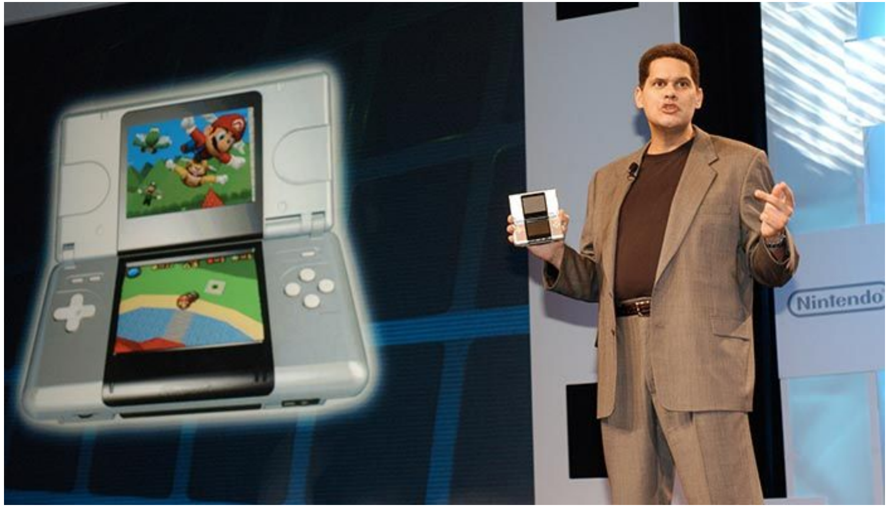
NitrousDS - The Journey

By Sarantis K

What is the Nintendo DS?

- Nintendo’s new portable game console in 2004
- Started development in 2002
- Announced in 2003
- Released in 2004



Reggie Fils-Aimé announcing a prototype of the Nintendo DS at E3 2003

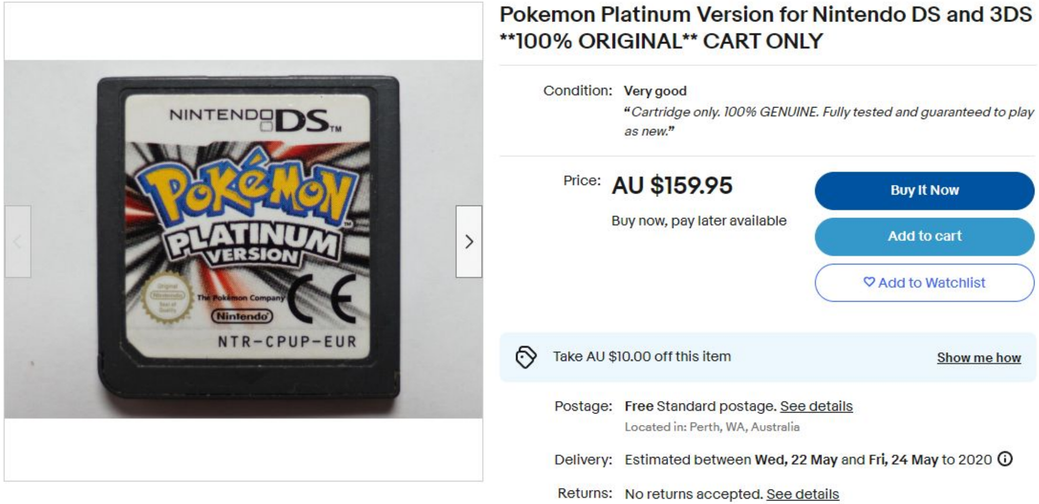
The new (and strange) features of the DS

- Two screens!?
- A touchscreen (before the iPhone)
- 3D graphics (on a handheld)
- Local wireless multiplayer (and Wi-Fi too)
- Backwards-compatibility with the Game Boy Advance
- A questionable design

The terrifying design of the original model

The Nintendo DS today

- Discontinued in 2014 (games too)
- Backwards-compatible successor Nintendo 3DS discontinued in 2020
- No longer possible to acquire new hardware or games



Pokemon Platinum Version for Nintendo DS and 3DS **100% ORIGINAL CART ONLY**

Condition: **Very good**
"Cartridge only. 100% GENUINE. Fully tested and guaranteed to play as new."

