

ちょっとイレギュラーな小技ばっかを書く予定。
どう利用するかは各人でガンガレ。

現在のネタは以下の4つ。

- ・ タッチした座標の検出
- ・ 垂直ブランク割り込みの無効化（高速化コードの基礎の基礎）
- ・ ボタン連打コードのサーチ
- ・ スローモーションコードの作成方法

タッチした座標の検出

本来はARM7でXKEYSレジスタ(アドレス04000136)にアクセスしないとダメだが、そこに格納されてる値から必要なデータを取り出すのは超マンドクセ。
素直に「絶対座標」がほしいならno\$gbaで**画面をタッチしたまま F1 を押す**といい。
これでヘルプ表示でエミュレーションが停止するので、タッチ情報などもそのまま止まるんだぜ。

大抵、ゲーム中では「どこかでタッチ情報を処理」しているので、それを一時的に保持しているアドレスがヒットするはず。
ちなみにDSのスクリーン解像度は 256 × 192 なのでこれに近い値のものを探すべし。

< タッチ！カービィでそれを検証 >

下で敵にボコボコにされてるカービィは無視して、適当な場所をクリックしながらF1。
エミュレーションが止まるので、emuhasteからクリック場所を基準に増加 or 減少で絞り込み。
俺の環境だとX座標がアドレス 020D9576、Y座標が020D9578 にそれっぽい値が入った。
(他にも色々いっぱいヒットしてるけど)。

変動アドレスかもしれんけど、そのときはそのときで考える！！
まあ、これらの値は2バイト(WORD)サイズだった。

垂直ブランク割り込みの無効化（高速化コードの基礎の基礎）

プログラム内に存在するthumbコードから以下の箇所を探す

```
2200 mov r2,0x00
DF05 swi #0x05
( ndsdis2ではthumbコードを吐けないので注意 )
```

最も簡単な方法としては、DSエミュレータで起動後にhasteDSもしくはemuhasteで16進数 "002205DF" をパラメータサーチ。
2箇所以上該当するので、そのアドレスに対して 00002200 を書き込むコードを実行する。

(例) 直感ヒトフデの場合 "002205DF"がアドレス 02000350 と 0238BD44 に合計2個ヒット。
改造コードはこうなる。
02000350 00002200
0238BD44 00002200

なお、ゲームによってはこれで高速化することもあるが、何も起こらない場合もある。

ボタン連打コードのサーチ

DSのボタン入力情報はアドレス04000130と04000136に格納されている。
んで、連打というのは「ボタン押して離してまた押して...」の繰り返しであり、
必ずプログラム側が「離して」という部分を判断している(はず)。
この「離して」の判断を押しっぱなしでも有効にするのが「ボタン連打コードのサーチ」である。

当然ながらゲームごとにボタン処理のルーチンが異なるため、ここで紹介している方法では連打できない場合もある。

とりあえず「ぶよぶよフィーバー」で実験。

対戦が始まったら...

Aボタン(に割り当てたキー)を押しながらF1キーを押してヘルプ表示(停止) 値 "01" を検索。
Bボタン(に割り当てたキー)を押しながらF1キーを押してヘルプ表示(停止) 値 "02" を検索。
何も押さずに値 "00" を検索。

以下の4つのアドレスに該当

0212C848
0212C84A
021454D8
021454DA

これらのアドレスを判定しているルーチンをいじれば連打状態にできると思うが、
ぶよフィだと

2212C848 00000000
2212C84A 00000000
221454D8 00000000
221454DA 00000000

こんな感じで00(何も押さない)を書き込むことで、

- 1.ユーザがボタンを押す
- 2.ゲーム中の"ボタンを押した際の処理"を実行
- 3.改造ツール側が"00"を書き込む
- 4.ボタンを放したと判断される

という流れが発生し、ボタンを押しっぱなしでもぶよがぐるぐる回るようになる。
ゲームによっては00,01,02ではなくビット反転した値(FF,FE,FD)で処理している可能性もあるので注意。

スローモーションコードの作成方法

過去ログ7から引用

663 : 前スレ346 : 2007/11/15(木) 01:14:41 ID:Te+tUkrg

とりあえず書いたので投稿します。

高速化とは対照的な「自作ルーチン埋込」の技術を応用したスローモーションコードの作成法です。
スローモーション機能付きのマジンコンを持っていない、または持っているけどスローモーションの
ON/OFFをもっと手軽に切り替えたいという場合に有効。

スローモーションは時間稼ぎをするプログラムを挿入し、故意に処理落ちを発生させることで実現できます。
では実際に「超執刀 カドゥケウス (アトラス ベストコレクション)」でスローモーションコードを作ってみます。
とりあえず逆アセンブルして04000130を検索すると以下の部分でヒット。

```
:02000F34 E59F10D8 ldr r1,[r15, #+0xd8] ;r15+0xd8=(02001014)=#671109168(0x04000130)  
:02000F38 E59F00D8 ldr r0,[r15, #+0xd8] ;r15+0xd8=(02001018)=#41942952(0x027ffa8)
```

```

:02000F3C E1D130B0 ldrh r3,[r1, #+0x0] ;r1+0x0=(04000130)=#0(0x00000000)
:02000F40 E1D010B0 ldrh r1,[r0, #+0x0] ;r0+0x0=(027ffa8)=#1916546356(0x723c2d34)
:02000F44 E59F20D0 ldr r2,[r15, #+0xd0] ;r15+0xd0=(0200101c)=#35515132(0x021deafc)
:02000F48 E59F00D0 ldr r0,[r15, #+0xd0] ;r15+0xd0=(02001020)=#12287(0x00002fff)
:02000F4C E1831001 orr r1,r3,r1
:02000F50 E0211000 eor r1,r1,r0
:02000F54 E0010000 and r0,r1,r0
:02000F58 E1A01800 mov r1,r0,lsl #0x10 ;r1=805240832(0x2fff0000)

