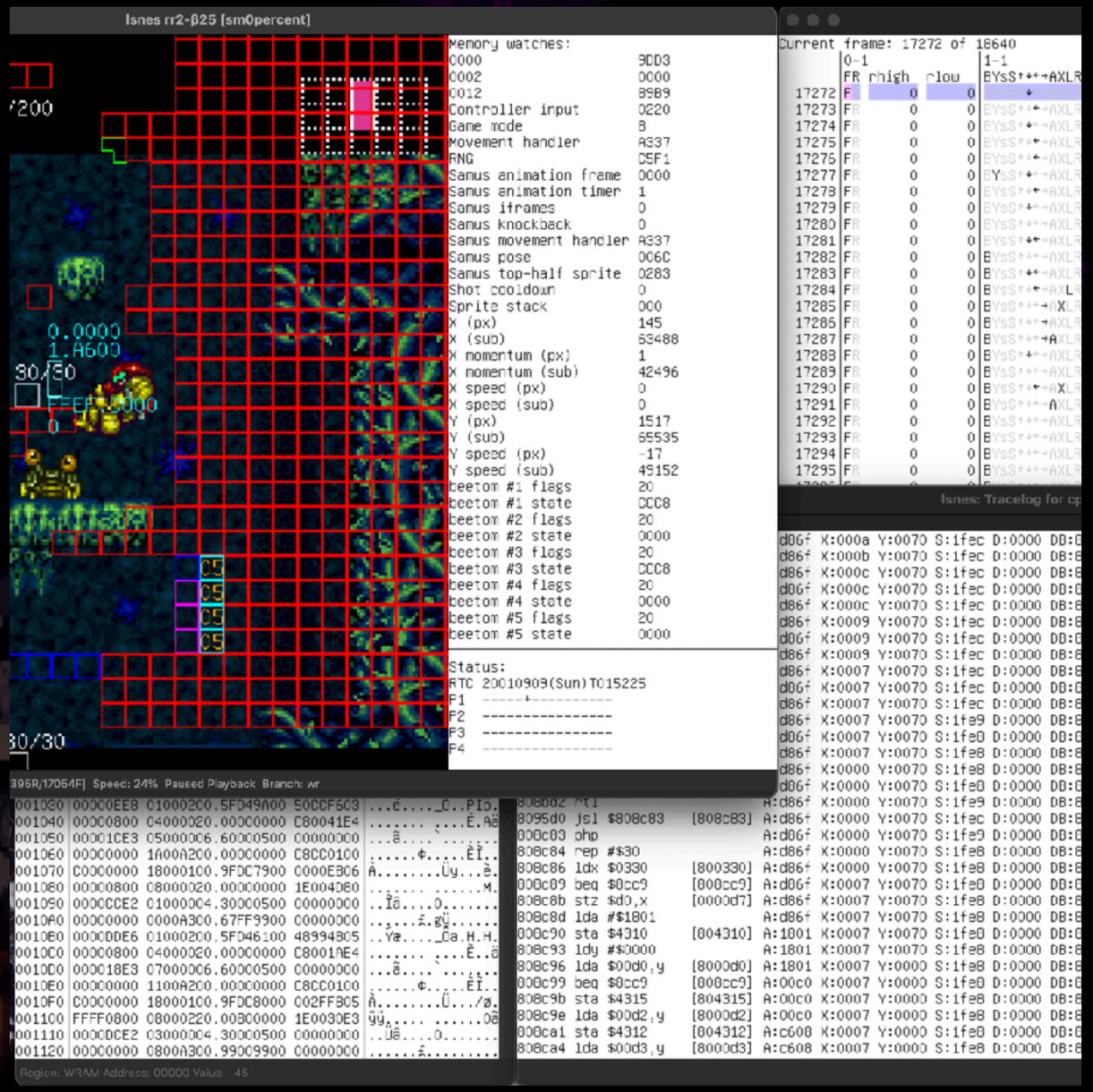
What counts as "beating the game?"

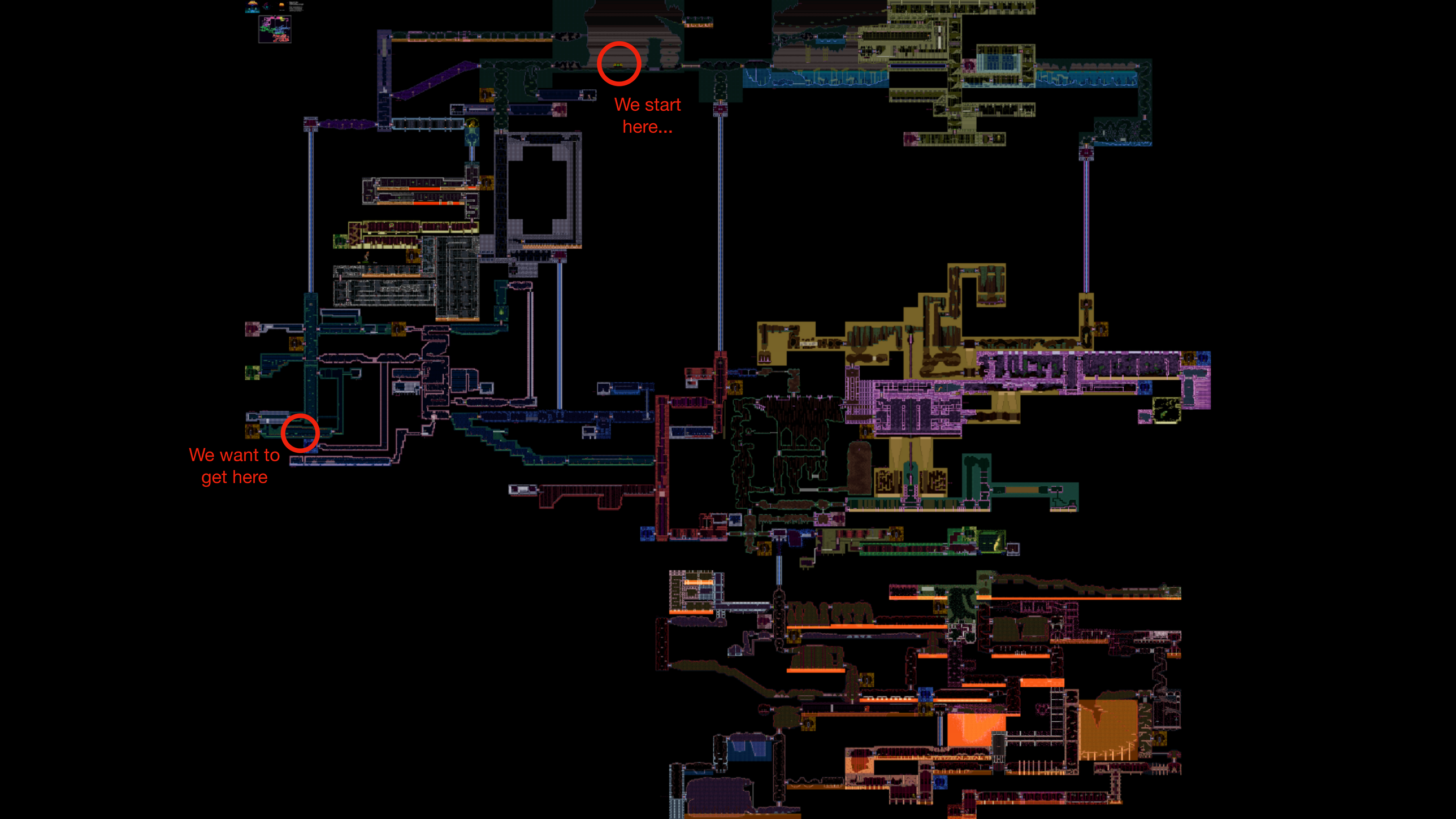
Category	Description	World Record	Restrictions
any%	Reach the credits	40:36 by Oatsngoats	No Major Glitches**; a human must be playing the game**
100%	Collect all items, reach the credits	1:12:30 by ShinyZeni	No Major Glitches**; a human must be playing the game**
any% glitched	Reach the credits	11:00 by Ruya	**a human must be playing the game**
0% (TAS)	Reach the credits	**3:47* by us!** *RTA timing	well, SOMETHING still has to play the game

What's a TAS?

(it's a Tool-Assisted Speedrun!)

- Every button press is meticulously planned out ahead-of-time, and played back by a computer
- "Theoretically perfect" speedrun
- Created with a "recording emulator":
  - Record/replay
  - Save/load state
  - Slowdown/frame advance
  - Rewind/fast-forward
  - Frame-by-frame editing
  - Macros
  - Branches/version control
  - Debugger/instruction trace
  - Memory introspection
  - Scripting




We start here...
We want to get here


We start here...
We want to get here

Obstacle #1

Bomb Blocks

- We want to get through the door to the left, but this wall is blocking our path
- Let's clip out of bounds!

Door Skip

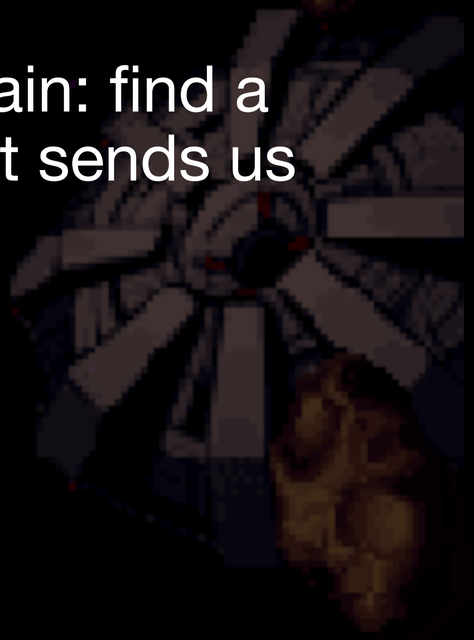
- Frame perfect unpause as Samus activates door transition
- Race condition: two changes to "game state" variable on same frame
- Unpause sets game state to "normal gameplay"; overwrites "starting door transition" value and we fall through the door!

Out of Bounds

- Game stores level data in a 2D array in memory
- No bounds checks; level design is expected to keep us in bounds!
- If we go below the bottom of the room, we buffer-overflow the level data and end up in garbage world
- Find a byte OoB that matches the tile ID of the door we want to go through

Out of Bounds

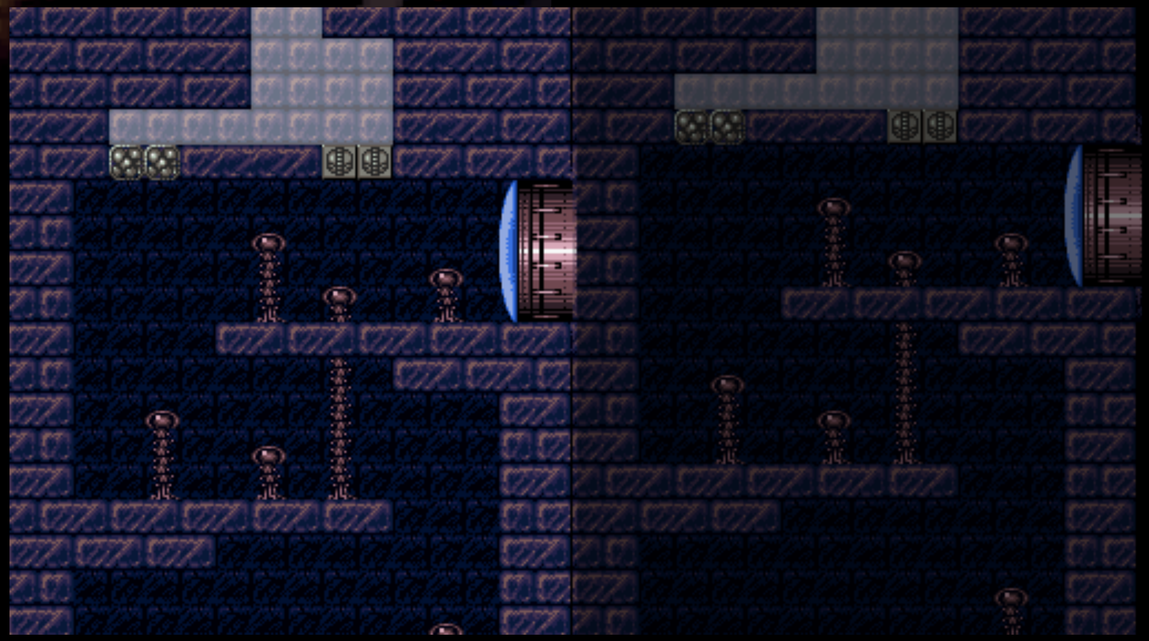
- Triggering a door out-of-bounds often puts us out-of-bounds in the next room. Sometimes, this is convenient!
- Do the same thing again: find a tile out-of-bounds that sends us to the next room



Out of Bounds

But horizontally!

- If we read past the *bottom* of a 2D array, we end up in garbage world
- What if we read past the right side? We wrap around to the next row!
- A[y][x] = *(A + y*width + x)



Out of Bounds

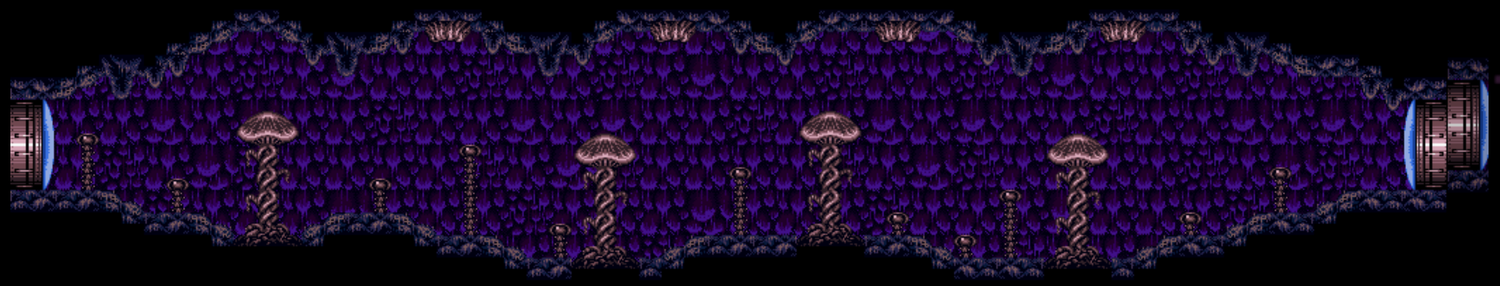
But horizontally!

- We can traverse the room "normally", even though our X position is way off to the right
- Conveniently, we end up in-bounds for the next room

Out of Bounds

But horizontally!