Price: **AU $159.95**
Buy now, pay later available

Take AU $10.00 off this item — Show me how

Postage: Free Standard postage. See details
Located in: Perth, WA, Australia

Delivery: Estimated between **Wed, 22 May** and **Fri, 24 May** to 2020 ⓘ

Returns: No returns accepted. See details

A second-hand DS game cartridge. Might even be fake. Who knows?

A DS with a broken hinge

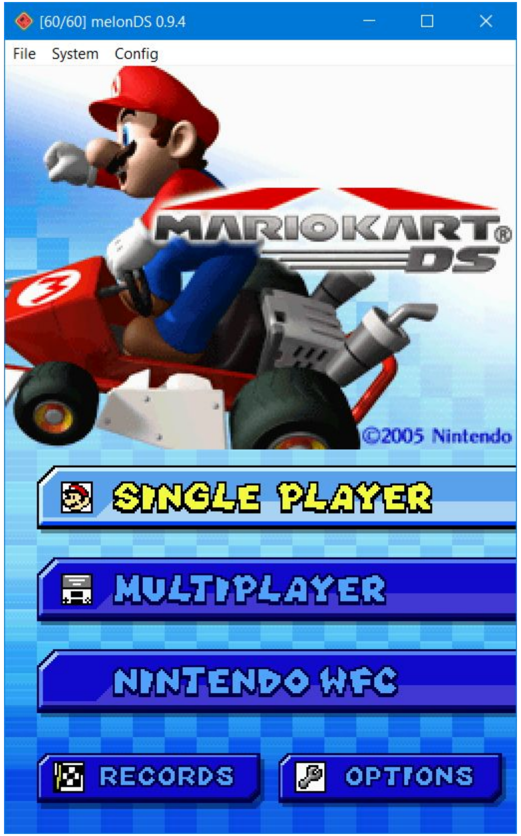
Playing Nintendo DS games on a computer

- DS games come on proprietary cartridges
- They don’t fit into your PC
- You *can* extract the contents to a file
- I can’t say how, though
- The files cannot be opened a computer
- You need special software

A Nintendo DS cartridge of Mario Kart DS next to a can of Coca-Cola for scale

Introducing: Emulators

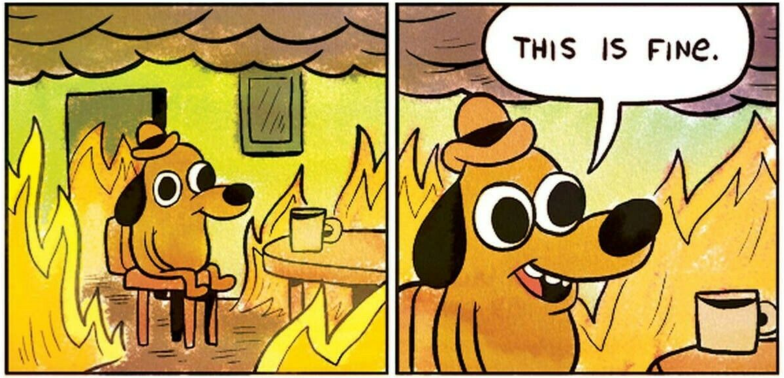
- They perform “magic” and “miracles” to simulate real hardware
- The special software needed to play Nintendo DS games on your computer!
- They are able to read NDS files, and mimic a real Nintendo DS in order to run the software
- They convert the code and contents into a format that your PC can understand and “emulate” a Nintendo DS.



Mario Kart DS running on my computer using an emulator called “melonDS”, wow!

