cracker's DS Trainer Making Tutorial

This tutorial will teach you the basics of how to create trainers for DS games. You don't necessarily need to know any ARM assembly language but you will need to learn some basics of programming. If this sounds to hard for you then you are probably in the wrong place. Otherwise proceed and have fun.

The first part of the tutorial is just to list the steps I use. In the second part I will walk you through the actual process.

Tools You Will Need

- Action Replay Code Manager which can be found at a link on this thread
- NDSDIS2 - an NDS disassembler of course!
- disnds.bat - batch file to disassemble
- devkitARM - to assemble your source code (the win32 release of course)
- filecutter - a tool to chop up files, this will be used for when you assemble your source code
- assemble.bat - a batch file to assemble your source code properly
- XVI32 - a good, free hex editor
- ips XP - an ips patching utility to create a patch for your new trainer
- Windows calculator set to Scientific mode, Hex

Setting Up

- Install the Action Replay Code Manager (make sure you install XML or it won't work properly)
- Install ips XP
- Extract XVI32 to wherever you want
- Extract devkitARM to c:\ (or the root of another drive but that will require a small edit of the assemble.bat)
- Place assemble.bat and filecutter.exe into c:\devkitARM\bin
- Create c:\dstrainers and place ndsdis2.exe and disnds.bat into it

Disassembling the Game

Place the .nds into c:\dstrainers. Drag and drop the .nds onto disnds.bat to generate the disassembled source.

Part I - Overview

Finding the Code to Hack

Open the .txt file created from disassembling the game. Create a new text file to use for hacking - name it something like <game name> hacking.txt. Open the Action Replay Code Manager (will refer to this as

ARCM from now on). Find the code you want to use - it must start with "02", "12", or "22". Change the first two digits to "02" if they aren't already. Search the disassembly for them. If it is found in the disassembled source copy the lines of source a few lines above it to a few lines below it and paste into your hacking text file. Then repeat the search until no more instances are found.

Finding the Code to Hack - Harder Method

If the address isn't found in the disassembled source then the other option is to create a small assembly routine that loads the address manually and stores the maximum value into it. In order to do this you need to find the area of the game which gets called repeatedly. The key handling function is the best place for an obvious reason. For many games the work has already been done for you. DipStar users have found many Mastercodes for games that will give you the general area to place your routine. You can find a list of them here. If the game you want to train isn't on the list then you can manually search for the area of the game that handles key input by searching for 0x04000130 in the disassembled source. If it is found then copy the surrounding code and paste it into your hacking text file and repeat the process until you can't find any other instances.

Part II - Hands on

Finding the Code to Hack

Disassemble your legally owned backup copy of Super Monkey Ball - Touch & Roll. Open the disassembled source file. Create a new text file named smb hacking.txt. Open ARCM. What trainer would be complete without Infinite Lives? The AR code is for Infinite Lives is 021fbf34 00000063. Since it already starts with "02" we can leave it as it is. Search for 021fbf34 in the disassembly and the first instance of it will be:

```
:020AAE0C 159F05EC ldrne r0,[r15, #+0x5ec]
;r15+0x5ec=*(020ab400)=#35634996(0x021fbf34)
```

So we will copy the code around it:

```
:020AADF8 E59F05FC ldr r0,[r15, #+0x5fc]      ;r15+0x5fc=*(020ab3fc)=#35635112(0x021fbfa8)
:020AADFC E59F25F4 ldr r2,[r15, #+0x5f4]      ;r15+0x5f4=*(020ab3f8)=#35635016(0x021fbf48)
:020AAE00 E5900000 ldr r0,[r0, #+0x0]       ;r0+0x0=*(021fbfa8)=#121667392(0x07407f40)
:020AAE04 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf48)=#1074217008(0x40074030)
:020AAE08 E2100001 ands r0,r0,#0x1
:020AAE0C 159F05EC ldrne r0,[r15, #+0x5ec]
 ;r15+0x5ec=*(020ab400)=#35634996(0x021fbf34)
:020AAE10 E243300A sub  r3,r3,#0xA
:020AAE14 15D01000 ldrneb r1,[r0, #+0x0]      ;r0+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AAE18 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf48)=#1074217008(0x40074030)
:020AAE1C 12811001 addne  r1,r1,#0x1
:020AAE20 15C01000 strneb r1,[r0, #+0x0]      ;r0+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AAE24 E59F05D8 ldr r0,[r15, #+0x5d8]      ;r15+0x5d8=*(020ab404)=#35653456(0x02200750)
:020AAE28 E59F15D8 ldr r1,[r15, #+0x5d8]      ;r15+0x5d8=*(020ab408)=#35653388(0x0220070c)
```