- This room's long, let's do another doorskip
- Going out-of-bounds through the right door wraps us into the left door
- We end up out-of-bounds to the right of the elevator room; standing on the "wrapped" copy of the elevator tiles lets us activate the elevator (which warps us inbounds)



Obstacle #2

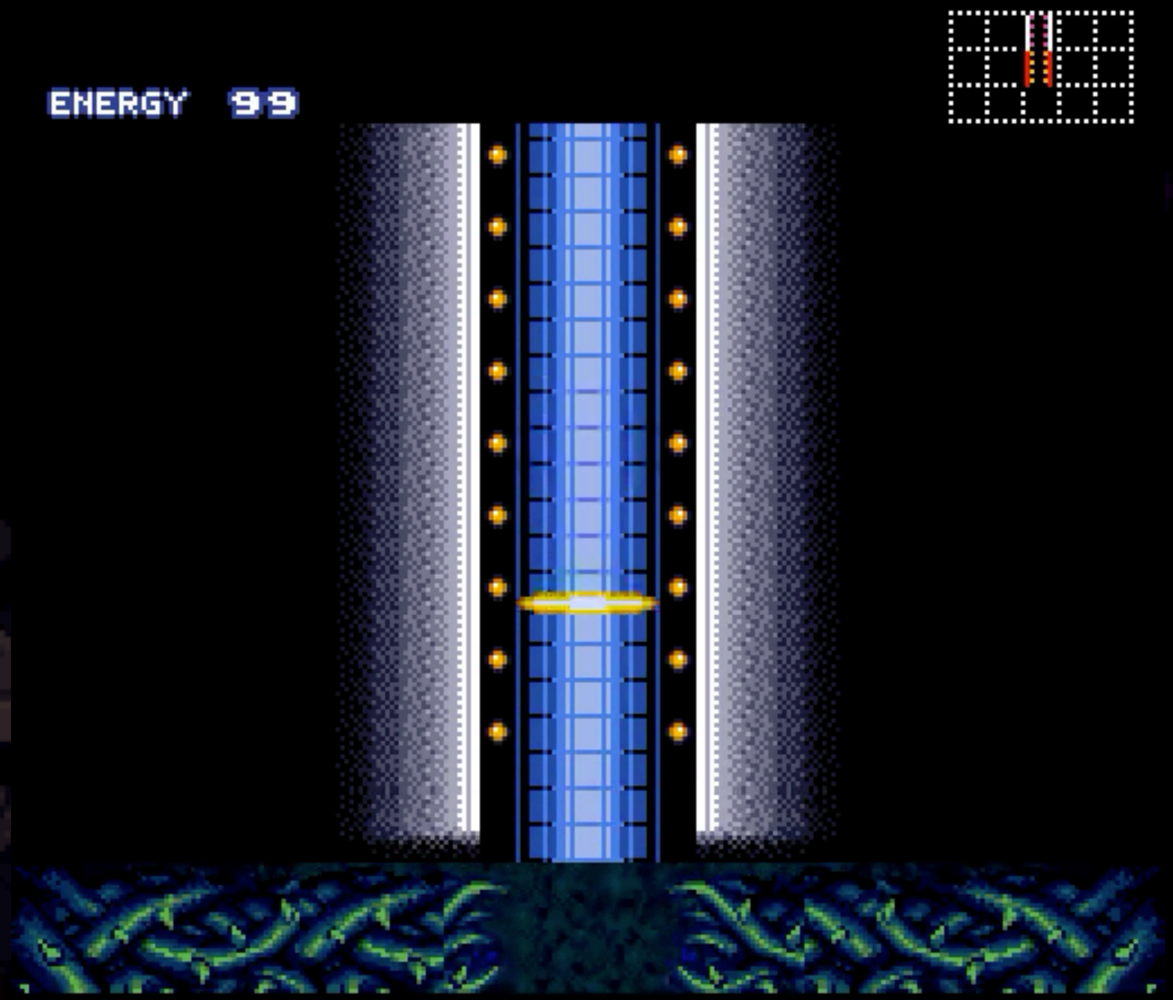
Power Bomb Blocks

- Intended to block this area until midgame
- Fortunately, vertical clips are easier than horizontal clips
- "Moonfall" ignores terminal velocity, allowing us to build up speed and clip through the floor

Obstacle #2

Power Bomb Blocks

- Intended to block this area until midgame
- Fortunately, vertical clips are easier than horizontal clips
- "Moonfall" ignores terminal velocity, allowing us to build up speed and clip through the floor



Obstacle #3

Spiky Tunnel

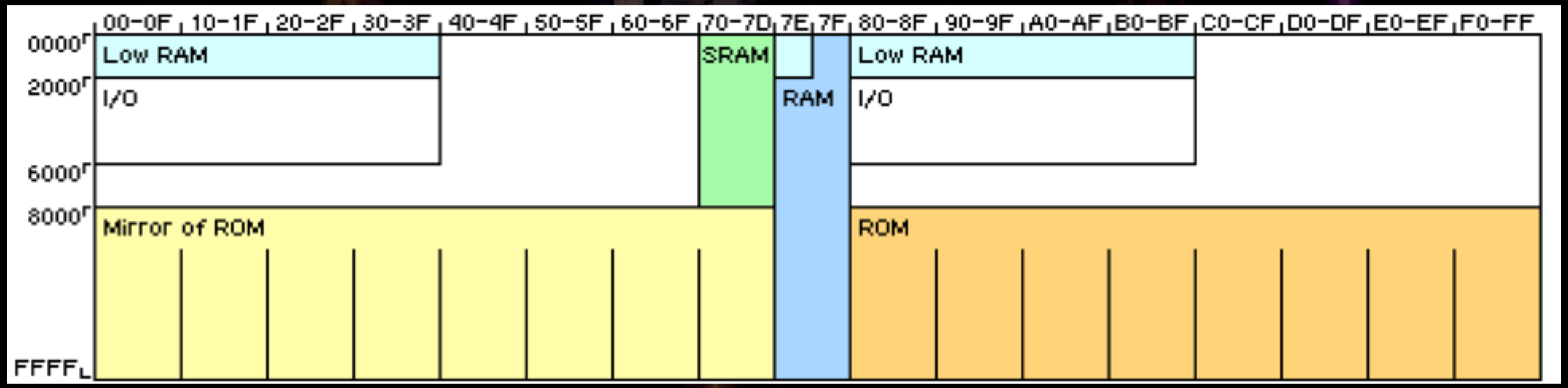
- Precise hitbox manipulation + damage boosts lets us get through without losing too much health
- Then, we do some strange movement, and...

From Speedrun to Pwn Challenge

- We are leaving the world of game mechanics, and entering the world of binary exploitation
- First step: reverse-engineer the relevant parts of the game, understand how everything works!

Architectural Overview
SNES CPU

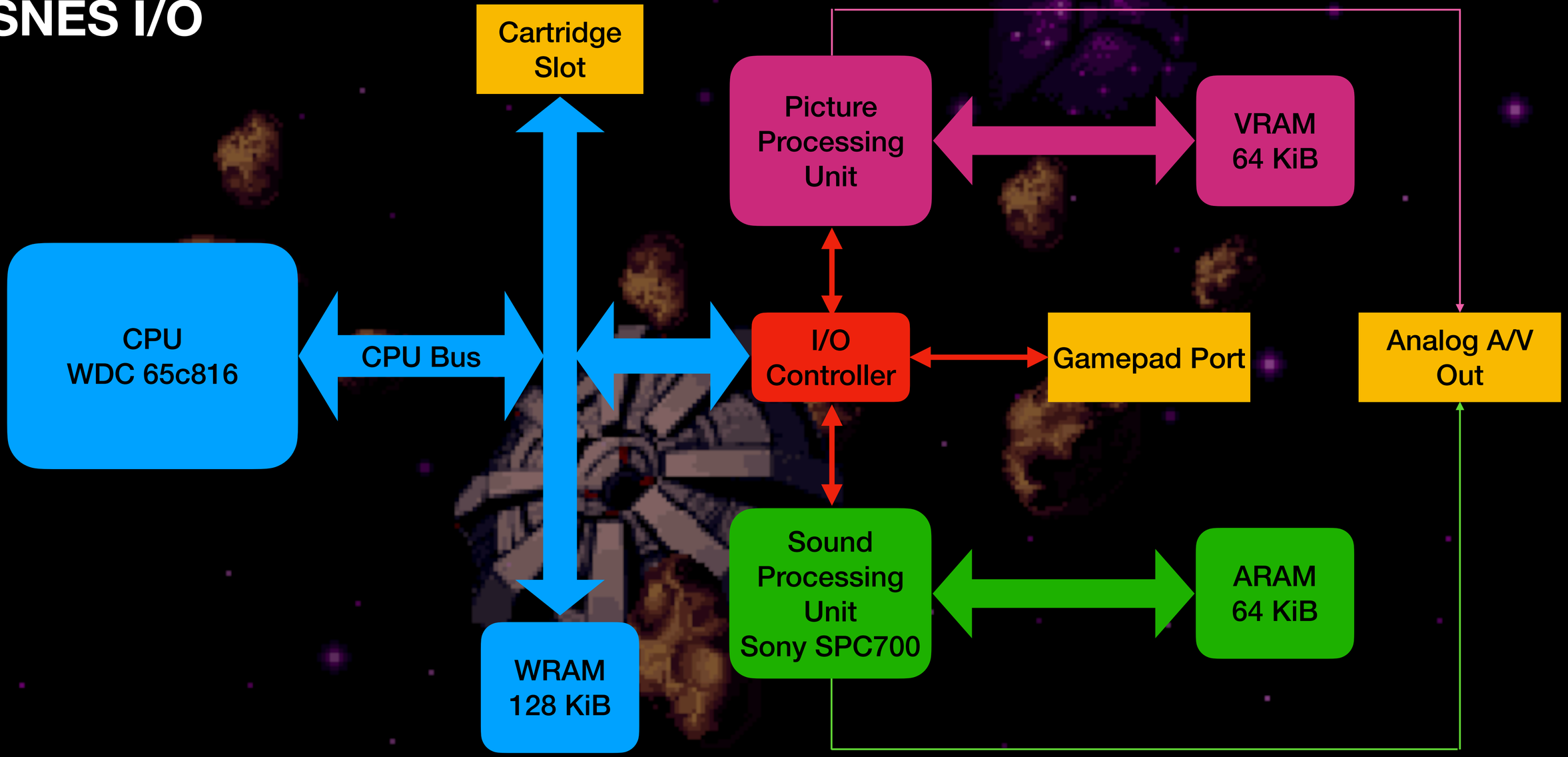
- WDC65c816: 16-bit extension of MOS 6502 family
- 24-bit address bus: 8-bit "bank" and 16-bit pointer (kind of like x86 segments)
- All external hardware access is MMIO



https://snes.nesdev.org/wiki/Memory_map

Architectural Overview

SNES I/O



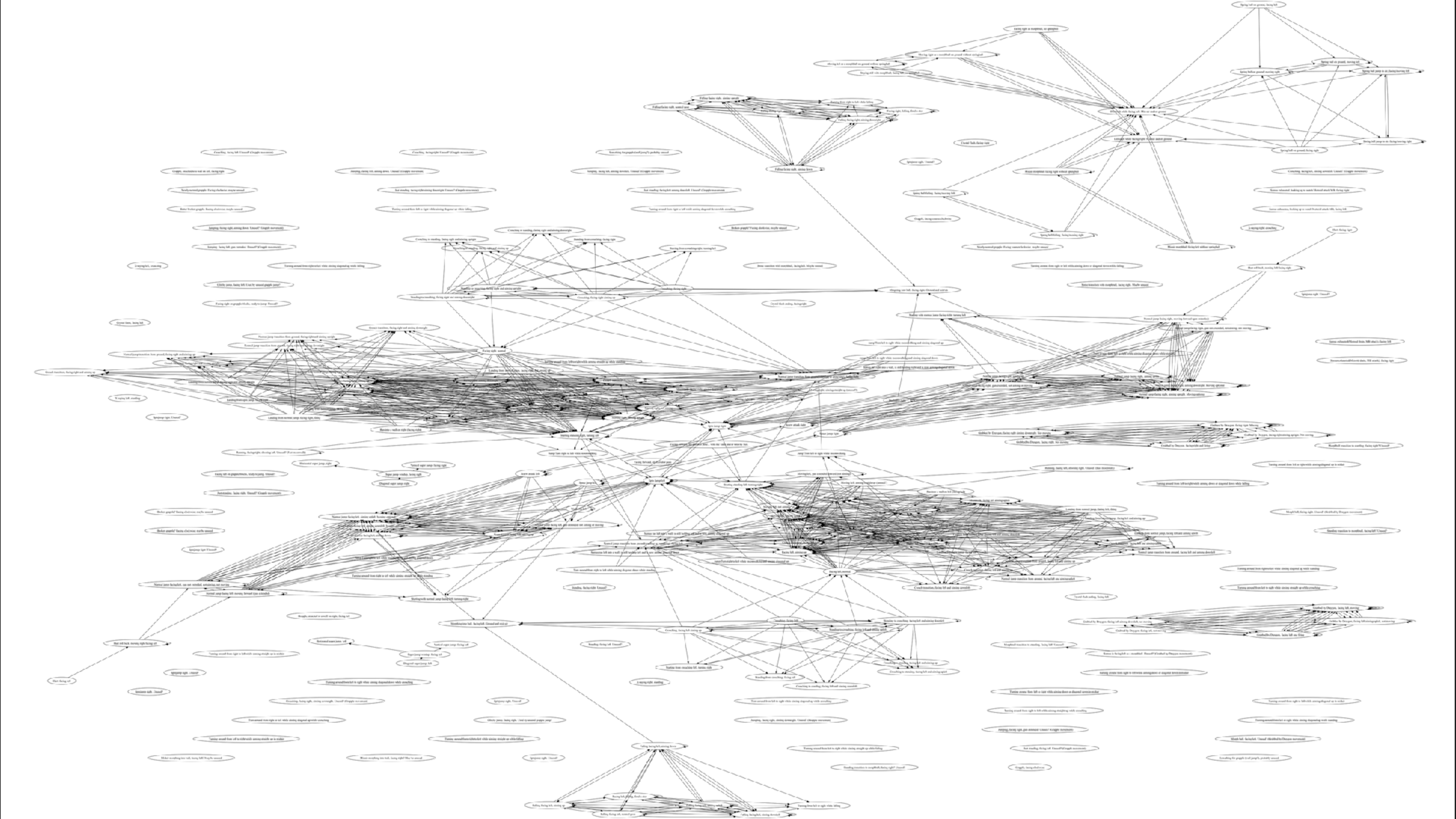
Architectural Overview

HDMA

Poses, Transitions, Animations, and Spritemaps

Poses & Pose Transitions

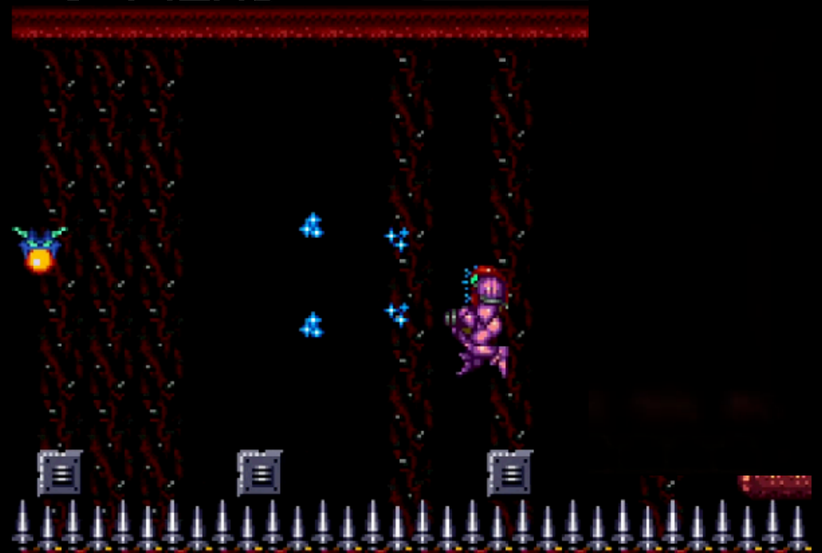
- Samus's movement is implemented as a finite-state machine
- 253 poses!
- Examples:
  - "Facing right - crouching - aiming down-right"
  - "Moving left - gun extended"
  - "Facing left - landing from spin jump"
- Controller input triggers transitions between poses -- edges in state machine
- Let's make a state machine diagram!