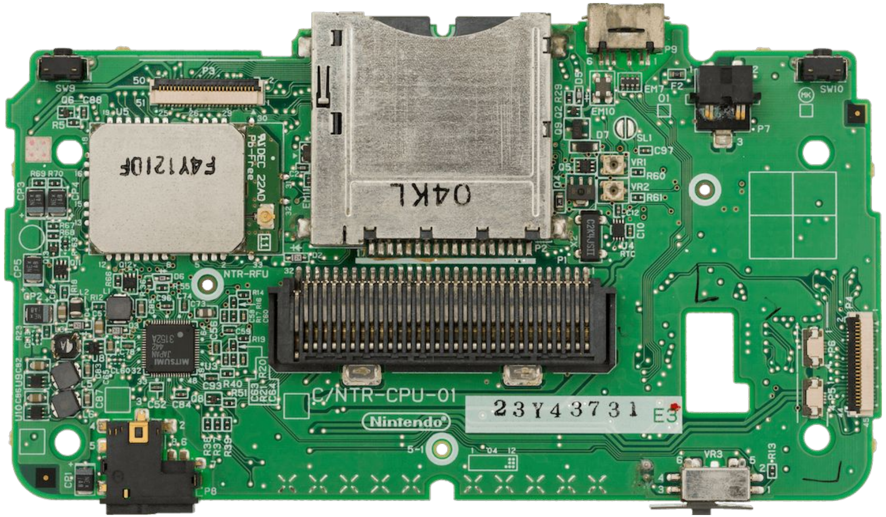
My Project: NitrousDS

- Doing my Honours at The University of Technology Sydney
- I thought it would be interesting to make a Nintendo DS Emulator
- My Honours lasts for about 1 year
- I had no idea where to start making an Emulator, most guides are very basic and aren't for the Nintendo DS
- I don't have time to practice or learn with something simple



Learning about the hardware

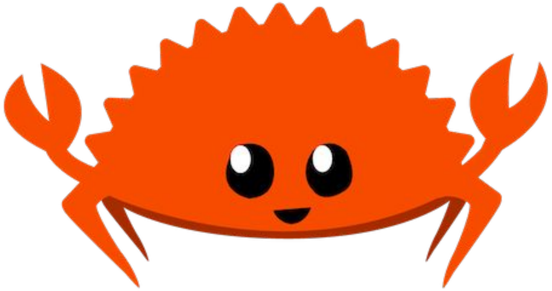
- ARM9 CPU clocked at 66mhz
- ARM7 CPU clocked at 33mhz
- Essentially two computers hacked together
- 4mb of RAM
- Nintendo’s Pixel Processing Unit (2D Graphics)
- 3D (also proprietary)
- A lot of other stuff



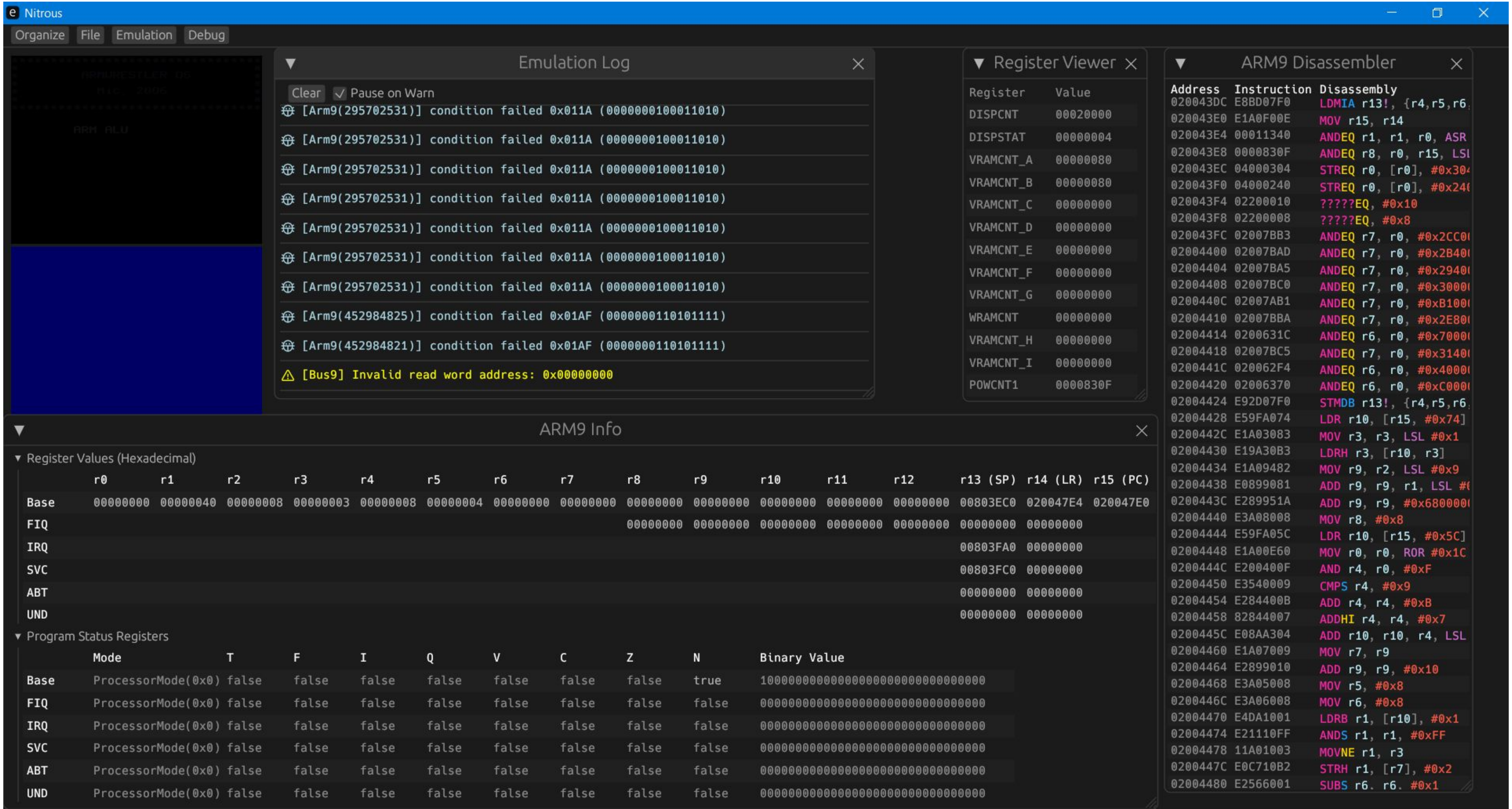
The main Nintendo DS motherboard (Rodrigo Copetti: copetti.org)