We move on and find the rest of the instances in the code:

```
:020AB618 E5D2C000 ldrb r12,[r2, #+0x0]       ;r2+0x0=*(021fbf64)=#121683975(0x0740c007)
:020AB61C E59F20CC ldr r2,[r15, #+0xcc]       ;r15+0xcc=*(020ab6f0)=#35634996(0x021fbf34)
:020AB620 E5D14000 ldrb r4,[r1, #+0x0]      ;r1+0x0=*(021fbf44)=#121678368(0x0740aa20)
:020AB624 E5903000 ldr r3,[r0, #+0x0]      ;r0+0x0=*(021fbf8c)=#-1878572928(0x90074080)
:020AB628 E59F00C4 ldr r0,[r15, #+0xc4]      ;r15+0xc4=*(020ab6f4)=#35635080(0x021fbf88)
:020AB62C E59F10C4 ldr r1,[r15, #+0xc4]      ;r15+0xc4=*(020ab6f8)=#35635016(0x021fbf48)
:020AB630 E5C2C000 strb r12,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AB634 E5803000 str r3,[r0, #+0x0]       ;r0+0x0=*(021fbf88)=#121663495(0x07407007)

:020AB648 E59F20A0 ldr r2,[r15, #+0xa0]       ;r15+0xa0=*(020ab6f0)=#35634996(0x021fbf34)
:020AB64C E59F10A4 ldr r1,[r15, #+0xa4]       ;r15+0xa4=*(020ab6f8)=#35635016(0x021fbf48)
:020AB650 E59F009C ldr r0,[r15, #+0x9c]       ;r15+0x9c=*(020ab6f4)=#35635080(0x021fbf88)
:020AB654 E5D2C000 ldrb r12,[r2, #+0x0]       ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AB658 E59F2084 ldr r2,[r15, #+0x84]      ;r15+0x84=*(020ab6e4)=#35635044(0x021fbf64)
:020AB65C E5D14000 ldrb r4,[r1, #+0x0]      ;r1+0x0=*(021fbf48)=#1074217008(0x40074030)
:020AB660 E59F1080 ldr r1,[r15, #+0x80]      ;r15+0x80=*(020ab6e8)=#35635012(0x021fbf44)
:020AB664 E5903000 ldr r3,[r0, #+0x0]       ;r0+0x0=*(021fbf88)=#121663495(0x07407007)
:020AB668 E59F007C ldr r0,[r15, #+0x7c]       ;r15+0x7c=*(020ab6ec)=#35635084(0x021fbf8c)
:020AB66C E5C2C000 strb r12,[r2, #+0x0]       ;r2+0x0=*(021fbf64)=#121683975(0x0740c007)
:020AB670 E5C14000 strb r4,[r1, #+0x0]      ;r1+0x0=*(021fbf44)=#121678368(0x0740aa20)

:020AD1D8 E59F20E4 ldr r2,[r15, #+0xe4]       ;r15+0xe4=*(020ad2c4)=#35634996(0x021fbf34)
:020AD1DC E59F00D8 ldr r0,[r15, #+0xd8]       ;r15+0xd8=*(020ad2bc)=#35635004(0x021fbf3c)
:020AD1E0 E59F10D0 ldr r1,[r15, #+0xd0]       ;r15+0xd0=*(020ad2b8)=#35634992(0x021fbf30)
:020AD1E4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD1E8 E1D000D0 ldrsb r0,[r0, #+0x0]       ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD1EC E1D110D0 ldrsb r1,[r1, #+0x0]       ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD1F0 E2433001 sub  r3,r3,#0x1
:020AD1F4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD1F8 EB054821 bl 021FF284

:020AD494 E59F00EC ldr r0,[r15, #+0xec]       ;r15+0xec=*(020ad588)=#35634996(0x021fbf34)
:020AD498 E5D00000 ldrb r0,[r0, #+0x0]      ;r0+0x0=*(e00740d0)
:020AD49C E3500000 cmp r0,#0x0
:020AD4A0 0A000015 beq 020AD4FC
:020AD4A4 EB0002D4 bl 020ADFFC
:020AD4A8 E59F20D8 ldr r2,[r15, #+0xd8]       ;r15+0xd8=*(020ad588)=#35634996(0x021fbf34)
:020AD4AC E59F00CC ldr r0,[r15, #+0xcc]       ;r15+0xcc=*(020ad580)=#35635004(0x021fbf3c)
:020AD4B0 E59F10C4 ldr r1,[r15, #+0xc4]       ;r15+0xc4=*(020ad57c)=#35634992(0x021fbf30)
:020AD4B4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD4B8 E1D000D0 ldrsb r0,[r0, #+0x0]       ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD4BC E1D110D0 ldrsb r1,[r1, #+0x0]       ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD4C0 E2433001 sub  r3,r3,#0x1
:020AD4C4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD4C8 EB05476D bl 021FF284

:020ADF18 E2110001 ands r0,r1,#0x1
:020ADF1C 159F00BC ldrne r0,[r15, #+0xbc]       ;r15+0xbc=*(020adfe0)=#35634996(0x021fbf34)
:020ADF20 13A02005 movne  r2,#0x5       ;r2=5(0x5)
:020ADF24 15C02000 strneb r2,[r0, #+0x0]       ;r0+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020ADF28 059F00B0 ldreq r0,[r15, #+0xb0]       ;r15+0xb0=*(020adfe0)=#35634996(0x021fbf34)
:020ADF2C 03A02063 moveq  r2,#0x63       ;r2=99(0x63)
:020ADF30 05C02000 streqb r2,[r0, #+0x0]       ;r0+0x0=*(021fbf34)=#-536395568(0xe00740d0)
```

```
:020ADF34 E59F0084 ldr r0,[r15, #+0x84]      ;r15+0x84=*(020adfc0)=#35635112(0x021fbfa8)
:020ADF38 E59C2000 ldr r2,[r12, #+0x0]      ;r12+0x0=*(021fbf84)=#1084907527(0x40aa6007)
:020ADF3C E1D060D4 ldrsb r6,[r0, #+0x4]      ;r0+0xd4=*(021fc07c)=#121678528(0x0740aac0)
:020ADF40 E3C2700C bic  r7,r2,#0xC
```