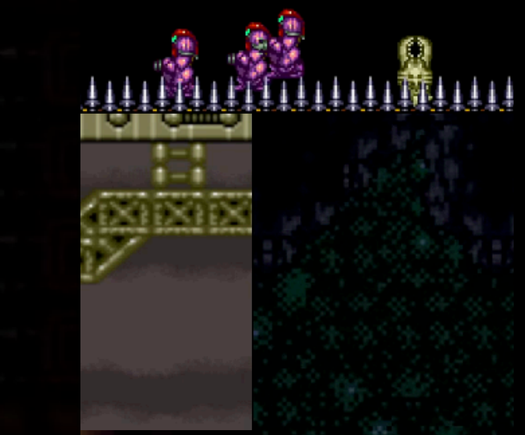


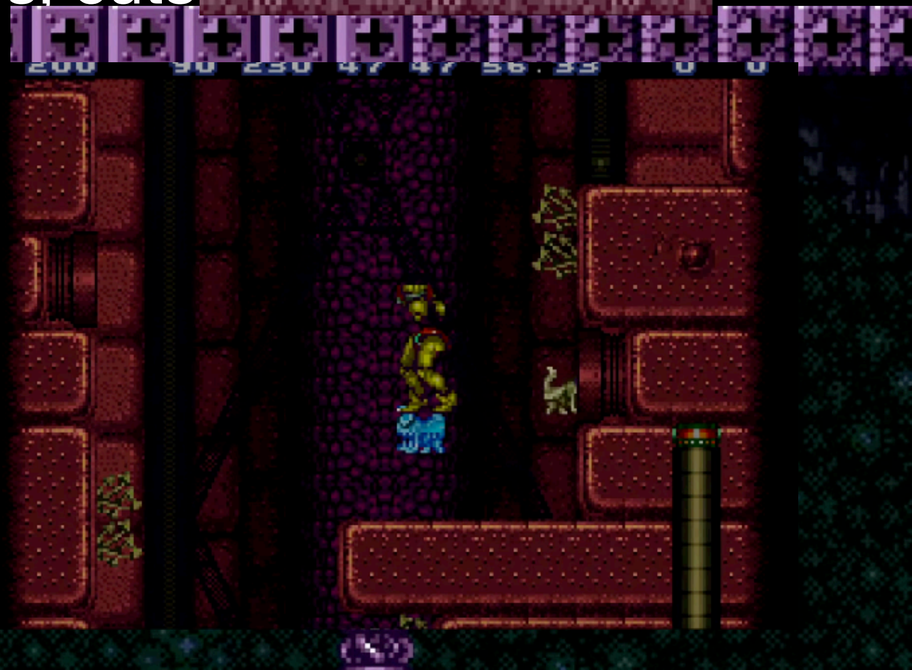
Poses T a Spritemaps

More Tra

- That wa ns
  couple t d pose transitions
  - Da , intera
    col s, cuts
- Elega







video credit: Edu207, ShinyZeni, sniq

Poses, Transitions, Animations, and Spritemaps

Animations

- Each pose defines an associated animation
- An animation is a sequence of animation frames
- Each frame has a spritemap, and an animation delay
- Can also include special instructions to trigger looping, pose transitions, conditional behavior, special interactions, etc.

Poses, Transitions, Animations, and Spritemaps

Examples of Animations

Pose $19

(Spin jump facing right)

Spritemap

Delay	4	3	2	3	2	3	2	3	2	loop

Pose $25

(Standing, turning left-to-right)

Spritemap

Delay	2	2	2	Transition to pose $2 (standing, facing right)

Poses, Transitions, Animations, and Spritemaps

Spritemaps

- A spritemap is a list of tiles to be drawn to the screen as sprites
- A spritemap consists of:
  - Size (number of sprites)
  - One or more sprites:
    - X and Y offset
    - Attributes (flip, palette, priority, etc.)
    - Tile ID

```
;;; $8A5F: Add spritemap to OAM ;;;
{
;; Parameters:
;;     DB:Y: Address of first entry in spritemap
;;     $12: Y position of spritemap centre
;;     $14: X position of spritemap centre
;;     $18: Number of entries

; Spritemap format is roughly:
;     nnnn          ; Number of entries (2 bytes)
;     xxxx yy aatt ; Entry 0 (5 bytes)
;     ...           ; Entry 1...
; Where:
;     n = number of entries
;     x = X offset of sprite from centre
;     y = Y offset of sprite from centre
;     a = attributes
;     t = tile number

; More specifically, a spritemap entry is:
;     s000000xxxxxxxxx yyyyyyyy YXppPPPttttttttt
; Where:
;     s = size bit
;     x = X offset of sprite from centre
;     y = Y offset of sprite from centre
;     Y = Y flip
;     X = X flip
;     P = palette
;     p = priority (relative to background)
;     t = tile number
```

Poses, Transitions, Animations, and Spritemaps

OAM

- Sprites need go to OAM: a "display list" in PPU memory
- But we can only access OAM during blanking periods!
- Solution: Gameplay routines render sprites to a CPU-side buffer; this buffer is DMA'd to OAM during vertical blanking
- Each OAM entry is 4 bytes: Y position, tile ID, attributes, X position

The Pose Glitch

- Discovered in 2020 by sniq
- If we trigger these 3 transitions on the same frame:
  - Damage knockback ends, transitioning Samus into a falling pose
  - Landing animation finishes, transitioning Samus into a standing pose
  - Player presses jump, transitioning Samus into a jumping pose
- Then these transitions cancel each other out, and no pose transition takes place at all!

The Pose Glitch

```
; BRANCH_KNOCKBACK_TIMER_ZERO
$90:DE20 AD 52 0A    LDA $0A52  [$7E:0A52]  ;\
$90:DE23 F0 53       BEQ $53    [$DE78]     ;} If [knockback direction] = 0: go to BRANCH_0A52_IS_0
$90:DE25 AD 1F 0A    LDA $0A1F  [$7E:0A1F]  ;\
$90:DE28 29 FF 00    AND #$00FF             ;|
$90:DE2B C9 0A 00    CMP #$000A             ;} If [Samus movement type] = knockback / crystal flash ending: go to BRANCH_KNOCKBACK_MOVEMENT
$90:DE2E F0 18       BEQ $18    [$DE48]     ;/
$90:DE30 AD 32 0A    LDA $0A32  [$7E:0A32]  ;\
$90:DE33 C9 03 00    CMP #$0003             ;} If [$0A32] = 3:
$90:DE36 D0 08       BNE $08    [$DE40]     ;/
$90:DE38 A9 08 00    LDA #$0008             ;\
$90:DE3B 8D 32 0A    STA $0A32  [$7E:0A32]  ;} $0A32 = 8
$90:DE3E 28          PLP
$90:DE3F 60          RTS                    ; Return

$90:DE40 AD 1C 0A    LDA $0A1C  [$7E:0A1C]  ;\
$90:DE43 8D 2C 0A    STA $0A2C  [$7E:0A2C]  ;} $0A2C = [Samus pose]
$90:DE46 80 28       BRA $28    [$DE70]     ; Go to BRANCH_DE70
```

The knockback transition sees that knockback was interrupted, and aborts

```
$90:8370 AD 60 0A    LDA $0A60  [$7E:0A60]  ;\
$90:8373 C9 1D E9    CMP #$E91D             ;} If [$0A60] != $E91D: (not demo)
$90:8376 F0 1D       BEQ $1D    [$8395]     ;/
$90:8378 AD 28 0A    LDA $0A28  [$7E:0A28]  ;\
$90:837B C9 4B 00    CMP #$004B             ;|
$90:837E F0 18       BEQ $18    [$8398]     ;} If [$0A28] = normal jump transition: return
$90:8380 C9 4C 00    CMP #$004C             ;|
$90:8383 F0 13       BEQ $13    [$8398]     ;/
$90:8385 C9 19 00    CMP #$0019             ;\
$90:8388 F0 0E       BEQ $0E    [$8398]     ;|
$90:838A C9 1A 00    CMP #$001A             ;} If [$0A28] = spin jump: return
$90:838D F0 09       BEQ $09    [$8398]     ;/
$90:838F A9 26 E9    LDA #$E926             ;\
$90:8392 8D 60 0A    STA $0A60  [$7E:0A60]  ;} $0A60 = $E926 (auto-jump hack)
```

The landing transition sees that Samus is trying to jump, and aborts

```
$91:EB8E AD 2C 0A    LDA $0A2C  [$7E:0A2C]  ;\
$91:EB91 30 32       BMI $32    [$EBC5]     ;} If [$0A2C] & 8000h != 0: go to BRANCH_NOT_0A2C
$91:EB93 48          PHA
$91:EB94 AD 32 0A    LDA $0A32  [$7E:0A32]  ;\
$91:EB97 C9 03 00    CMP #$0003             ;} If [$0A32] = 3: go to BRANCH_0A32_3
$91:EB9A F0 08       BEQ $08    [$EBA4]     ;/
$91:EB9C C9 01 00    CMP #$0001             ;\
$91:EB9F D0 0E       BNE $0E    [$EBAF]     ;} If [$0A32] != 1: go to BRANCH_0A2C
$91:EBA1 68          PLA
$91:EBA2 80 17       BRA $17    [$EBBB]     ; Go to BRANCH_0A32_1

; BRANCH_0A32_3
$91:EBA4 AD 30 0A    LDA $0A30  [$7E:0A30]  ;\
$91:EBA7 C9 09 00    CMP #$0009             ;} If [$0A30] = 9:
$91:EBAA D0 03       BNE $03    [$EBAF]     ;/
$91:EBAC 68          PLA
$91:EBAD 80 16       BRA $16    [$EBC5]     ; Go to BRANCH_NOT_0A2C

; BRANCH_0A2C
$91:EBAF 68          PLA
$91:EBB0 8D 1C 0A    STA $0A1C  [$7E:0A1C]  ; Samus pose = [$0A2C]
$91:EBB3 22 33 F4 91 JSL $91F433[$91:F433]  ; Execute $91:F433
$91:EBB7 22 08 FB 91 JSL $91FB08[$91:FB08]  ; Set Samus animation frame if pose changed
```

Knockback takes priority over jumping, so the jump never occurs

The Pose Glitch

- End result: no pose transition occurs, and the landing animation advances to the next frame
- But we've reached the end of the landing animation
- Buffer overflow! We start reading instructions for the next pose in the animation delay table

```
; E5h: Facing left  - landing from normal jump - aiming down-left
$91:B253              db 05, 02, F8,08

; E8h: Facing right - Samus drained - crouching/falling
$91:B257              db 02, 02, 02, 10, F7,
                         01, FE,01,
                         10, 10, 10, 10, FE,04,
                         03, FD,01
```

The Pose Glitch

```
; E5h: Facing left  - landing from normal jump - aiming down-left
$91:B253              db 05, 02, F8,08

; E8h: Facing right - Samus drained - crouching/falling
$91:B257              db 02, 02, 02, 10, F7,
                         01, FE,01,
                         10, 10, 10, 10, FE,04,
                         03, FD,01
```

- We overflowed into animation instructions for pose $E8, a special pose only used in cutscenes: Samus falls to the ground and collapses
- Implemented using animation instruction $F7: "When Samus touches the ground, go to animation frame 7"
- Not intended to be triggered while the player has control over Samus
- If we trigger a pose transition while we touch the ground, we'll go to frame 7 of the new pose!

Sprite Lag

- What can we do with this?
- Get buffer overflows in most poses
- Access unused or invalid animation frames, including frames with null spritemap
- Causes game to interpret memory address 0 as number of sprites in spritemap
- If this location contains a pointer, tries to draw tens of thousands of sprites -> very low FPS
- Known for years, not very interesting...or is it?