Finally started writing the Emulator

- Most emulators are written in C++ because it’s fast
- Rust is arguably as fast as C++ (but I like Rust a lot more)
- I don’t need to be as conscious about performance when writing clean code
- A lot easier to refactor code, which will happen a lot when making an Emulator
- Using egui has helped substantially for making dev tools
- Took 1 hour to support WASM (runs in your web browser)



April 4th, 2024 - It's alive!



```
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(452984825)] condition failed 0x01AF (0000000110101111)
[Arm9(452984821)] condition failed 0x01AF (0000000110101111)
⚠ [Bus9] Invalid read word address: 0x00000000
```

Register	Value
DISPCNT	00020000
DISPSTAT	00000004
VRAMCNT_A	00000080
VRAMCNT_B	00000080
VRAMCNT_C	00000000
VRAMCNT_D	00000000
VRAMCNT_E	00000000
VRAMCNT_F	00000000
VRAMCNT_G	00000000
WRAMCNT	00000000
VRAMCNT_H	00000000
VRAMCNT_I	00000000
POWCNT1	0000830F

```
Address  Instruction Disassembly
020043DC E8BD07F0    LDMIA r13!, {r4,r5,r6,
020043E0 E1A0F00E    MOV r15, r14
020043E4 00011340    ANDEQ r1, r1, r0, ASR
020043E8 0000830F    ANDEQ r8, r0, r15, LSL
020043EC 04000304    STREQ r0, [r0], #0x304
020043F0 04000240    STREQ r0, [r0], #0x240
020043F4 02200010    ?????EQ, #0x10
020043F8 02200008    ?????EQ, #0x8
020043FC 02007BB3    ANDEQ r7, r0, #0x2CC0
02004400 02007BAD    ANDEQ r7, r0, #0x2B40
02004404 02007BA5    ANDEQ r7, r0, #0x2940
02004408 02007BC0    ANDEQ r7, r0, #0x3000
0200440C 02007AB1    ANDEQ r7, r0, #0xB100
02004410 02007BBA    ANDEQ r7, r0, #0x2E80
02004414 0200631C    ANDEQ r6, r0, #0x7000
02004418 02007BC5    ANDEQ r7, r0, #0x3140
0200441C 020062F4    ANDEQ r6, r0, #0x4000
02004420 02006370    ANDEQ r6, r0, #0xC000
02004424 E92D07F0    STMDB r13!, {r4,r5,r6,
02004428 E59FA074    LDR r10, [r15, #0x74]
0200442C E1A03083    MOV r3, r3, LSL #0x1
02004430 E19A30B3    LDRH r3, [r10, r3]
02004434 E1A09482    MOV r9, r2, LSL #0x9
02004438 E0899081    ADD r9, r9, r1, LSL #
0200443C E289951A    ADD r9, r9, #0x680000
02004440 E3A08008    MOV r8, #0x8
02004444 E59FA05C    LDR r10, [r15, #0x5C]
02004448 E1A00E60    MOV r0, r0, ROR #0x1C
0200444C E200400F    AND r4, r0, #0xF
02004450 E3540009    CMPS r4, #0x9
02004454 E284400B    ADD r4, r4, #0xB
02004458 82844007    ADDHI r4, r4, #0x7
0200445C E08AA304    ADD r10, r10, r4, LSL
02004460 E1A07009    MOV r7, r9
02004464 E2899010    ADD r9, r9, #0x10
02004468 E3A05008    MOV r5, #0x8
0200446C E3A06008    MOV r6, #0x8
02004470 E4DA1001    LDRB r1, [r10], #0x1
02004474 E21110FF    ANDS r1, r1, #0xFF
02004478 11A01003    MOVNE r1, r3
0200447C E0C710B2    STRH r1, [r7], #0x2
02004480 E2566001    SUBS r6. r6. #0x1
```