What we are most worried about are where we see the number of lives being retrieved from the address, subtracted from and then stored back to the address. Looking at the hits we got we can see that this occurs in 2 of the them:

```
:020AD1E4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD1E8 E1D000D0 ldrsb r0,[r0, #+0x0]      ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD1EC E1D110D0 ldrsb r1,[r1, #+0x0]      ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD1F0 E2433001 sub  r3,r3,#0x1
:020AD1F4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)

:020AD4B4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD4B8 E1D000D0 ldrsb r0,[r0, #+0x0]      ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD4BC E1D110D0 ldrsb r1,[r1, #+0x0]      ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD4C0 E2433001 sub  r3,r3,#0x1
:020AD4C4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
```

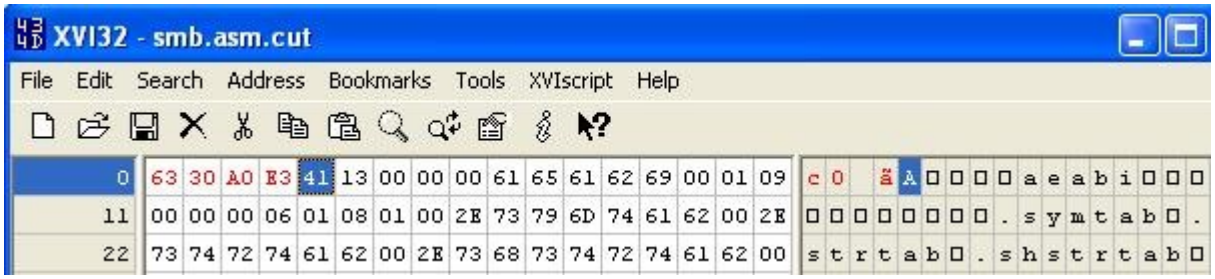
More specifically:

```
:020AD1F0 E2433001 sub  r3,r3,#0x1

:020AD4C0 E2433001 sub  r3,r3,#0x1
```