Sprite Lag

```
;;; $89AE: Add Samus spritemap to OAM ;;;
{
;; Parameters:
;;     A: Index into $92:808D table
;;     X: X position of spritemap centre
;;     Y: Y position of spritemap centre

; Actually uses the palette defined in the spritemap
; See $8A5F for spritemap format
; Also called for some atmospheric graphics:
;     186h..18Eh: Bubbles
;     18Fh..197h: Diving splash
$81:89AE 8B          PHB
$81:89AF F4 00 92    PEA $9200                ;\
$81:89B2 AB          PLB                      ;} DB = $92
$81:89B3 AB          PLB                      ;/
$81:89B4 84 12       STY $12     [$7E:0012]   ; $12 = [Y]
$81:89B6 86 14       STX $14     [$7E:0014]   ; $14 = [X]
$81:89B8 0A          ASL A                    ;\
$81:89B9 AA          TAX                      ;} Y = [$808D + [A] * 2] (address of spritemap)
$81:89BA BC 8D 80    LDY $808D,x[$92:83D3]    ;/
$81:89BD B9 00 00    LDA $0000,y[$92:A0B4]    ;\
$81:89C0 F0 73       BEQ $73     [$8A35]      ;} If [[Y]] = 0: return
$81:89C2 85 18       STA $18     [$7E:0018]   ; $18 = [[Y]] (size)
$81:89C4 C8          INY                      ;\
$81:89C5 C8          INY                      ;} Y += 2
$81:89C6 AE 90 05    LDX $0590   [$7E:0590]   ; X = [OAM stack pointer]
$81:89C9 18          CLC

; LOOP
$81:89CA B9 00 00    LDA $0000,y[$92:A0B6]    ;\
$81:89CD 65 14       ADC $14     [$7E:0014]   ;} OAM entry X position = [$14] + [[Y]] (X position)
$81:89CF 9D 70 03    STA $0370,x[$7E:0370]    ;/
$81:89D2 29 00 01    AND #$0100               ;\
$81:89D5 F0 27       BEQ $27     [$89FE]      ;} If [OAM entry X position] & 200h < 100h: BRANCH_X_HIGH_CLEAR
$81:89D7 B9 00 00    LDA $0000,y[$92:A0B6]    ;\
$81:89DA 10 11       BPL $11     [$89ED]      ;} If [[Y]] & 8000h (size bit) != 0:
$81:89DC BF 9F 85 81 LDA $81859F,x[$81:85BF]  ;\
$81:89E0 85 16       STA $16     [$7E:0016]   ;|
$81:89E2 B2 16       LDA ($16)   [$7E:0572]   ;} Set OAM entry high X position bit and size bit
$81:89E4 1F A1 85 81 ORA $8185A1,x[$81:85C1]  ;|
$81:89E8 92 16       STA ($16)   [$7E:0572]   ;/
$81:89EA 4C 11 8A    JMP $8A11   [$81:8A11]   ; Go to BRANCH_MERGE

$81:89ED BF 9F 85 81 LDA $81859F,x[$81:85C7]  ;\
$81:89F1 85 16       STA $16     [$7E:0016]   ;|
$81:89F3 B2 16       LDA ($16)   [$7E:0572]   ;} Set OAM entry high X position bit
$81:89F5 1F 9F 83 81 ORA $81839F,x[$81:83C7]  ;|
$81:89F9 92 16       STA ($16)   [$7E:0572]   ;/
$81:89FB 4C 11 8A    JMP $8A11   [$81:8A11]   ; Go to BRANCH_MERGE

; BRANCH_X_HIGH_CLEAR
$81:89FE B9 00 00    LDA $0000,y[$92:A0B6]    ;\
$81:8A01 10 0E       BPL $0E     [$8A11]      ;} If [[Y]] & 8000h (size bit) != 0:
$81:8A03 BF 9F 85 81 LDA $81859F,x[$81:859F]  ;\
$81:8A07 85 16       STA $16     [$7E:0016]   ;|
$81:8A09 B2 16       LDA ($16)   [$7E:0570]   ;} Set OAM entry size bit
$81:8A0B 1F A1 83 81 ORA $8183A1,x[$81:83A1]  ;|
$81:8A0F 92 16       STA ($16)   [$7E:0570]   ;/

; BRANCH_MERGE
$81:8A11 B9 02 00    LDA $0002,y[$92:A0B8]    ;\
$81:8A14 18          CLC                      ;|
$81:8A15 65 12       ADC $12     [$7E:0012]   ;} OAM entry Y position = [$12] + [[Y] + 2] (Y position)
$81:8A17 9D 71 03    STA $0371,x[$7E:0371]    ;/
$81:8A1A B9 03 00    LDA $0003,y[$92:A0B9]    ;\
$81:8A1D 9D 72 03    STA $0372,x[$7E:0372]    ;} OAM entry tile number and attributes = [[Y] + 3]
$81:8A20 98          TYA                      ;\
$81:8A21 18          CLC                      ;|
$81:8A22 69 05 00    ADC #$0005               ;} Y += 5 (next sprite map entry)
$81:8A25 A8          TAY                      ;/
$81:8A26 8A          TXA                      ;\
$81:8A27 69 04 00    ADC #$0004               ;|
$81:8A2A 29 FF 01    AND #$01FF               ;} X += 4 (next OAM entry)
$81:8A2D AA          TAX                      ;/
$81:8A2E C6 18       DEC $18     [$7E:0018]   ; Decrement $18
$81:8A30 D0 98       BNE $98     [$89CA]      ; If [$18] != 0: go to LOOP
$81:8A32 8E 90 05    STX $0590   [$7E:0590]   ; OAM stack pointer = [X]

$81:8A35 AB          PLB
$81:8A36 6B          RTL
}
```

Sprite Lag

```
;;; $896E: Finalise OAM ;;;
{
; Move unused sprites to Y = F0h and reset OAM stack pointer
; Uses one hell of an unrolled loop
$80:896E 08          PHP
$80:896F C2 30       REP #$30
$80:8971 AD 90 05    LDA $0590  [$7E:0590]  ;\
$80:8974 C9 00 02    CMP #$0200             ;} If [OAM stack pointer] < 200h:
$80:8977 10 14       BPL $14    [$898D]     ;/
$80:8979 4A          LSR A                  ;\
$80:897A 85 12       STA $12    [$7E:0012]  ;|
$80:897C 4A          LSR A                  ;|
$80:897D 65 12       ADC $12    [$7E:0012]  ;} $12 = $8992 + [OAM stack pointer] / 4 * 3
$80:897F 18          CLC                    ;|
$80:8980 69 92 89    ADC #$8992             ;|
$80:8983 85 12       STA $12    [$7E:0012]  ;/
$80:8985 A9 F0 00    LDA #$00F0             ; A = F0h (sprite Y position)
$80:8988 E2 30       SEP #$30
$80:898A 6C 12 00    JMP ($0012)[$80:8992]  ; Go to [$12]

$80:898D 9C 90 05    STZ $0590  [$7E:0590]  ; Clear OAM stack pointer = 0
$80:8990 28          PLP
$80:8991 6B          RTL

$80:8992 8D 71 03    STA $0371  [$7E:0371]  ; Sprite 0 Y position = F0h
$80:8995 8D 75 03    STA $0375  [$7E:0375]  ; Sprite 1 Y position = F0h
$80:8998 8D 79 03    STA $0379  [$7E:0379]  ; Sprite 2 Y position = F0h
$80:899B 8D 7D 03    STA $037D  [$7E:037D]  ; Sprite 3 Y position = F0h
$80:899E 8D 81 03    STA $0381  [$7E:0381]  ; Sprite 4 Y position = F0h
$80:89A1 8D 85 03    STA $0385  [$7E:0385]  ; Sprite 5 Y position = F0h
$80:89A4 8D 89 03    STA $0389  [$7E:0389]  ; Sprite 6 Y position = F0h
$80:89A7 8D 8D 03    STA $038D  [$7E:038D]  ; Sprite 7 Y position = F0h
$80:89AA 8D 91 03    STA $0391  [$7E:0391]  ; Sprite 8 Y position = F0h
$80:89AD 8D 95 03    STA $0395  [$7E:0395]  ; Sprite 9 Y position = F0h
$80:89B0 8D 99 03    STA $0399  [$7E:0399]  ; Sprite Ah Y position = F0h
$80:89B3 8D 9D 03    STA $039D  [$7E:039D]  ; Sprite Bh Y position = F0h
$80:89B6 8D A1 03    STA $03A1  [$7E:03A1]  ; Sprite Ch Y position = F0h

$80:8AFA 8D 51 05    STA $0551  [$7E:0551]  ; Sprite 78h Y position = F0h
$80:8AFD 8D 55 05    STA $0555  [$7E:0555]  ; Sprite 79h Y position = F0h
$80:8B00 8D 59 05    STA $0559  [$7E:0559]  ; Sprite 7Ah Y position = F0h
$80:8B03 8D 5D 05    STA $055D  [$7E:055D]  ; Sprite 7Bh Y position = F0h
$80:8B06 8D 61 05    STA $0561  [$7E:0561]  ; Sprite 7Ch Y position = F0h
$80:8B09 8D 65 05    STA $0565  [$7E:0565]  ; Sprite 7Dh Y position = F0h
$80:8B0C 8D 69 05    STA $0569  [$7E:0569]  ; Sprite 7Eh Y position = F0h
$80:8B0F 8D 6D 05    STA $056D  [$7E:056D]  ; Sprite 7Fh Y position = F0h
$80:8B12 9C 90 05    STZ $0590  [$7E:0590]  ;\
$80:8B15 9C 91 05    STZ $0591  [$7E:0591]  ;} OAM stack pointer = 0
$80:8B18 28          PLP
$80:8B19 6B          RTL
}
```

Riding the Open Bus

```
$80:8AFA 8D 51 05    STA $0551  [$7E:0551]  ; Sprite 78h Y position = F0h
$80:8AFD 8D 55 05    STA $0555  [$7E:0555]  ; Sprite 79h Y position = F0h
$80:8B00 8D 59 05    STA $0559  [$7E:0559]  ; Sprite 7Ah Y position = F0h
$80:8B03 8D 5D 05    STA $055D  [$7E:055D]  ; Sprite 7Bh Y position = F0h
$80:8B06 8D 61 05    STA $0561  [$7E:0561]  ; Sprite 7Ch Y position = F0h
$80:8B09 8D 65 05    STA $0565  [$7E:0565]  ; Sprite 7Dh Y position = F0h
$80:8B0C 8D 69 05    STA $0569  [$7E:0569]  ; Sprite 7Eh Y position = F0h
$80:8B0F 8D 6D 05    STA $056D  [$7E:056D]  ; Sprite 7Fh Y position = F0h
$80:8B12 9C 90 05    STZ $0590  [$7E:0590]  ;\
$80:8B15 9C 91 05    STZ $0591  [$7E:0591]  ;} OAM stack pointer = 0
$80:8B18 28          PLP
$80:8B19 6B          RTL
}
```

```
80898a jmp ($0012)   [800012] A:00f0 X:0002 Y:0003 S:1ffa D:0000 DB:82 nvMXdizc V:294 H: 682 MDR: 30
808b11 ora $9c       [00009c] A:00f0 X:0002 Y:0003 S:1ffa D:0000 DB:82 nvMXdizc V:294 H: 716 MDR: 8b
808b13 bcc $8b1a     [808b1a] A:00f0 X:0002 Y:0003 S:1ffa D:0000 DB:82 NvMXdizc V:294 H: 736 MDR: 00
808b1a php                    A:00f0 X:0002 Y:0003 S:1ffa D:0000 DB:82 NvMXdizc V:294 H: 754 MDR: 05
808b1b rep #$30               A:00f0 X:0002 Y:0003 S:1ff9 D:0000 DB:82 NvMXdizc V:294 H: 774 MDR: b0
[...]
808b4d plp                    A:00f0 X:0002 Y:0003 S:1ff9 D:0000 DB:82 Nvmxdizc V:294 H:1336 MDR: 00
808b4e rtl                    A:00f0 X:0002 Y:0003 S:1ffa D:0000 DB:82 NvMXdizc V:294 H:1362 MDR: b0
897501 bit #$89               A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:295 H:  40 MDR: 89
897503 bit #$89               A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:295 H:  56 MDR: 89
897505 bit #$89               A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:295 H:  72 MDR: 89
897507 bit #$89               A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:295 H:  88 MDR: 89
897509 bit #$89               A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:295 H: 104 MDR: 89
```

Riding the Open Bus

11:12 PM **sniq** it worked

sniq wtf

sniq i just added 1 peashot as you suggested

11:12 PM **jonathankeller** LET'S GO

11:12 PM **sniq** game crashed

sniq let's see the crash

11:13 PM **jonathankeller** heck yes I got it too

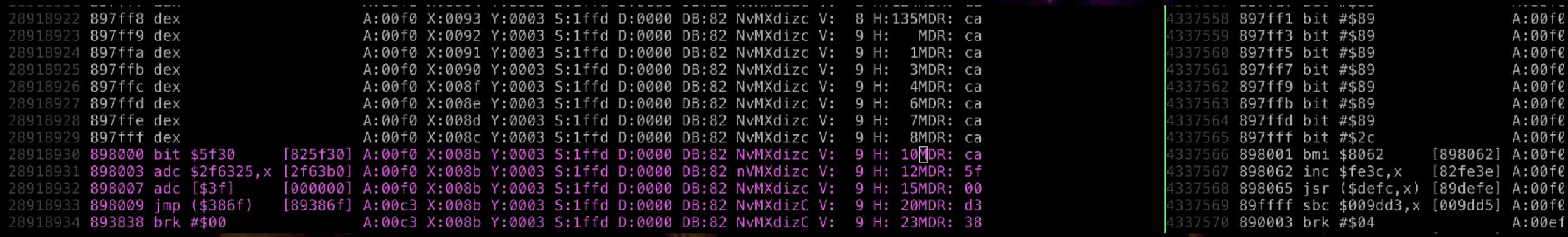
11:13 PM **sniq**

```
897ff5 bit #$89              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H:  66
897ff7 bit #$89              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H:  82
897ff9 bit #$89              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H:  98
897ffb bit #$89              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H: 114
897ffd bit #$89              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H: 130
897fff bit #$2c              A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H: 146
898001 bmi $8062     [898062] A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H: 160
898062 inc $fe3c,x   [82fe3e] A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 NvMXdizc V:268 H: 178
898065 jsr ($defc,x) [89defe] A:00f0 X:0002 Y:0003 S:1ffd D:0000 DB:82 nvMXdiZc V:268 H: 220
89ffff sbc $009dd3,x [009dd5] A:00f0 X:0002 Y:0003 S:1ffb D:0000 DB:82 nvMXdiZc V:268 H: 272
890003 brk #$04              A:00ef X:0002 Y:0003 S:1ffb D:0000 DB:82 NvMXdizC V:268 H: 310
008573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 374
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 406
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 430
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 454
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 478
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 502
808573 jml $808573   [808573] A:00ef X:0002 Y:0003 S:1ff7 D:0000 DB:82 NvMXdIzC V:268 H: 526
```

sniq not far enough yet

jonathankeller ~~stupid 6502 where opcode 00 raises a software interrupt~~

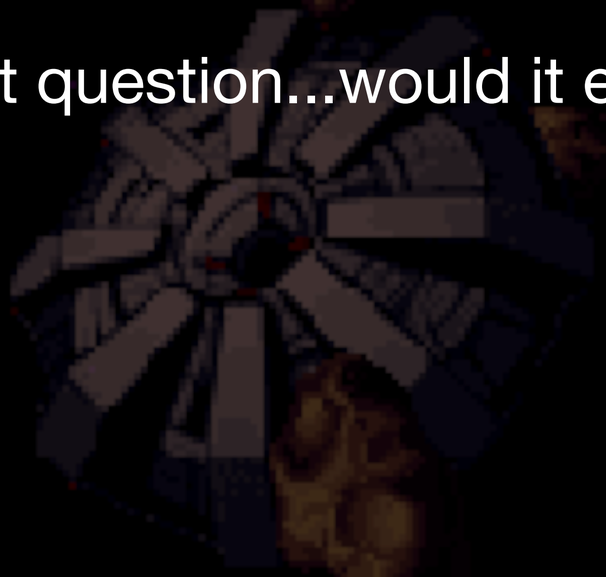
Riding the Open Bus



- If we can execute address $8000 instead of $8001, we land at $3838
- This is in the middle of MMIO space...$4218 is the controller input register
- How can we influence instructions executed from open bus?
- If only some hardware could suddenly use the bus for some reason..

Bus Manipulation with HDMA

- An HDMA transfer is scheduled for the start of the next frame -- about 7500 CPU clock cycles away, or 2.8ms
- Can we get the crash to happen 2.8ms later?
- Before we answer that question...would it even work?