```

上記の部分から以下のことが読み取れる。

1. この部分は通常なら毎秒60回実行される。
2. プログラムカウンタが\$02000F58に来ている時点ではr3レジスタにキーデータが格納されている。
3. アドレス\$02000F58から追加ルーチンへ飛ばしても追加ルーチンの最後で mov r1,r0,lsl #0x10 を実行すれば追加ルーチンではr1レジスタを自由に使用できる。

この3つの条件を頭に入れながら\$02000F58の部分に時間稼ぎをするプログラムを挿入する。
\$021BD9E0 - \$021BD9FCの部分が空き領域として使えるので追加ルーチンはこの部分に埋め込みます。

```

02000F58 EA06F2A0 b $021BD9E0 ;追加ルーチンへジャンプ

021BD9E0 E3130C01 tst r3, #0 ;Rボタンをチェックして
021BD9E4 0A000003 beq $021BD9F8 ;押されていたら潰した命令を実行して復帰

```

```

Rボタンが押されていなければ時間稼ぎをする
021BD9E8 E3A01A40 mov r1, #0 ;r1 = #0
021BD9EC E2411001 sub r1, r1, #0 ;r1 = r1 - #0
021BD9F0 E3510000 cmps r1, #0 ;r1 = #0 かチェック
021BD9F4 1AFFFFFC bne $021BD9EC ;r1 = #0 ならループ脱出

```

```

021BD9F8 E1A01800 mov r1, r0, lsl #10 ;潰した命令を実行
021BD9FC EAF90D56 b $02000F5C ;追加ルーチンから復帰

```

このままでも動くけどARDSの機能を利用して無駄を省いてコード完成。
超執刀 カドゥケウス (アトラス ベストコレクション) AKDJ BA5320CB

```

スローモーション1/2
52000F58 E1A01800
E21BD9E0 00000020
E3130C01 0A000003
E3A01A40 E2411001
E3510000 1AFFFFFC
E1A01800 EAF90D56
02000F58 EA06F2A0
D0000000 00000000

```

ゲームの速度が通常の1/2になります。Rボタンを押している間は通常の速度になります。

過去ログ8から引用

854 : 名無しさん@お腹いっぱい : 2008/01/06(日) 04:34:02 ID:1ltsVXNi

前スレのスローモーションコードの作り方を参考に
眼力トレーニングでスローモーションコード作ろうとしているのですが

```

:0201245C E59F1260 ldr r1,[r15, #+0x260] ;r15+0x260=(020126c4)=#67109168(0x04000130)
:02012460 E59F0250 ldr r0,[r15, #+0x250] ;r15+0x250=(020126b8)=#41942952(0x027ffa8)
:02012464 E1D130B0 ldrh r3,[r1, #+0x0] ;r1+0x0=(04000130)=#0(0x00000000)
:02012468 E1D010B0 ldrh r1,[r0, #+0x0] ;r0+0x0=(027ffa8)=#1347436608(0x50504040)
:0201246C E59F0254 ldr r0,[r15, #+0x254] ;r15+0x254=(020126c8)=#12287(0x00002fff)
:02012470 E59F223C ldr r2,[r15, #+0x23c] ;r15+0x23c=(020126b4)=#34712596(0x0211ac14)
:02012474 E1831001 orr r1,r3,r1
:02012478 E0211000 eor r1,r1,r0
:0201247C E0010000 and r0,r1,r0
:02012480 E1A01800 mov r1,r0,lsl #0x10 ;r1=805240832(0x2fff0000)

```

\$02106090-&\$021060ACの部分が空き領域として使えるので、ここに追加ルーチン埋め込もうとしているのですが

```

02012480 ???????? b $02106090 ;追加ルーチンへジャンプ

02106090 E3130C01 tst r3, #0 ;Rボタンをチェックして
02106094 0A000003 beq $021060A8 ;押されていたら潰した命令を実行して復帰

```

Rボタンが押されていなければ時間稼ぎをする

```

02106098 E3A01A40  mov r1, #$00040000 ;r1 = #$00040000
0210609C E2411001  sub r1, r1, #$01    ;r1 = r1 - #$01
021060A0 E3510000  cmps r1, #$00      ;r1 = #$00 かチェック
021060A4 1AFFFFFC  bne $0210609C     ;r1 = #$00 ならループ脱出

021060A8 E1A01800  mov r1, r0, lsl #$10 ;潰した命令を実行
021060AC ?????????? b $02012484       ;追加ルーチンから復帰

```

と、?????????のこの、追加ルーチンへジャンプと、復帰の値の算出方法がわからなくて詰まっています。どなたかご存知でしたら教えて頂きたいです。

857 : 名無しさん@お腹いっぱい, : 2008/01/06(日) 10:12:02 ID:Q9egwDtY

>>854

b命令のコードはEAxxxxxx,

プラス方向へのジャンプの値は『(飛び先-コード実行位置)/4 - 2』で求められる。

(例):02012480 b \$02106090

EA000000 + (02106090-02012480)/4 - 2

EA000000 + 3CF02

EA03CF02

マイナス方向へのジャンプは『(飛び先アドレス-命令実行アドレス-8)/4』で求められる。

(例):021060AC b \$02012484

EA000000 + (02012484-021060AC-8)/4

EA000000 + 3FFFFFFF30F4

EA000000 + FC30F4 下6桁のみを切り出す

EAF30F4

でも常駐ルーチンはアドレス02000000に配置したほうが良いと思う。