There are a few things you can do here. You can replace them with dummy commands, change the sub to add or set it to the max number of lives. The safest option is setting the max number of lives just in case another area of the game that wasn't found decreases the number of lives. So now we need to compile a single line of code. Open up c:\devkitARM\bin and create a new text file named smb.asm. Put this assembler command into the file and hit return a couple times:

```
mov r3,#0x63
```

Save the text file and drag and drop it onto assemble.bat. Make sure there were no errors printed in the console window and close then close the window. You will now have a file called smb.asm.cut. This is a raw object file that has the first part stripped off to easily get what we need from it. Open up XVI32 and drop smb.asm.cut onto it.

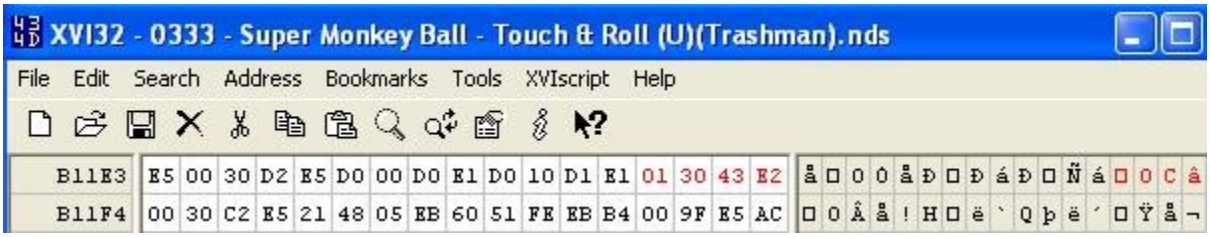


Every line of code is composed of 4 bytes so we only have to worry about the first 4 bytes of the .cut file. An easy way to figure out how much of the file is needed is to look for "aeabi" and then find the "A" and that marks the start of the 'crap data' so the address right before it will be the end of what you need to copy. To

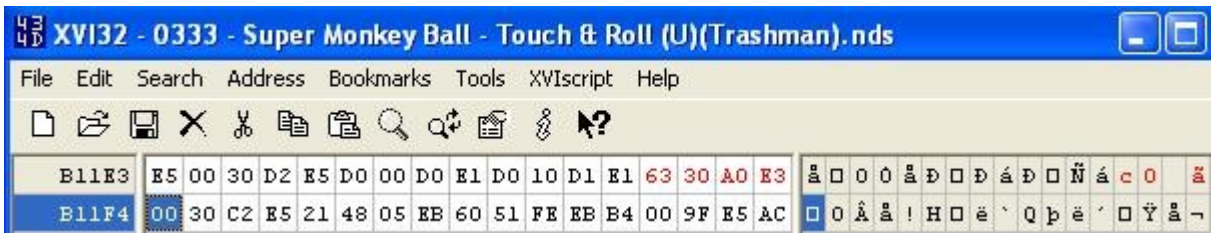
copy the code first go to the start of the file (address 0) then go to Edit -> Block mark. Select the end of the code (address 3 in our case) and go to Edit -> Block mark again. Then go to Edit -> Clipboard -> Copy as hex string. At this point I always put the assembler code I used and the assembled bytecode that we just copied via the hex editor into our hacking text file to make it easy to follow. I also calculate the address where it will placed inside the .nds. This is *usually* the address seen in the disassembled source minus 0x1FFC000. So for the first instance that works out to 0xB11F0 and the second works out to 0xB14C0.

```
:020AD1E4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD1E8 E1D000D0 ldrsb r0,[r0, #+0x0]       ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD1EC E1D110D0 ldrsb r1,[r1, #+0x0]       ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD1F0 E2433001 sub  r3,r3,#0x1
:020AD1F4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)

mov r3,#0x63

63 30 A0 E3 @ B11F0

:020AD4B4 E5D23000 ldrb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)
:020AD4B8 E1D000D0 ldrsb r0,[r0, #+0x0]       ;r0+0xd0=*(40a00810)=#0(0x00000000)
:020AD4BC E1D110D0 ldrsb r1,[r1, #+0x0]       ;r1+0xd0=*(50600810)=#0(0x00000000)
:020AD4C0 E2433001 sub  r3,r3,#0x1
:020AD4C4 E5C23000 strb r3,[r2, #+0x0]      ;r2+0x0=*(021fbf34)=#-536395568(0xe00740d0)

mov r3,#0x63

63 30 A0 E3 @ B14C0
```