Executing I/O Registers

- Most of the I/O space is unmapped or write-only
- Some input devices don't use all 8 data lines (partial open bus)
- We're starting at address $3838, **controller input is stored at $4218**
- If we can get past these 10 readable registers, we have ACE

Address	Value
$4016	`.... ..01`
$4017	`...1 1101`
$4210	`0... 0010`
$4211	`0... ....`
$4212	`0H.. ...0`
$4213	`0000 0000`
$4214	`0000 1001`
$4215	`0000 0000`
$4216	`0000 0101`
$4217	`0000 0000`

Executing I/O Registers

- Several interrupts and HDMA's occur; when we reach $4016, the bus has $ca (DEX)
- $4016-4017 reads c9 dd (CMP #dd)

Address	Value
$4016	`.... ..01`
$4017	`...1 1101`
$4210	`0... 0010`
$4211	`0... ....`
$4212	`0H.. ...0`
$4213	`0000 0000`
$4214	`0000 1001`
$4215	`0000 0000`
$4216	`0000 0101`
$4217	`0000 0000`

Executing I/O Registers

- $4018-401A reads dd dd dd (CMP $dddd,x)
  - Reads from an address dependent on the value of X
  - The number of DEX instructions we executed earlier is highly significant!
  - With our setup, X is $cb, so we read address $82DEA8 (value $8d)
- $401B reads 8d 8d 8d (STA $8d8d)
  - Writes the A register to a ROM address, which puts the value of A on the bus ($c3)
- $401E reads c3 c3 (CMP $c3, S -- this instruction reads from open bus and does not change the value on the bus, so we're safe for now)
- Eventually another HDMA triggers and puts $ca back on the bus

Address	Value
$4016	`.... ..01`
$4017	`...1 1101`
$4210	`0... 0010`
$4211	`0... ....`
$4212	`0H.. ...0`
$4213	`0000 0000`
$4214	`0000 1001`
$4215	`0000 0000`
$4216	`0000 0101`
$4217	`0000 0000`

Executing I/O Registers

- $4210-4211 reads 42 42 (WDM #42)
- 2-byte NOP instruction, named after the processor's designer: William D. Mensch, Jr
- Leaves $42 on the bus

Address	Value
$4016	01
$4017	...1 1101
$4210	0... 0010
$4211	0... ....
$4212	0H.. ...0
$4213	0000 0000
$4214	0000 1001
$4215	0000 0000
$4216	0000 0101
$4217	0000 0000

Executing I/O Registers

- This one's a pain...
- If we read outside of blanking, we get $02 (**COP** - coprocessor interrupt)
- If we read during horizontal blanking, we get $42 (WDM)
- H-blank is a 12.8µs window (~34 CPU clocks)
- *Much* easier to come in aligned to odd addresses and treat $4212 as an operand, but this does not work later on

Address	Value
$4016	01
$4017	...1 1101
$4210	0... 0010
$4211	0... ....
$4212	0H.. ...0
$4213	
$4214	
$4215	
$4216	
$4217	

HVBJOY - Screen and Joypad status ($4212 read)

```
7  bit  0
---- ----
VHxx xxxJ
|||| ||||
|||| |||+- Joypad auto-read in-progress flag
||++-+++-- (Open bus)
|+-------- Hblank flag
+--------- Vblank flag
```

- J - Set during joypad auto-read.
- H - Set during horizontal blank period.
- V - Set during vertical blank period.

Executing I/O Registers

- $4213-4217 contain results from the hardware divider, last used to compute the coordinates of this E-tank:

- Need to execute 2-byte opcodes at $4212, $4214, and $4217

Address	Value
$4016	`.... ..01`
$4017	`...1 1101`
$4210	`0... 0010`
$4211	`0... ....`
$4212	`0H.. ...0`
$4213	`0000 0000`
$4214	`0000 1001`
$4215	`0000 0000`
$4216	`0000 0101`
$4217	`0000 0000`

Executing I/O Registers

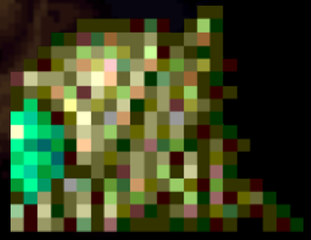
- *Finally* we made it to $4218: controller port input!
- Press buttons on the controller corresponding to our payload
- Fix up the stack (push a return address within the game's main loop, and a processor status byte)
- Jump into a convenient spot in the routine that sets up the ending cutscene

```
F4 45 89      PEA $8945
08            PHP
5C D3 84 82   JML $8284D3
```

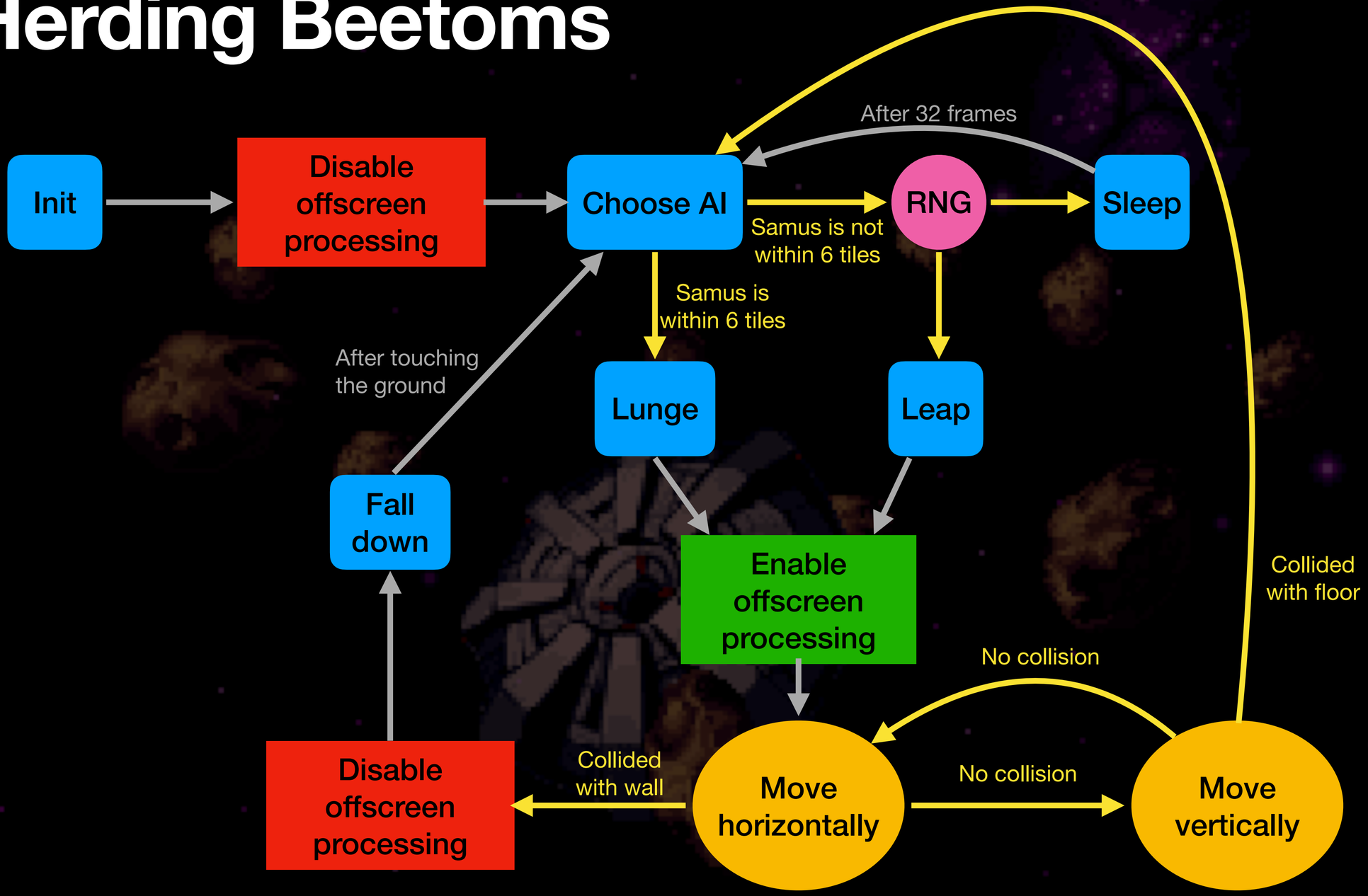
```
;;; $84BD: Game state 26h (Samus escapes from Zebes) ;;;
{
$82:84BD 08         PHP
$82:84BE C2 30      REP #$30
$82:84C0 20 44 8B   JSR $8B44  [$82:8B44]  ; Main gameplay
$82:84C3 22 24 89 80 JSL $808924[$80:8924]  ; Handle fading out
$82:84C7 E2 20      SEP #$20
$82:84C9 A5 51      LDA $51    [$7E:0051]  ;\
$82:84CB C9 80      CMP #$80               ;} If not finished fading out:
$82:84CD F0 04      BEQ $84D3  [$84D3]     ;/
$82:84CF C2 20      REP #$20               ; >_<;
$82:84D1 28         PLP
$82:84D2 60         RTS                    ; Return

$82:84D3 22 4B 83 80 JSL $80834B[$80:834B]  ; Enable NMI
$82:84D7 C2 20      REP #$20
$82:84D9 22 9E 82 88 JSL $88829E[$88:829E]  ; Wait until the end of a v-blank and clear (H)DMA enable flags
$82:84DD 22 5F 98 80 JSL $80985F[$80:985F]  ; Disable h/v-counter interrupts
$82:84E1 9C 82 19   STZ $1982  [$7E:1982]  ; Default layer blending configuration = 0
$82:84E4 E2 20      SEP #$20
$82:84E6 64 6E      STZ $6E    [$7E:006E]  ;\
$82:84E8 64 71      STZ $71    [$7E:0071]  ;} Disable colour math
$82:84EA A9 10      LDA #$10               ;\
$82:84EC 85 69      STA $69    [$7E:0069]  ;} Main screen layers = sprites
$82:84EE 64 6B      STZ $6B    [$7E:006B]  ; Disable subscreen layers
$82:84F0 64 6C      STZ $6C    [$7E:006C]  ;\
$82:84F2 64 6D      STZ $6D    [$7E:006D]  ;} Enable all layers in window area
$82:84F4 A9 09      LDA #$09               ;\
$82:84F6 85 55      STA $55    [$7E:0055]  ;} Use mode 1 with BG3 priority and 8x8 tile sizes
$82:84F8 C2 20      REP #$20
$82:84FA 9C 23 07   STZ $0723  [$7E:0723]  ; Screen fade delay = 0
$82:84FD 9C 25 07   STZ $0725  [$7E:0725]  ; Screen fade counter = 0
$82:8500 A9 27 00   LDA #$0027             ;\
$82:8503 8D 98 09   STA $0998  [$7E:0998]  ;} Game state = 27h (ending and credits)
$82:8506 A9 80 D4   LDA #$D480             ;\
$82:8509 8D 51 1F   STA $1F51  [$7E:1F51]  ;} Cinematic function = $D480
$82:850C 9C 43 09   STZ $0943  [$7E:0943]  ; Timer status = inactive
$82:850F A9 00 00   LDA #$0000             ;\
$82:8512 22 C1 8F 80 JSL $808FC1[$80:8FC1]  ;} Queue music stop
$82:8516 A9 02 00   LDA #$0002             ;\
$82:8519 22 21 90 80 JSL $809021[$80:9021]  ;} Queue sound 2, sound library 1, max queued sounds allowed = 15 (silence)
$82:851D A9 71 00   LDA #$0071             ;\
$82:8520 22 A3 90 80 JSL $8090A3[$80:90A3]  ;} Queue sound 71h, sound library 2, max queued sounds allowed = 15 (silence)
$82:8524 A9 01 00   LDA #$0001             ;\
$82:8527 22 25 91 80 JSL $809125[$80:9125]  ;} Queue sound 1, sound library 3, max queued sounds allowed = 15 (silence)
$82:852B 28         PLP
$82:852C 60         RTS
}
```

Doing It for Real

- *If* we get an HDMA to trigger at the right time, we theoretically win the game
- But the crash occurs 2.8ms too early -- can we manipulate things so the game has 2.8ms more processing to do?
- *Much* more precise than it looks -- HBlank requirement, number of DEX instructions, timing of further HDMAs and IRQs means the window of time to trigger the crash is only a handful of clock cycles
- 2.8ms is a *lot* of extra processing, what could use up that much CPU time?



Herding Beetoms



Aside

I don't like beetoms very much

jonathankeller also the beetom happens to be in a really bad spot -- i need to fire an upwards shot on the frame of the crash, but he blocks it

9:33 PM jonathankeller i couldn't get the ACE to work at all with the dropless setup because of the lack of bug onscreen, and I couldn't edit it to pick up a without it ending up slower than the other setup (beetom was troublesome) (edited)

12:14 AM sniq iirc at least 1 second saved so far might be 2

12:20 AM jonathankeller pog

jonathankeller that's another second or two for silly beetom RNG manips, if that's what this comes down to

sniq in that case beetom can't be adjusted to jump right without timeloss

sniq so it isn't possible to leave the 1st beetom jumping left..

@sniq *Click to see attachment*

jonathankeller i haven't been able to get this one to work (it's *sooo close*)

jonathankeller it might be because the beetom is falling instead of jumping at the time of the crash

jonathankeller beetoms just make it very hard

12:24 PM jonathankeller 2 beetoms on screen would mean it doesn't crash because we don't have th

jonathankeller there's so many conditions -_-

jonathankeller now i'm envious of SM64 tasers having a "lock the camera" button

12:28 PM jonathankeller i already have a bunch of stalls (including a pause) for decent RNG, and starting the manips at slightly different times means the beetoms behave completely differently

@sniq *Click to see attachment*

jonathankeller this is really unlucky: beetom #2 is in its 2 frame window between jumps on the crash frame

jonathankeller right now beetom #3 falls down and deactivates

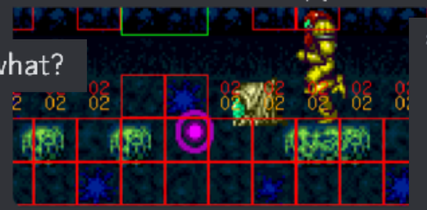
have to dodge beetoms

1:39 PM jonathankeller i don't believe it

jonathankeller the beetom is STILL in the way

the frame the beetom appears on screen is highly significant

sniq 3 beetoms in screen = rip no matter what?