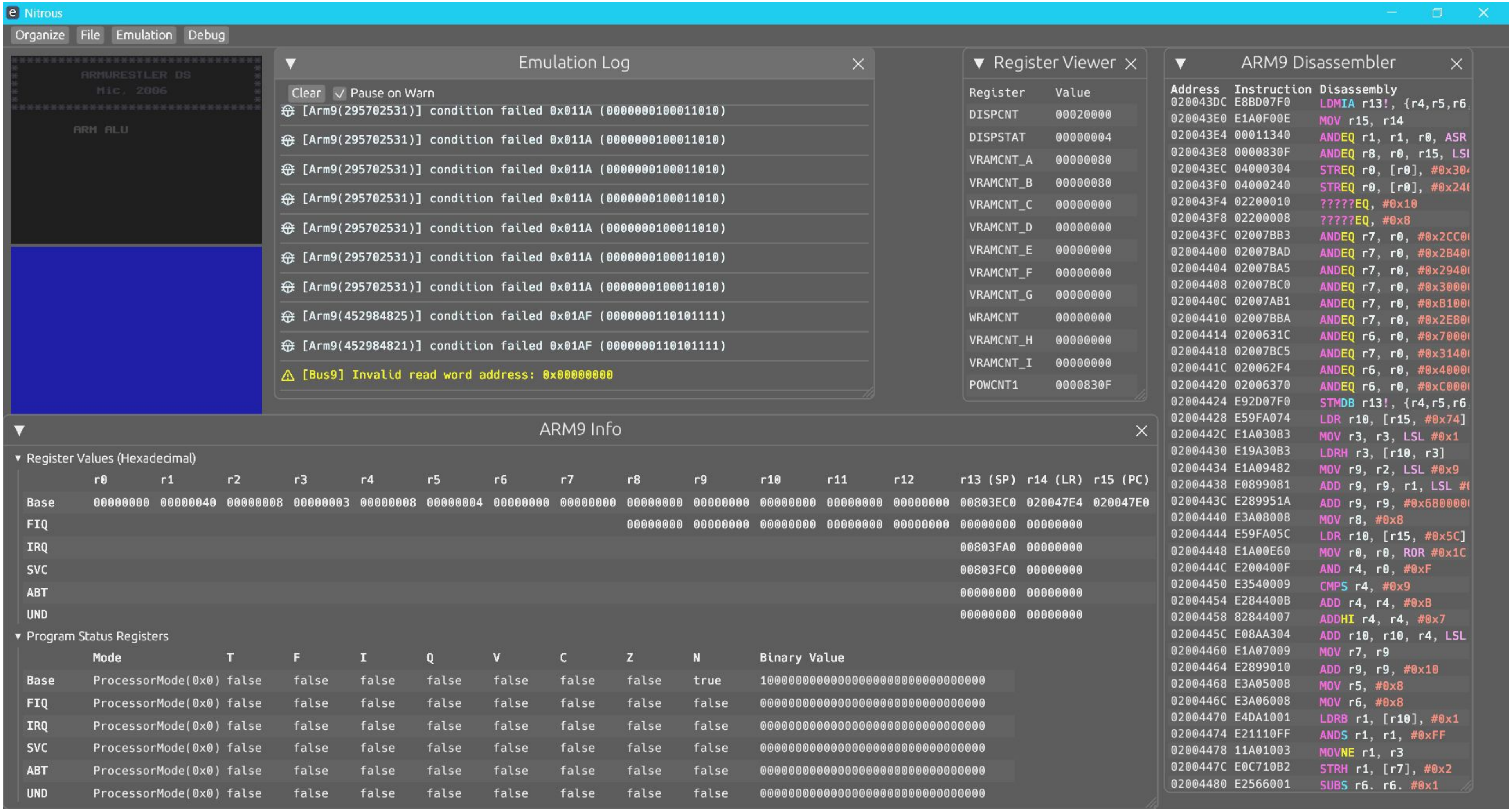
▾ Register Values (Hexadecimal)