Open another instance of XVI32 and load the .nds into it. Go to Address -> Goto. Make sure it is set to hexadecimal and put in the first address (B11F0). Compare the bytes found at the address to the original source code. The bytes are stored backwards in the .nds so E2433001 is 013043E2 -- which is what we see.



Then to apply our patch all we do is go to Edit -> Overwrite string, Select Hex string paste in our code (63 30 A0 E3). Finish by hitting OK. If everything worked right it will look like this:



Then just repeat the same process for the next address to complete your trainer. After you are done click File -> Save As. I suggest you don't save overtop the original (untrained) file since it will mean you have to restore a good copy if further editing is necessary. I usually name them trained - <game name>.nds. Play the

trained game and if everything went well you will have just created your first trainer.

Now to make an ips patch. Open up ips XP. Click Select the source file(s). Change the file type to All files and select the original .nds file. Click Select the target file, All files, and select your trained .nds file. Click Create Ips patch and name it whatever you want.

Finding the Code to Hack - Harder Method

Now that you have an understanding of the easy method we will move on to the harder method. It actually isn't too hard in a lot of cases but it can be frustrating trying to find a good place to put your code without having crashes or other ill effects. For this part you will need your legally owned backup copy of Fullmetal Alchemist - Dual Sympathy. Disassemble the game then open the disassembled source file. The AR codes for Infinite health are:

```
1223bab4 0000270f
1223bab6 0000270f
1223bb84 0000270f
1223bb86 0000270f
```

Note that the value of 0x270F is overkill and setting it to 0xFF works fine as you will see.

We take the codes and change the first two characters to "02" to get our addresses:

```
0223bab4
0223bab6
0223bb84
0223bb86
```

There will actually be some false hits for those addresses but you don't need to worry about searching for them since I already did the dirty work and know that they are false hits. So the next step is to find the area of the game where we can place our function. Luckily some cheaters who use DipStar have done the work for us for this game so that makes it a lot easier on us. Once again the link to the code library is here. The Mastercode for the game is F2000EA8 E12FFF1E. The first part is all we need so we change it to 02000EA8. Search for this in the disassembled source. When you find it copy a good chunk of the code (I recommend copying from the last bx r14 -- 02000CC4 in our case -- to 02000EA8). Paste this into your hacking text file.

Now before we can find a good place to put our routine we need to know how many lines of code it will be. BTW here's the part where you will need to start learning some ARM assembly. What we need to do is manually load the address into one register (think of it as a variable if you are familiar with higher level programming) and load the value to store inside that address into another register. The trick is that with ARM assembly you can't load most addresses with one command because of how the opcodes are constructed. For example you can't do:

```
mov r0,#0x223bab4
```

but you can do:

```
mov r0,#0x2200000
add r0,r0,#0x3b000 ; r0 now contains 0x223b000
add r0,r0,#0xab4   ; r0 now contains 0x223bab4
```

You can actually cut out the last add and I will should you how in a second.