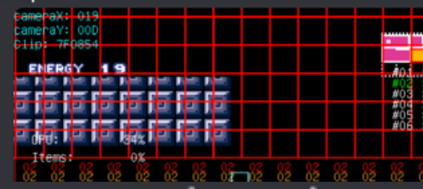
sniq how the hell is this beetom chilling midair

sniq magician for sure

2:07 PM sniq I think I did the magic

sniq I beat the 1st beetom barely

sniq



beetom was troublesome)

the beetom torture chamber

sniq gosh dang it there's a 2nd beetom on screen just barely

sniq but it should be fixable

sniq nevermind, the 4th beetom doesn't even wake up

jonathankeller we have made beetoms the final boss of SM

sniq it jumps right now and doesn't collide with a wa

2:18 PM sniq yup

sniq this will destroy the whole game

sniq current ace is 18406

12:01 PM jonathankeller as long as the beetom doesn't block the shot

2:15 PM sniq ok

sniq beetom blocks the shot

jonathankeller this is the only way I was ever able to get a beetom to go into the right spot on the pipe

jonathankeller i was wondering if that could work with the old setup by messing around with the timing of when we pick up the drop, but the beetom jumping around might make that impossible

@sniq i want the strategy to avoid boosting that far right

the beetom's drawing code writes a bad value to $0000

er the trouble I had was getting the beetom to activate

Bus Manipulation with HDMA and Beetoms

List of Requirements

- We need to burn 2.8 milliseconds, while having an exact number of sprites on screen (so we jump to the right spot in that unrolled loop)
- At least 4 of the 5 beetoms must be jumping
  - Exactly one beetom must be on screen
  - Three beetoms processing offscreen, must never touch a wall
  - Beetoms must never touch Samus
- Exactly one bug must be on screen
- A shot must be exploding on screen
- Screen must be fading out for a pause
- A shot must be fired the frame before the crash
- Item-select must be pressed the frame before the crash
- Samus must be drawn close to the right side of the screen

Current frame: 17937 of 18640

	0-1		1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
17937	FR	0	0	
17938	FR	0	0	BYsS↑↓←→AXLR0123
17939	FR	0	0	BYsS↑↓←→AXLR0123
17940	FR	0	0	BYsS↑↓←→AXLR0123
17941	FR	0	0	BYsS↑↓←→AXLR0123
17942	FR	0	0	BYsS↑↓←→AXLR0123
17943	FR	0	0	BYsS↑↓←→AXLR0123
17944	FR	0	0	BYsS↑↓←→AXLR0123
17945	FR	0	0	BYsS↑↓←→AXLR0123
17946	FR	0	0	BYsS↑↓←→AXLR0123
17947	FR	0	0	BYsS↑↓←→AXLR0123
17948	FR	0	0	BYsS↑↓←→AXLR0123
17949	FR	0	0	BYsS↑↓←→AXLR0123
17950	FR	0	0	BYsS↑↓←→AXLR0123
17951	FR	0	0	BYsS↑↓←→AXLR0123
17952	FR	0	0	BYsS↑↓←→AXLR0123
17953	FR	0	0	BYsS↑↓←→AXLR0123
17954	FR	0	0	BYsS↑↓←→AXLR0123
17955	FR	0	0	BYsS↑↓←→AXLR0123
17956	FR	0	0	BYsS↑↓←→AXLR0123
17957	FR	0	0	BYsS↑↓←→AXLR0123
17958	FR	0	0	BYsS↑↓←→AXLR0123
17959	FR	0	0	BYsS↑↓←→AXLR0123
17960	FR	0	0	BYsS↑↓←→AXLR0123
17961	FR	0	0	BYsS↑↓←→AXLR0123
17962	FR	0	0	BYsS↑↓←→AXLR0123
17963	FR	0	0	BYsS↑↓←→AXLR0123
17964	FR	0	0	BYsS↑↓←→AXLR0123
17965	FR	0	0	BYsS↑↓←→AXLR0123
17966	FR	0	0	BYsS↑↓←→AXLR0123
17967	FR	0	0	BYsS↑↓←→AXLR0123
17968	FR	0	0	BYsS↑↓←→AXLR0123
17969	FR	0	0	BYsS↑↓←→AXLR0123
17970	FR	0	0	BYsS↑↓←→AXLR0123
17971	FR	0	0	BYsS↑↓←→AXLR0123
17972	FR	0	0	BYsS↑↓←→AXLR0123
17973	FR	0	0	BYsS↑↓←→AXLR0123
17974	FR	0	0	BYsS↑↓←→AXLR0123
17975	FR	0	0	BYsS↑↓←→AXLR0123
17976	FR	0	0	BYsS↑↓←→AXLR0123
17977	FR	0	0	BYsS↑↓←→AXLR0123
17978	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	4000
0012	89B0
Controller input	0600
Game mode	8
Movement handler	A337
RNG	AEFA
Samus animation frame	0001
Samus animation timer	13
Samus iframes	0
Samus knockback	0
Samus movement handler	A337
Samus pose	0018
Samus top-half sprite	0274
Shot cooldown	0
Sprite stack	000
X (px)	117
X (sub)	0
X momentum (px)	0
X momentum (sub)	0
X speed (px)	0
X speed (sub)	0
Y (px)	366
Y (sub)	45055
Y speed (px)	6
Y speed (sub)	6144
beetom #1 flags	20
beetom #1 state	B824
beetom #2 flags	20
beetom #2 state	B824
beetom #3 flags	20
beetom #3 state	B824
beetom #4 flags	20
beetom #4 state	B824
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015238
P1 -------→--------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18198 of 18640

	0-1		1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18198	FR	0	0	
18199	FR	0	0	BYsS↑↓←→AXLR0123
18200	FR	0	0	BYsS↑↓←→AXLR0123
18201	FR	0	0	BYsS↑↓←→AXLR0123
18202	FR	0	0	BYsS↑↓←→AXLR0123
18203	FR	0	0	BYsS↑↓←→AXLR0123
18204	FR	0	0	BYsS↑↓←→AXLR0123
18205	FR	0	0	BYsS↑↓←→AXLR0123
18206	FR	0	0	BYsS↑↓←→AXLR0123
18207	FR	0	0	BYsS↑↓←→AXLR0123
18208	FR	0	0	BYsS↑↓←→AXLR0123
18209	FR	0	0	BYsS↑↓←→AXLR0123
18210	FR	0	0	BYsS↑↓←→AXLR0123
18211	FR	0	0	BYsS↑↓←→AXLR0123
18212	FR	0	0	BYsS↑↓←→AXLR0123
18213	FR	0	0	BYsS↑↓←→AXLR0123
18214	FR	0	0	BYsS↑↓←→AXLR0123
18215	FR	0	0	BYsS↑↓←→AXLR0123
18216	FR	0	0	BYsS↑↓←→AXLR0123
18217	FR	0	0	BYsS↑↓←→AXLR0123
18218	FR	0	0	BYsS↑↓←→AXLR0123
18219	FR	0	0	BYsS↑↓←→AXLR0123
18220	FR	0	0	BYsS↑↓←→AXLR0123
18221	FR	0	0	BYsS↑↓←→AXLR0123
18222	FR	0	0	BYsS↑↓←→AXLR0123
18223	FR	0	0	BYsS↑↓←→AXLR0123
18224	FR	0	0	BYsS↑↓←→AXLR0123
18225	FR	0	0	BYsS↑↓←→AXLR0123
18226	FR	0	0	BYsS↑↓←→AXLR0123
18227	FR	0	0	BYsS↑↓←→AXLR0123
18228	FR	0	0	BYsS↑↓←→AXLR0123
18229	FR	0	0	BYsS↑↓←→AXLR0123
18230	FR	0	0	BYsS↑↓←→AXLR0123
18231	FR	0	0	BYsS↑↓←→AXLR0123
18232	FR	0	0	BYsS↑↓←→AXLR0123
18233	FR	0	0	BYsS↑↓←→AXLR0123
18234	FR	0	0	BYsS↑↓←→AXLR0123
18235	FR	0	0	BYsS↑↓←→AXLR0123
18236	FR	0	0	BYsS↑↓←→AXLR0123
18237	FR	0	0	BYsS↑↓←→AXLR0123
18238	FR	0	0	BYsS↑↓←→AXLR0123
18239	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89A1
Controller input	0000
Game mode	8
Movement handler	A337
RNG	4D9E
Samus animation frame	0003
Samus animation timer	1
Samus iframes	13
Samus knockback	0
Samus movement handler	A337
Samus pose	0019
Samus top-half sprite	074F
Shot cooldown	0
Sprite stack	000
X (px)	379
X (sub)	54271
X momentum (px)	1
X momentum (sub)	42496
X speed (px)	0
X speed (sub)	0
Y (px)	446
Y (sub)	22527
Y speed (px)	0
Y speed (sub)	61440
beetom #1 flags	28
beetom #1 state	B83F
beetom #2 flags	20
beetom #2 state	B824
beetom #3 flags	20
beetom #3 state	B824
beetom #4 flags	20
beetom #4 state	B824
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015243
P1 ----------------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18240 of 18640

	0-1		1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18240	FR	0	0	BYsS↑↓←→A
18241	FR	0	0	BYsS↑↓←→AXLR0123
18242	FR	0	0	BYsS↑↓←→AXLR0123
18243	FR	0	0	BYsS↑↓←→AXLR0123
18244	FR	0	0	BYsS↑↓←→AXLR0123
18245	FR	0	0	BYsS↑↓←→AXLR0123
18246	FR	0	0	BYsS↑↓←→AXLR0123
18247	FR	0	0	BYsS↑↓←→AXLR0123
18248	FR	0	0	BYsS↑↓←→AXLR0123
18249	FR	0	0	BYsS↑↓←→AXLR0123
18250	FR	0	0	BYsS↑↓←→AXLR0123
18251	FR	0	0	BYsS↑↓←→AXLR0123
18252	FR	0	0	BYsS↑↓←→AXLR0123
18253	FR	0	0	BYsS↑↓←→AXLR0123
18254	FR	0	0	BYsS↑↓←→AXLR0123
18255	FR	0	0	BYsS↑↓←→AXLR0123
18256	FR	0	0	BYsS↑↓←→AXLR0123
18257	FR	0	0	BYsS↑↓←→AXLR0123
18258	FR	0	0	BYsS↑↓←→AXLR0123
18259	FR	0	0	BYsS↑↓←→AXLR0123
18260	FR	0	0	BYsS↑↓←→AXLR0123
18261	FR	0	0	BYsS↑↓←→AXLR0123
18262	FR	0	0	BYsS↑↓←→AXLR0123
18263	FR	0	0	BYsS↑↓←→AXLR0123
18264	FR	0	0	BYsS↑↓←→AXLR0123
18265	FR	0	0	BYsS↑↓←→AXLR0123
18266	FR	0	0	BYsS↑↓←→AXLR0123
18267	FR	0	0	BYsS↑↓←→AXLR0123
18268	FR	0	0	BYsS↑↓←→AXLR0123
18269	FR	0	0	BYsS↑↓←→AXLR0123
18270	FR	0	0	BYsS↑↓←→AXLR0123
18271	FR	0	0	BYsS↑↓←→AXLR0123
18272	FR	0	0	BYsS↑↓←→AXLR0123
18273	FR	0	0	BYsS↑↓←→AXLR0123
18274	FR	0	0	BYsS↑↓←→AXLR0123
18275	FR	0	0	BYsS↑↓←→AXLR0123
18276	FR	0	0	BYsS↑↓←→AXLR0123
18277	FR	0	0	BYsS↑↓←→AXLR0123
18278	FR	0	0	BYsS↑↓←→AXLR0123
18279	FR	0	0	BYsS↑↓←→AXLR0123
18280	FR	0	0	BYsS↑↓←→AXLR0123
18281	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89A1
Controller input	0180
Game mode	8
Movement handler	A337
RNG	1EED
Samus animation frame	0003
Samus animation timer	1
Samus iframes	45
Samus knockback	0
Samus movement handler	A337
Samus pose	004F
Samus top-half sprite	029A
Shot cooldown	0
Sprite stack	000
X (px)	498
X (sub)	60927
X momentum (px)	6
X momentum (sub)	0
X speed (px)	0
X speed (sub)	0
Y (px)	324
Y (sub)	6144
Y speed (px)	0
Y speed (sub)	40960
beetom #1 flags	28
beetom #1 state	BC66
beetom #2 flags	20
beetom #2 state	B824
beetom #3 flags	20
beetom #3 state	B824
beetom #4 flags	20
beetom #4 state	B824
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015244
P1 -------→A-------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18287 of 18640

	0-1		1-1
	FR	rhigh rlow	BYsS↑↓←→AXLR0123
18288	FR	0 0	BYsS↑↓←→AXLR0123
18289	FR	0 0	BYsS↑↓←→AXLR0123
18290	FR	0 0	BYsS↑↓←→AXLR0123
18291	FR	0 0	BYsS↑↓←→AXLR0123
18292	FR	0 0	BYsS↑↓←→AXLR0123
18293	FR	0 0	BYsS↑↓←AXLR0123
18294	FR	0 0	BYsS↑↓←→AXLR0123
18295	FR	0 0	BYsS↑↓←→AXLR0123
18296	FR	0 0	BYsS↑↓←→AXLR0123
18297	FR	0 0	BYsS↑↓←→AXLR0123
18298	FR	0 0	BYsS↑↓←→AXLR0123
18299	FR	0 0	BYsS↑↓←→AXLR0123
18300	FR	0 0	BYsS↑↓←→AXLR0123
18301	FR	0 0	BYsS↑↓←→AXLR0123
18302	FR	0 0	BYsS↑↓←→AXLR0123
18303	FR	0 0	BYsS↑↓←→AXLR0123
18304	FR	0 0	BYsS↑↓←→AXLR0123
18305	FR	0 0	BYsS↑↓←→AXLR0123
18306	FR	0 0	BYsS↑↓←→AXLR0123
18307	FR	0 0	BYsS↑↓←→AXLR0123
18308	FR	0 0	BYsS↑↓←→AXLR0123
18309	FR	0 0	BYsS↑↓←→AXLR0123
18310	FR	0 0	BYsS↑↓←→AXLR0123
18311	FR	0 0	BYsS↑↓←→AXLR0123
18312	FR	0 0	BYsS↑↓←→AXLR0123
18313	FR	0 0	BYsS↑↓←→AXLR0123
18314	FR	0 0	BYsS↑↓←→AXLR0123
18315	FR	0 0	BYsS↑↓←→AXLR0123
18316	FR	0 0	BYsS↑↓←→AXLR0123
18317	FR	0 0	BYsS↑↓←→AXLR0123
18318	FR	0 0	BYsS↑↓←→AXLR0123
18319	FR	0 0	BYsS↑↓←→AXLR0123
18320	FR	0 0	BYsS↑↓←→AXLR0123
18321	FR	0 0	BYsS↑↓←→AXLR0123
18322	FR	0 0	BYsS↑↓←→AXLR0123
18323	FR	0 0	BYsS↑↓←→AXLR0123
18324	FR	0 0	BYsS↑↓←→AXLR0123
18325	FR	0 0	BYsS↑↓←→AXLR0123
18326	FR	0 0	BYsS↑↓←→AXLR0123
18327	FR	0 0	BYsS↑↓←→AXLR0123
18328	FR	0 0	BYsS↑↓←→AXLR0123
18329	FR	0 0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89CB
Controller input	8180
Game mode	8
Movement handler	A337
RNG	8FED
Samus animation frame	0004
Samus animation timer	2
Samus iframes	0
Samus knockback	0
Samus movement handler	A337
Samus pose	0019
Samus top-half sprite	0750
Shot cooldown	1
Sprite stack	000
X (px)	727
X (sub)	44287
X momentum (px)	1
X momentum (sub)	42496
X speed (px)	0
X speed (sub)	30720
Y (px)	364
Y (sub)	65279
Y speed (px)	4
Y speed (sub)	28928
beetom #1 flags	28
beetom #1 state	BB98
beetom #2 flags	28
beetom #2 state	BB65
beetom #3 flags	20
beetom #3 state	B9C2
beetom #4 flags	20
beetom #4 state	B824
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015245
P1 B------→A-------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18314 of 18640