	r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13 (SP)	r14 (LR)	r15 (PC)
Base	00000000	00000040	00000008	00000003	00000008	00000004	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00803EC0	020047E4	020047E0
FIQ									00000000	00000000	00000000	00000000	00000000	00000000	00000000	
IRQ														00803FA0	00000000	
SVC														00803FC0	00000000	
ABT														00000000	00000000	
UND														00000000	00000000	

▾ Program Status Registers

	Mode	T	F	I	Q	V	C	Z	N	Binary Value
Base	ProcessorMode(0x0)	false	false	false	false	false	false	false	true	10000000000000000000000000000000
FIQ	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
IRQ	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
SVC	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
ABT	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
UND	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000

April 4th, 2024 - It’s alive! (brightness turned up to see)



```
ARMWRESTLER DS
Mic, 2006

ARM ALU
```

```
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(295702531)] condition failed 0x011A (0000000100011010)
[Arm9(452984825)] condition failed 0x01AF (0000000110101111)
[Arm9(452984821)] condition failed 0x01AF (0000000110101111)
⚠ [Bus9] Invalid read word address: 0x00000000
```

Register	Value
DISPCNT	00020000
DISPSTAT	00000004
VRAMCNT_A	00000080
VRAMCNT_B	00000080
VRAMCNT_C	00000000
VRAMCNT_D	00000000
VRAMCNT_E	00000000
VRAMCNT_F	00000000
VRAMCNT_G	00000000
WRAMCNT	00000000
VRAMCNT_H	00000000
VRAMCNT_I	00000000
POWCNT1	0000830F

```
Address  Instruction Disassembly
020043DC E8BD07F0    LDMIA r13!, {r4,r5,r6,
020043E0 E1A0F00E    MOV r15, r14
020043E4 00011340    ANDEQ r1, r1, r0, ASR
020043E8 0000830F    ANDEQ r8, r0, r15, LSL
020043EC 04000304    STREQ r0, [r0], #0x304
020043F0 04000240    STREQ r0, [r0], #0x24
020043F4 02200010    ?????EQ, #0x10
020043F8 02200008    ?????EQ, #0x8
020043FC 02007BB3    ANDEQ r7, r0, #0x2CC0
02004400 02007BAD    ANDEQ r7, r0, #0x2B40
02004404 02007BA5    ANDEQ r7, r0, #0x2940
02004408 02007BC0    ANDEQ r7, r0, #0x3000
0200440C 02007AB1    ANDEQ r7, r0, #0xB100
02004410 02007BBA    ANDEQ r7, r0, #0x2E80
02004414 0200631C    ANDEQ r6, r0, #0x7000
02004418 02007BC5    ANDEQ r7, r0, #0x3140
0200441C 020062F4    ANDEQ r6, r0, #0x4000
02004420 02006370    ANDEQ r6, r0, #0xC000
02004424 E92D07F0    STMDB r13!, {r4,r5,r6,
02004428 E59FA074    LDR r10, [r15, #0x74]
0200442C E1A03083    MOV r3, r3, LSL #0x1
02004430 E19A30B3    LDRH r3, [r10, r3]
02004434 E1A09482    MOV r9, r2, LSL #0x9
02004438 E0899081    ADD r9, r9, r1, LSL #
0200443C E289951A    ADD r9, r9, #0x680000
02004440 E3A08008    MOV r8, #0x8
02004444 E59FA05C    LDR r10, [r15, #0x5C]
02004448 E1A00E60    MOV r0, r0, ROR #0x1C
0200444C E200400F    AND r4, r0, #0xF
02004450 E3540009    CMPS r4, #0x9
02004454 E284400B    ADD r4, r4, #0xB
02004458 82844007    ADDHI r4, r4, #0x7
0200445C E08AA304    ADD r10, r10, r4, LSL
02004460 E1A07009    MOV r7, r9
02004464 E2899010    ADD r9, r9, #0x10
02004468 E3A05008    MOV r5, #0x8
0200446C E3A06008    MOV r6, #0x8
02004470 E4DA1001    LDRB r1, [r10], #0x1
02004474 E21110FF    ANDS r1, r1, #0xFF
02004478 11A01003    MOVNE r1, r3
0200447C E0C710B2    STRH r1, [r7], #0x2
02004480 E2566001    SUBS r6. r6. #0x1
```