Since we don't know which registers will be safe to use for now we will use rX for the value to store and rY for the address:

```
mov rX,#0xff
mov rY,#0x2200000
add rY,rY,#0x3b000
strb rX,[rY,#+0xab4] ; stores the byte in rX to rY + 0xab4 (0x223bab4)
strb rX,[rY,#+0xab6]
strb rX,[rY,#+0xb84]
strb rX,[rY,#+0xb86]
```

So now we know that we will need enough space for 7 instructions to be overwritten. So let's take a look at the source code to see if there's anything that doesn't seem vital and may result in a crash if we overwrite it. You want to avoid any area of code that has an unconditional branch (b 02000000 or bx r0 for example), any area of code that loads a value into a register that will be used as an address to load from or store a value in or that loads a value to store into an address, etc.. This is sort of hard to explain so I hope you can understand that.

```
:02000CC8 E92D4010 stmdb  r13!,{r4,r14}
:02000CCC E59F1070 ldr r1,[r15, #+0x70]       ;r15+0x70=*(02000d44)=#36021816(0x0225a638)
:02000CD0 E3E00001 mvn  r0,#0x1
:02000CD4 E5911008 ldr r1,[r1, #+0x8]       ;r1+0x8=*(0225a640)=#1709507141(0x65e50245)
:02000CD8 E1510000 cmp r1,r0
:02000CDC 0A000003 beq 02000CF0
:02000CE0 E2800001 add  r0,r0,#0x1
:02000CE4 E1510000 cmp r1,r0
:02000CE8 0A000011 beq 02000D34
:02000CEC EA000005 b 02000D08
:02000CF0 E59F0050 ldr r0,[r15, #+0x50]       ;r15+0x50=*(02000d48)=#41942076(0x027ffc3c)
:02000CF4 E5900000 ldr r0,[r0, #+0x0]       ;r0+0x0=*(027ffc3c)=#-1542826689(0xa40a553f)
:02000CF8 E35000C0 cmp r0,#0xC0
:02000CFC 2A00000C bcs 02000D34
:02000D00 EB0265E2 bl 0209A490
:02000D04 EA00000A b 02000D34
:02000D08 EB0265E0 bl 0209A490
:02000D0C E59F4030 ldr r4,[r15, #+0x30]       ;r15+0x30=*(02000d44)=#36021816(0x0225a638)
:02000D10 E594100C ldr r1,[r4, #+0xc]       ;r4+0xc=*(0225a644)=#13580320(0x00cf3820)
:02000D14 E5940008 ldr r0,[r4, #+0x8]       ;r4+0x8=*(0225a640)=#1709507141(0x65e50245)
:02000D18 E1510000 cmp r1,r0
:02000D1C AA000004 bge 02000D34
:02000D20 EB0265DA bl 0209A490
:02000D24 E594100C ldr r1,[r4, #+0xc]       ;r4+0xc=*(0225a644)=#13580320(0x00cf3820)
:02000D28 E5940008 ldr r0,[r4, #+0x8]       ;r4+0x8=*(0225a640)=#1709507141(0x65e50245)
:02000D2C E1510000 cmp r1,r0
:02000D30 BAFFFFFA blt 02000D20
:02000D34 E59F0008 ldr r0,[r15, #+0x8]       ;r15+0x8=*(02000d44)=#36021816(0x0225a638)
:02000D38 E3A01000 mov  r1,#0x0       ;r1=0(0x0)
:02000D3C E580100C str r1,[r0, #+0xc]       ;r0+0xc=*(0225a644)=#13580320(0x00cf3820)
:02000D40 E8BD8010 ldmia  r13!,{r4,r15}
:02000D44 0225A638 eoreq  r10,r5,#0x3800000
```

```
:02000D48 027FFC3C rsbeqs r15,r15,#0x3C00
:02000D4C E92D4008 stmdb  r13!,{r3,r14}