	0-1		1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18314	FR	0	0	BYsS↑↓←→AXLR0123
18315	FR	0	0	BYsS↑↓←→AXLR0123
18316	FR	0	0	BYsS↑↓←→AXLR0123
18317	FR	0	0	BYsS↑↓←→AXLR0123
18318	FR	0	0	BYsS↑↓→AXLR0123
18319	FR	0	0	BYsS↑↓←→AXLR0123
18320	FR	0	0	BYsS↑↓→AXLR0123
18321	FR	0	0	BYsS↑↓→AXLR0123
18322	FR	0	0	BYsS↑↓←→AXLR0123
18323	FR	0	0	BYsS↑↓←→AXLR0123
18324	FR	0	0	BYsS↑↓→AXLR0123
18325	FR	0	0	BYsS↑↓→AXLR0123
18326	FR	0	0	BYsS↑↓←→AXLR0123
18327	FR	0	0	BYsS↑↓←→AXLR0123
18328	FR	0	0	BYsS↑↓←→AXLR0123
18329	FR	0	0	BYsS↑↓←→AXLR0123
18330	FR	0	0	BYsS↑↓←→AXLR0123
18331	FR	0	0	BYsS↑↓←→AXLR0123
18332	FR	0	0	BYsS↑↓←→AXLR0123
18333	FR	0	0	BYsS↑↓←→AXLR0123
18334	FR	0	0	BYsS↑↓←→AXLR0123
18335	FR	0	0	BYsS↑↓←→AXLR0123
18336	FR	0	0	BYsS↑↓←→AXLR0123
18337	FR	0	0	BYsS↑↓←→AXLR0123
18338	FR	0	0	BYsS↑↓←→AXLR0123
18339	FR	0	0	BYsS↑↓→AXLR0123
18340	FR	0	0	BYsS↑↓→AXLR0123
18341	FR	0	0	BYsS↑↓→AXLR0123
18342	FR	0	0	BYsS↑↓→AXLR0123
18343	FR	0	0	BYsS↑↓→AXLR0123
18344	FR	0	0	BYsS↑↓→AXLR0123
18345	FR	0	0	BYsS↑↓←→AXLR0123
18346	FR	0	0	BYsS↑↓←→AXLR0123
18347	FR	0	0	BYsS↑↓←→AXLR0123
18348	FR	0	0	BYsS↑↓←→AXLR0123
18349	FR	0	0	BYsS↑↓←→AXLR0123
18350	FR	0	0	BYsS↑↓←→AXLR0123
18351	FR	0	0	BYsS↑↓→AXLR0123
18352	FR	0	0	BYsS↑↓←→AXLR0123
18353	FR	0	0	BYsS↑↓←→AXLR0123
18354	FR	0	0	BYsS↑↓←→AXLR0123
18355	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89C8
Controller input	0200
Game mode	8
Movement handler	A337
RNG	6D96
Samus animation frame	0000
Samus animation timer	1
Samus iframes	43
Samus knockback	0
Samus movement handler	A337
Samus pose	0052
Samus top-half sprite	0291
Shot cooldown	0
Sprite stack	000
X (px)	826
X (sub)	13823
X momentum (px)	0
X momentum (sub)	58880
X speed (px)	0
X speed (sub)	0
Y (px)	408
Y (sub)	4863
Y speed (px)	0
Y speed (sub)	61440
beetom #1 flags	28
beetom #1 state	BB65
beetom #2 flags	28
beetom #2 state	BB65
beetom #3 flags	20
beetom #3 state	B9C2
beetom #4 flags	20
beetom #4 state	B8DB
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015246
P1 ------←---------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18340 of 18640

	0-1			1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18340	FR	0	0	B→
18341	FR	0	0	BYsS↑↓←→AXLR0123
18342	FR	0	0	BYsS↑↓←→AXLR0123
18343	FR	0	0	BYsS↑↓←→AXLR0123
18344	FR	0	0	BYsS↑↓←→AXLR0123
18345	FR	0	0	BYsS↑↓←→AXLR0123
18346	FR	0	0	BYsS↑↓←→AXLR0123
18347	FR	0	0	BYsS↑↓←→AXLR0123
18348	FR	0	0	BYsS↑↓←→AXLR0123
18349	FR	0	0	BYsS↑↓←→AXLR0123
18350	FR	0	0	BYsS↑↓←→AXLR0123
18351	FR	0	0	BYsS↑↓←→AXLR0123
18352	FR	0	0	BYsS↑↓←→AXLR0123
18353	FR	0	0	BYsS↑↓←→AXLR0123
18354	FR	0	0	BYsS↑↓←→AXLR0123
18355	FR	0	0	BYsS↑↓←→AXLR0123
18356	FR	0	0	BYsS↑↓←→AXLR0123
18357	FR	0	0	BYsS↑↓←→AXLR0123
18358	FR	0	0	BYsS↑↓←→AXLR0123
18359	FR	0	0	BYsS↑↓←→AXLR0123
18360	FR	0	0	BYsS↑↓←→AXLR0123
18361	FR	0	0	BYsS↑↓←→AXLR0123
18362	FR	0	0	BYsS↑↓←→AXLR0123
18363	FR	0	0	BYsS↑↓←→AXLR0123
18364	FR	0	0	BYsS↑↓←→AXLR0123
18365	FR	0	0	BYsS↑↓←→AXLR0123
18366	FR	0	0	BYsS↑↓←→AXLR0123
18367	FR	0	0	BYsS↑↓←→AXLR0123
18368	FR	0	0	BYsS↑↓←→AXLR0123
18369	FR	0	0	BYsS↑↓←→AXLR0123
18370	FR	0	0	BYsS↑↓←→AXLR0123
18371	FR	0	0	BYsS↑↓←→AXLR0123
18372	FR	0	0	BYsS↑↓←→AXLR0123
18373	FR	0	0	BYsS↑↓←→AXLR0123
18374	FR	0	0	BYsS↑↓←→AXLR0123
18375	FR	0	0	BYsS↑↓←→AXLR0123
18376	FR	0	0	BYsS↑↓←→AXLR0123
18377	FR	0	0	BYsS↑↓←→AXLR0123
18378	FR	0	0	BYsS↑↓←→AXLR0123
18379	FR	0	0	BYsS↑↓←→AXLR0123
18380	FR	0	0	BYsS↑↓←→AXLR0123
18381	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89B0
Controller input	8200
Game mode	8
Movement handler	A337
RNG	B0BA
Samus animation frame	0007
Samus animation timer	1
Samus iframes	17
Samus knockback	0
Samus movement handler	A337
Samus pose	000A
Samus top-half sprite	020A
Shot cooldown	0
Sprite stack	000
X (px)	790
X (sub)	12287
X momentum (px)	3
X momentum (sub)	17664
X speed (px)	0
X speed (sub)	61440
Y (px)	411
Y (sub)	65535
Y speed (px)	0
Y speed (sub)	0
beetom #1 flags	28
beetom #1 state	BB65
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	28
beetom #3 state	BC36
beetom #4 flags	28
beetom #4 state	BB65
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015246
P1 B------→--------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18348 of 18640

	0-1			1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18348	FR	0	0	←[illegible]
18349	FR	0	0	BYsS↑↓←→AXLR0123
18350	FR	0	0	BYsS↑↓←→AXLR0123
18351	FR	0	0	BYsS↑↓←→AXLR0123
18352	FR	0	0	BYsS↑↓←→AXLR0123
18353	FR	0	0	BYsS↑↓←→AXLR0123
18354	FR	0	0	BYsS↑↓←→AXLR0123
18355	FR	0	0	BYsS↑↓←→AXLR0123
18356	FR	0	0	BYsS↑↓←→AXLR0123
18357	FR	0	0	BYsS↑↓←→AXLR0123
18358	FR	0	0	BYsS↑↓←→AXLR0123
18359	FR	0	0	BYsS↑↓←→AXLR0123
18360	FR	0	0	BYsS↑↓←→AXLR0123
18361	FR	0	0	BYsS↑↓←→AXLR0123
18362	FR	0	0	BYsS↑↓←→AXLR0123
18363	FR	0	0	BYsS↑↓←→AXLR0123
18364	FR	0	0	BYsS↑↓←→AXLR0123
18365	FR	0	0	BYsS↑↓←→AXLR0123
18366	FR	0	0	BYsS↑↓←→AXLR0123
18367	FR	0	0	BYsS↑↓←→AXLR0123
18368	FR	0	0	BYsS↑↓←→AXLR0123
18369	FR	0	0	BYsS↑↓←→AXLR0123
18370	FR	0	0	BYsS↑↓←→AXLR0123
18371	FR	0	0	BYsS↑↓←→AXLR0123
18372	FR	0	0	BYsS↑↓←→AXLR0123
18373	FR	0	0	BYsS↑↓←→AXLR0123
18374	FR	0	0	BYsS↑↓←→AXLR0123
18375	FR	0	0	BYsS↑↓←→AXLR0123
18376	FR	0	0	BYsS↑↓←→AXLR0123
18377	FR	0	0	BYsS↑↓←→AXLR0123
18378	FR	0	0	BYsS↑↓←→AXLR0123
18379	FR	0	0	BYsS↑↓←→AXLR0123
18380	FR	0	0	BYsS↑↓←→AXLR0123
18381	FR	0	0	BYsS↑↓←→AXLR0123
18382	FR	0	0	BYsS↑↓←→AXLR0123
18383	FR	0	0	BYsS↑↓←→AXLR0123
18384	FR	0	0	BYsS↑↓←→AXLR0123
18385	FR	0	0	BYsS↑↓←→AXLR0123
18386	FR	0	0	BYsS↑↓←→AXLR0123
18387	FR	0	0	BYsS↑↓←→AXLR0123
18388	FR	0	0	BYsS↑↓←→AXLR0123
18389	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89A4
Controller input	8200
Game mode	8
Movement handler	A337
RNG	AD4B
Samus animation frame	0001
Samus animation timer	2
Samus iframes	9
Samus knockback	0
Samus movement handler	A337
Samus pose	001A
Samus top-half sprite	074C
Shot cooldown	0
Sprite stack	000
X (px)	772
X (sub)	65535
X momentum (px)	0
X momentum (sub)	8448
X speed (px)	0
X speed (sub)	0
Y (px)	425
Y (sub)	2559
Y speed (px)	0
Y speed (sub)	20480
beetom #1 flags	28
beetom #1 state	BB65
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	28
beetom #3 state	BC36
beetom #4 flags	28
beetom #4 state	B824
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015246
P1 ------←---------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18352 of 18640

	0-1			1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18352	FR	0	0	
18353	FR	0	0	BYsS↑↓←→AXLR0123
18354	FR	0	0	BYsS↑↓←→AXLR0123
18355	FR	0	0	BYsS↑↓←→AXLR0123
18356	FR	0	0	BYsS↑↓←→AXLR0123
18357	FR	0	0	BYsS↑↓←→AXLR0123
18358	FR	0	0	BYsS↑↓→AXLR0123
18359	FR	0	0	BYsS↑↓←→AXLR0123
18360	FR	0	0	BYsS↑↓←→AXLR0123
18361	FR	0	0	BYsS↑↓←AXLR0123
18362	FR	0	0	BYsS↑↓←→AXLR0123
18363	FR	0	0	BYsS↑↓←→AXLR0123
18364	FR	0	0	BYsS↑↓←AXLR0123
18365	FR	0	0	BYsS↑↓→AXLR0123
18366	FR	0	0	BYsS↑↓→AXLR0123
18367	FR	0	0	BYsS↑↓←AXLR0123
18368	FR	0	0	BYsS↑↓←AXLR0123
18369	FR	0	0	BYsS↑↓←→AXLR0123
18370	FR	0	0	BYsS↑↓←→AXLR0123
18371	FR	0	0	BYsS↑↓←→AXLR0123
18372	FR	0	0	BYsS↑↓←→AXLR0123
18373	FR	0	0	BYsS↑↓←→AXLR0123
18374	FR	0	0	BYsS↑↓←→AXLR0123
18375	FR	0	0	BYsS↑↓←AXLR0123
18376	FR	0	0	BYsS↑↓←→AXLR0123
18377	FR	0	0	BYsS↑↓←→AXLR0123
18378	FR	0	0	BYsS↑↓←→AXLR0123
18379	FR	0	0	BYsS↑↓←→AXLR0123
18380	FR	0	0	BYsS↑↓←→AXLR0123
18381	FR	0	0	BYsS↑↓←→AXLR0123
18382	FR	0	0	BYsS↑↓←→AXLR0123
18383	FR	0	0	BYsS↑↓←→AXLR0123
18384	FR	0	0	BYsS↑↓←→AXLR0123
18385	FR	0	0	BYsS↑↓←→AXLR0123
18386	FR	0	0	BYsS↑↓←→AXLR0123
18387	FR	0	0	BYsS↑↓←→AXLR0123
18388	FR	0	0	BYsS↑↓←→AXLR0123
18389	FR	0	0	BYsS↑↓←→AXLR0123
18390	FR	0	0	BYsS↑↓←→AXLR0123
18391	FR	0	0	BYsS↑↓←→AXLR0123
18392	FR	0	0	BYsS↑↓←→AXLR0123
18393	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89AD
Controller input	0000
Game mode	8
Movement handler	A337
RNG	A564
Samus animation frame	000B
Samus animation timer	4
Samus iframes	5
Samus knockback	0
Samus movement handler	A337
Samus pose	001A
Samus top-half sprite	076F
Shot cooldown	0
Sprite stack	000
X (px)	769
X (sub)	46079
X momentum (px)	1
X momentum (sub)	42496
X speed (px)	0
X speed (sub)	0
Y (px)	427
Y (sub)	14847
Y speed (px)	0
Y speed (sub)	61440
beetom #1 flags	28
beetom #1 state	BB65
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	28
beetom #3 state	BC36
beetom #4 flags	28
beetom #4 state	BB65
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015246
P1 ----------------
P2 ----------------
P3 ----------------
P4 ----------------