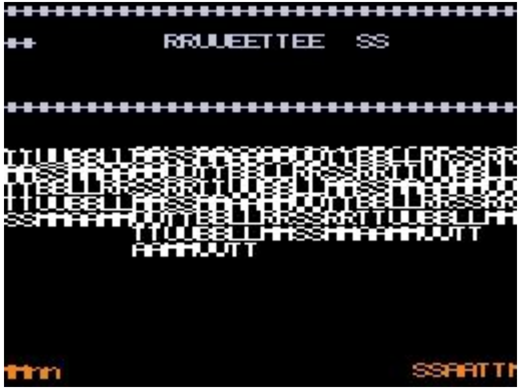
▾ Register Values (Hexadecimal)

	r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13 (SP)	r14 (LR)	r15 (PC)
Base	00000000	00000040	00000008	00000003	00000008	00000004	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00803EC0	020047E4	020047E0
FIQ									00000000	00000000	00000000	00000000	00000000	00000000	00000000	
IRQ														00803FA0	00000000	
SVC														00803FC0	00000000	
ABT														00000000	00000000	
UND														00000000	00000000	

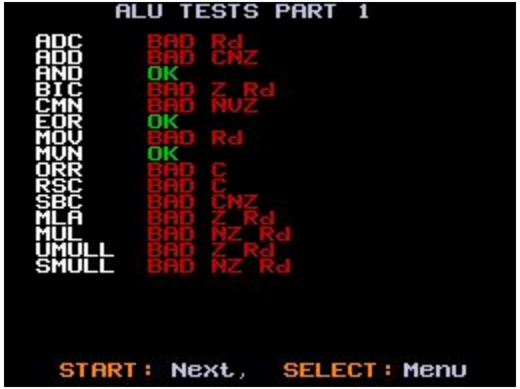
▾ Program Status Registers

	Mode	T	F	I	Q	V	C	Z	N	Binary Value
Base	ProcessorMode(0x0)	false	false	false	false	false	false	false	true	10000000000000000000000000000000
FIQ	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
IRQ	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
SVC	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
ABT	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000
UND	ProcessorMode(0x0)	false	false	false	false	false	false	false	false	00000000000000000000000000000000

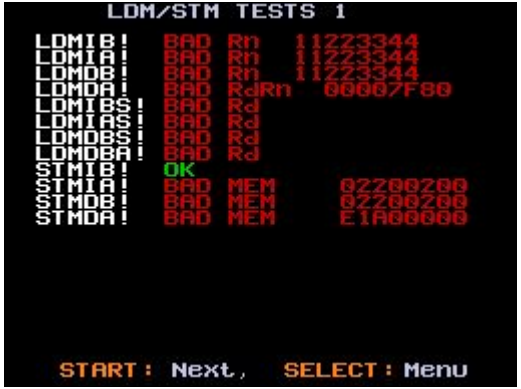
Fixing/improving Emulation



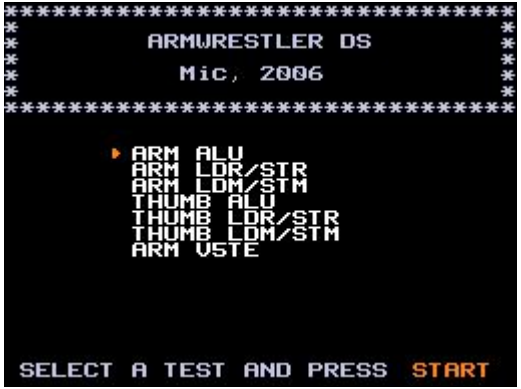
```
            RRUUEETTEE  SS

       TTUUEELLAASSAAAAAUUTT
       AAAAUUTT

                        SSAATTM
```



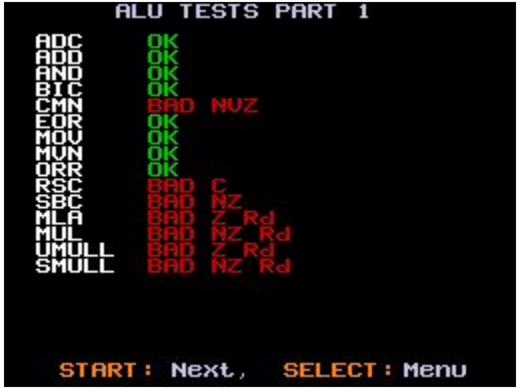
```
       ALU TESTS PART 1

ADC      BAD Rd
ADD      BAD CNZ
AND      OK
BIC      BAD Z Rd
CMN      BAD NVZ
EOR      OK
MOV      BAD Rd
MVN      OK
ORR      BAD C
RSC      BAD C
SBC      BAD CNZ
MLA      BAD Z Rd
MUL      BAD NZ Rd
UMULL    BAD Z Rd
SMULL    BAD NZ Rd

   START: Next,   SELECT: Menu
```



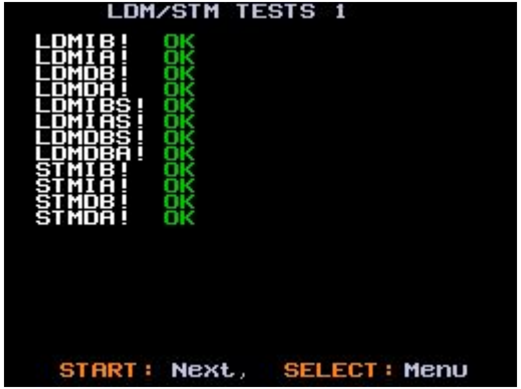
```
       LDM/STM TESTS 1

LDMIB!   BAD Rn    11223344
LDMIA!   BAD Rn    11223344
LDMDB!   BAD Rn    11223344
LDMDA!   BAD RdRn   00007F80
LDMIBS!  BAD Rd
LDMIAS!  BAD Rd
LDMDBS!  BAD Rd
LDMDBA!  BAD Rd
STMIB!   OK
STMIA!   BAD MEM     02200200
STMDB!   BAD MEM     02200200
STMDA!   BAD MEM     E1A00000

   START: Next,   SELECT: Menu
```



```
********************************
*                              *
*        ARMWRESTLER DS        *
*                              *
*          Mic, 2006           *
*                              *
********************************

          ▸ ARM ALU
            ARM LDR/STR
            ARM LDM/STM
            THUMB ALU
            THUMB LDR/STR
            THUMB LDM/STM
            ARM V5TE

 SELECT A TEST AND PRESS START
```



```
       ALU TESTS PART 1

ADC      OK
ADD      OK
AND      OK
BIC      OK
CMN      BAD NVZ
EOR      OK
MOV      OK
MVN      OK
ORR      OK
RSC      BAD C
SBC      BAD NZ
MLA      BAD Z Rd
MUL      BAD NZ Rd
UMULL    BAD Z Rd
SMULL    BAD NZ Rd

   START: Next,   SELECT: Menu
```



```
       LDM/STM TESTS 1

LDMIB!   OK
LDMIA!   OK
LDMDB!   OK
LDMDA!   OK
LDMIBS!  OK
LDMIAS!  OK
LDMDBS!  OK
LDMDBA!  OK
STMIB!   OK
STMIA!   OK
STMDB!   OK
STMDA!   OK

   START: Next,   SELECT: Menu
```

What’s next?

- The basic core parts are nearly done
- The ARM CPUs have seen a lot of progress
- Progress should be quicker from here on out
- Running more test software
- Eventually loading games

Thank you for listening

GitHub Repository

https://github.com/Spantle/nitrous

Discord

https://discord.gg/zw3GrRzxww