Current frame: 18375 of 18640

	0-1			1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18375	FR	0	0	A
18376	FR	0	0	BYsS↑↓←→AXLR0123
18377	FR	0	0	BYsS↑↓←→AXLR0123
18378	FR	0	0	BYsS↑↓←→AXLR0123
18379	FR	0	0	BYsS↑↓←→AXLR0123
18380	FR	0	0	BYsS↑↓←→AXLR0123
18381	FR	0	0	BYsS↑↓←→AXLR0123
18382	FR	0	0	BYsS↑↓←→AXLR0123
18383	FR	0	0	BYsS↑↓←→AXLR0123
18384	FR	0	0	BYsS↑↓←→AXLR0123
18385	FR	0	0	BYsS↑↓←→AXLR0123
18386	FR	0	0	BYsS↑↓←→AXLR0123
18387	FR	0	0	BYsS↑↓←→AXLR0123
18388	FR	0	0	BYsS↑↓←→AXLR0123
18389	FR	0	0	BYsS↑↓←→AXLR0123
18390	FR	0	0	BYsS↑↓←→AXLR0123
18391	FR	0	0	BYsS↑↓←→AXLR0123
18392	FR	0	0	BYsS↑↓←→AXLR0123
18393	FR	0	0	BYsS↑↓←→AXLR0123
18394	FR	0	0	BYsS↑↓←→AXLR0123
18395	FR	0	0	BYsS↑↓←→AXLR0123
18396	FR	0	0	BYsS↑↓←→AXLR0123
18397	FR	0	0	BYsS↑↓←→AXLR0123
18398	FR	0	0	BYsS↑↓←→AXLR0123
18399	FR	0	0	BYsS↑↓←→AXLR0123
18400	FR	0	0	BYsS↑↓←→AXLR0123
18401	FR	0	0	BYsS↑↓←→AXLR0123
18402	FR	0	0	BYsS↑↓←→AXLR0123
18403	FR	0	0	BYsS↑↓←→AXLR0123
18404	FR	0	0	BYsS↑↓←→AXLR0123
18405	FR	0	0	BYsS↑↓←→AXLR0123
18406	FR	0	0	BYsS↑↓←→AXLR
18407	FR	0	0	BYsS↑↓←→AXLR0123
18408	FR	0	0	BYsS↑↓←→AXLR0123
18409	FR	0	0	BYsS↑↓←→AXLR0123
18410	FR	0	0	BYsS↑↓←→AXLR0123
18411	FR	0	0	BYsS↑↓←→AXLR0123
18412	FR	0	0	BYsS↑↓←→AXLR0123
18413	FR	0	0	BYsS↑↓←→AXLR0123
18414	FR	0	0	BYsS↑↓←→AXLR0123
18415	FR	0	0	BYsS↑↓←→AXLR0123
18416	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89B9
Controller input	0000
Game mode	8
Movement handler	A337
RNG	F1CC
Samus animation frame	0001
Samus animation timer	2
Samus iframes	51
Samus knockback	1
Samus movement handler	A337
Samus pose	00E5
Samus top-half sprite	01B8
Shot cooldown	0
Sprite stack	000
X (px)	772
X (sub)	26112
X momentum (px)	0
X momentum (sub)	0
X speed (px)	0
X speed (sub)	0
Y (px)	443
Y (sub)	65535
Y speed (px)	0
Y speed (sub)	0
beetom #1 flags	28
beetom #1 state	BB65
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	28
beetom #3 state	BC36
beetom #4 flags	28
beetom #4 state	BB65
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015247
P1 --------A-------
P2 ----------------
P3 ----------------
P4 ----------------

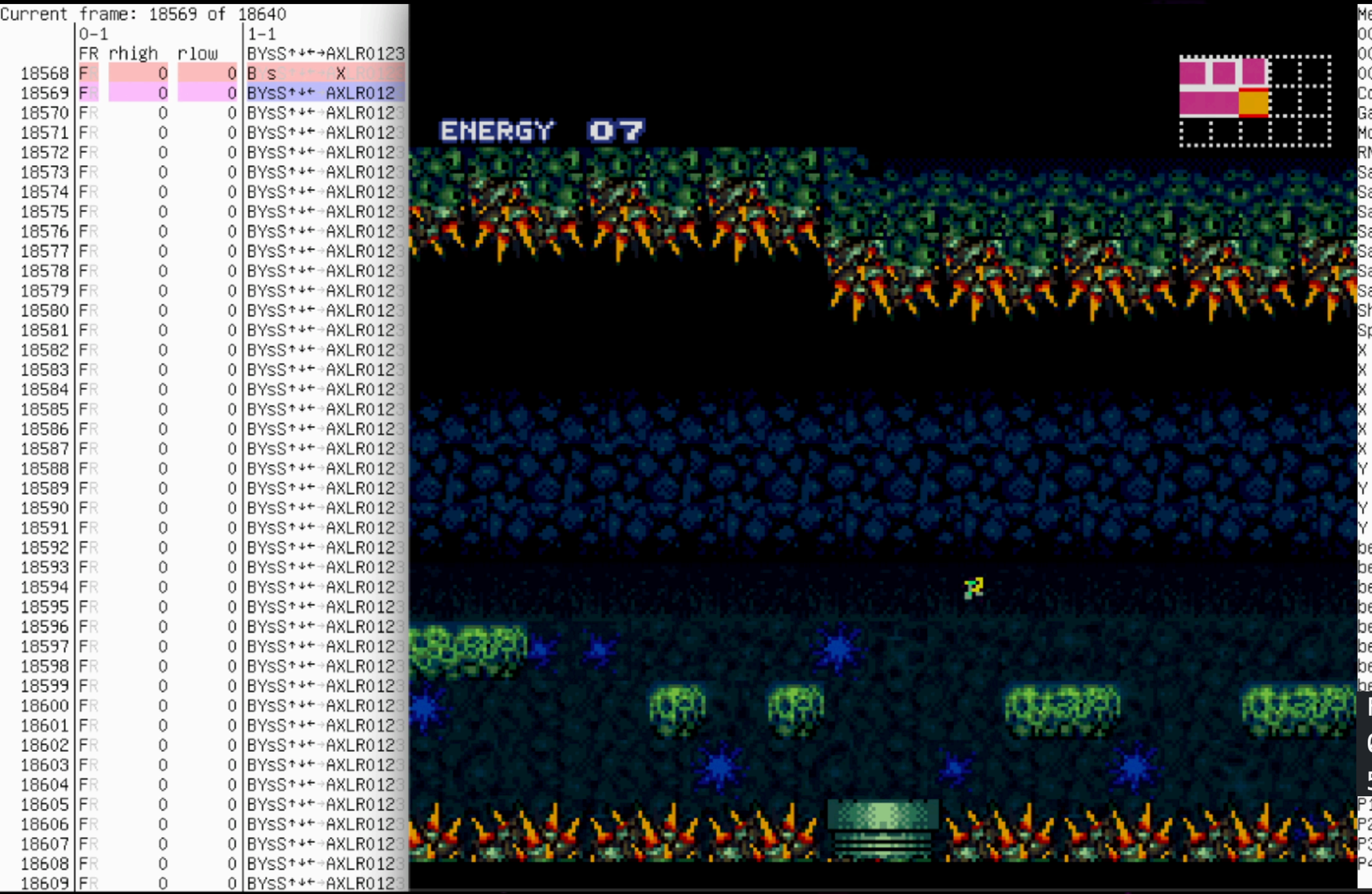
Current frame: 18410 of 18640

	0-1		1-1
	FR	rhigh	rlow	BYsS↑↓←→AXLR0123
18410	FR	0	0	BYsS↑↓←→AXLR0123
18411	FR	0	0	BYsS↑↓←→AXLR0123
18412	FR	0	0	BYsS↑↓←→AXLR0123
18413	FR	0	0	BYsS↑↓←→AXLR0123
18414	FR	0	0	BYsS↑↓←→AXLR0123
18415	FR	0	0	BYsS↑↓←→AXLR0123
18416	FR	0	0	BYsS↑↓←→AXLR0123
18417	FR	0	0	BYsS↑↓←→AXLR0123
18418	FR	0	0	BYsS↑↓←→AXLR0123
18419	FR	0	0	BYsS↑↓←→AXLR0123
18420	FR	0	0	BYsS↑↓←→AXLR0123
18421	FR	0	0	BYsS↑↓←→AXLR0123
18422	FR	0	0	BYsS↑↓←→AXLR0123
18423	FR	0	0	BYsS↑↓←→AXLR0123
18424	FR	0	0	BYsS↑↓←→AXLR0123
18425	FR	0	0	BYsS↑↓←→AXLR0123
18426	FR	0	0	BYsS↑↓←→AXLR0123
18427	FR	0	0	BYsS↑↓←→AXLR0123
18428	FR	0	0	BYsS↑↓←→AXLR0123
18429	FR	0	0	BYsS↑↓←→AXLR0123
18430	FR	0	0	BYsS↑↓←→AXLR0123
18431	FR	0	0	BYsS↑↓←→AXLR0123
18432	FR	0	0	BYsS↑↓←→AXLR0123
18433	FR	0	0	BYsS↑↓←→AXLR0123
18434	FR	0	0	BYsS↑↓←→AXLR0123
18435	FR	0	0	BYsS↑↓←→AXLR0123
18436	FR	0	0	BYsS↑↓←→AXLR0123
18437	FR	0	0	BYsS↑↓←→AXLR0123
18438	FR	0	0	BYsS↑↓←→AXLR0123
18439	FR	0	0	BYsS↑↓←→AXLR0123
18440	FR	0	0	BYsS↑↓←→AXLR0123
18441	FR	0	0	BYsS↑↓←→AXLR0123
18442	FR	0	0	BYsS↑↓←→AXLR0123
18443	FR	0	0	BYsS↑↓←→AXLR0123
18444	FR	0	0	BYsS↑↓←→AXLR0123
18445	FR	0	0	BYsS↑↓←→AXLR0123
18446	FR	0	0	BYsS↑↓←→AXLR0123
18447	FR	0	0	BYsS↑↓←→AXLR0123
18448	FR	0	0	BYsS↑↓←→AXLR0123
18449	FR	0	0	BYsS↑↓←→AXLR0123
18450	FR	0	0	BYsS↑↓←→AXLR0123
18451	FR	0	0	BYsS↑↓←→AXLR0123

Memory watches:

0000	9DD3
0002	0000
0012	89AD
Controller input	0000
Game mode	8
Movement handler	A337
RNG	946E
Samus animation frame	0007
Samus animation timer	7
Samus iframes	16
Samus knockback	0
Samus movement handler	A337
Samus pose	00F2
Samus top-half sprite	01CF
Shot cooldown	0
Sprite stack	000
X (px)	772
X (sub)	26112
X momentum (px)	0
X momentum (sub)	0
X speed (px)	0
X speed (sub)	0
Y (px)	448
Y (sub)	65535
Y speed (px)	0
Y speed (sub)	0
beetom #1 flags	28
beetom #1 state	B9C2
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	28
beetom #3 state	BC66
beetom #4 flags	28
beetom #4 state	B9C2
beetom #5 flags	20
beetom #5 state	B824

Status:
RTC 20010909(Sun)T015248
P1 ----------------
P2 ----------------
P3 ----------------
P4 ----------------



Current frame: 18569 of 18640

	0-1		1-1
	FR rhigh	rlow	BYsS↑↓←→AXLR0123
18568	F 0	0	B s X
18569	F 0	0	BYsS↑↓← AXLR012
18570	FR 0	0	BYsS↑↓←AXLR012
18571	FR 0	0	BYsS↑↓←AXLR012
18572	FR 0	0	BYsS↑↓←AXLR012
18573	FR 0	0	BYsS↑↓←AXLR012
18574	FR 0	0	BYsS↑↓←AXLR012
18575	FR 0	0	BYsS↑↓←AXLR012
18576	FR 0	0	BYsS↑↓←AXLR012
18577	FR 0	0	BYsS↑↓←AXLR012
18578	FR 0	0	BYsS↑↓←AXLR012
18579	FR 0	0	BYsS↑↓←AXLR012
18580	FR 0	0	BYsS↑↓←AXLR012
18581	FR 0	0	BYsS↑↓←AXLR012
18582	FR 0	0	BYsS↑↓←AXLR012
18583	FR 0	0	BYsS↑↓←AXLR012
18584	FR 0	0	BYsS↑↓←AXLR012
18585	FR 0	0	BYsS↑↓←AXLR012
18586	FR 0	0	BYsS↑↓←AXLR012
18587	FR 0	0	BYsS↑↓←AXLR012
18588	FR 0	0	BYsS↑↓←AXLR012
18589	FR 0	0	BYsS↑↓←AXLR012
18590	FR 0	0	BYsS↑↓←AXLR012
18591	FR 0	0	BYsS↑↓←AXLR012
18592	FR 0	0	BYsS↑↓←AXLR012
18593	FR 0	0	BYsS↑↓←AXLR012
18594	FR 0	0	BYsS↑↓←AXLR012
18595	FR 0	0	BYsS↑↓←AXLR012
18596	FR 0	0	BYsS↑↓←AXLR012
18597	FR 0	0	BYsS↑↓←AXLR012
18598	FR 0	0	BYsS↑↓←AXLR012
18599	FR 0	0	BYsS↑↓←AXLR012
18600	FR 0	0	BYsS↑↓←AXLR012
18601	FR 0	0	BYsS↑↓←AXLR012
18602	FR 0	0	BYsS↑↓←AXLR012
18603	FR 0	0	BYsS↑↓←AXLR012
18604	FR 0	0	BYsS↑↓←AXLR012
18605	FR 0	0	BYsS↑↓←AXLR012
18606	FR 0	0	BYsS↑↓←AXLR012
18607	FR 0	0	BYsS↑↓←AXLR012
18608	FR 0	0	BYsS↑↓←AXLR012
18609	FR 0	0	BYsS↑↓←AXLR012

Memory watches:

0000	9DD3
0002	0000
0012	89B0
Controller input	0000
Game mode	12
Movement handler	A337
RNG	8151
Samus animation frame	0007
Samus animation timer	1
Samus iframes	10
Samus knockback	0
Samus movement handler	A337
Samus pose	00F2
Samus top-half sprite	01D6
Shot cooldown	0
Sprite stack	000
X (px)	772
X (sub)	26112
X momentum (px)	0
X momentum (sub)	0
X speed (px)	0
X speed (sub)	0
Y (px)	448
Y (sub)	65535
Y speed (px)	0
Y speed (sub)	0
beetom #1 flags	28
beetom #1 state	B9C2
beetom #2 flags	28
beetom #2 state	B9C2
beetom #3 flags	20
beetom #3 state	BDA9
beetom #4 flags	28
beetom #4 state	B9C2

```
F4 45 89     PEA $8945
08           PHP
5C D3 84 82  JML $8284D3
```

P1 BYsS↑↓←-AXLR012-
P2 BYsS↑↓←-AXLR012-
P3 BYsS↑↓←-AXLR012-
P4 BYsS↑↓←-AXLR012-


ENERGY 99