:02000D50 EB0151BC bl 02055448
:02000D54 EB0157B2 bl 02056C24
:02000D58 EB023749 bl 0208EA84
:02000D5C E59F0008 ldr r0,[r15, #+0x8]        ;r15+0x8=*(02000d6c)=#67110208(0x04000540)
:02000D60 E3A01000 mov  r1,#0x0        ;r1=0(0x0)
:02000D64 E5801000 str r1,[r0, #+0x0]        ;r0+0x0=*(04000540)=#0(0x00000000)
:02000D68 E8BD8008 ldmia  r13!,{r3,r15}
:02000D6C 04000540 streq r0,[r0],#-0x540
:02000D70 E92D4008 stmdb  r13!,{r3,r14}
:02000D74 E59F20FC ldr r2,[r15, #+0xfc]        ;r15+0xfc=*(02000e78)=#34458048(0x020dc9c0)
:02000D78 E1D211DD ldrsb r1,[r2, #+0x1d]        ;r2+0x1dd=*(020dcb9d)=#-68988928(0xfbe35000)
:02000D7C E3510000 cmp r1,#0x0
:02000D80 DA000007 ble 02000DA4
:02000D84 E3A00000 mov  r0,#0x0        ;r0=0(0x0)
:02000D88 E1C201B8 strh r0,[r2, #+0x18]        ;r2+0x1b8=*(020dcb78)=#-443760120(0xe58cc208)
:02000D8C E1C201B6 strh r0,[r2, #+0x16]        ;r2+0x1b6=*(020dcb76)=#-1039604832(0xc208e3a0)
:02000D90 E1C201B4 strh r0,[r2, #+0x14]        ;r2+0x1b4=*(020dcb74)=#-476003583(0xe3a0c301)
:02000D94 E5C2001C strb r0,[r2, #+0x1c]        ;r2+0x1c=*(020dc9dc)=#-535752702(0xe0111002)
:02000D98 E2410001 sub  r0,r1,#0x1
:02000D9C E5C2001D strb r0,[r2, #+0x1d]        ;r2+0x1d=*(020dc9dd)=#14684432(0x00e01110)
:02000DA0 E8BD8008 ldmia  r13!,{r3,r15}
:02000DA4 E1D231B4 ldrh r3,[r2, #+0x14]        ;r2+0x1b4=*(020dcb74)=#-476003583(0xe3a0c301)
:02000DA8 E59F10CC ldr r1,[r15, #+0xcc]        ;r15+0xcc=*(02000e7c)=#3315(0x00000cf3)
:02000DAC E1E03003 mvn  r3,r3
:02000DB0 E0033000 and  r3,r3,r0
:02000DB4 E1C231B6 strh r3,[r2, #+0x16]        ;r2+0x1b6=*(020dcb76)=#-1039604832(0xc208e3a0)
:02000DB8 E1C201B4 strh r0,[r2, #+0x14]        ;r2+0x1b4=*(020dcb74)=#-476003583(0xe3a0c301)
:02000DBC E1D201B6 ldrh r0,[r2, #+0x16]        ;r2+0x1b6=*(020dcb76)=#-
1039604832(0xc208e3a0)
:02000DC0 E1100001 tst r0,r1
:02000DC4 0A000003 beq 02000DD8
:02000DC8 E1C201B8 strh r0,[r2, #+0x18]        ;r2+0x1b8=*(020dcb78)=#-443760120(0xe58cc208)
:02000DCC E1D201DA ldrsb r0,[r2, #+0x1a]        ;r2+0x1da=*(020dcb9a)=#123392(0x0001e200)
:02000DD0 E5C2001C strb r0,[r2, #+0x1c]        ;r2+0x1c=*(020dc9dc)=#-535752702(0xe0111002)
:02000DD4 EA000009 b 02000E00
:02000DD8 E1D231B4 ldrh r3,[r2, #+0x14]        ;r2+0x1b4=*(020dcb74)=#-476003583(0xe3a0c301)
:02000DDC E1130001 tst r3,r1
:02000DE0 0A000006 beq 02000E00
:02000DE4 E1D211DC ldrsb r1,[r2, #+0x1c]        ;r2+0x1dc=*(020dcb9c)=#-
481296383(0xe3500001)
:02000DE8 E2410001 sub  r0,r1,#0x1
:02000DEC E5C2001C strb r0,[r2, #+0x1c]        ;r2+0x1c=*(020dc9dc)=#-535752702(0xe0111002)
:02000DF0 E3510000 cmp r1,#0x0
:02000DF4 D1C231B8 strleh r3,[r2, #+0x18]        ;r2+0x1b8=*(020dcb78)=#-
443760120(0xe58cc208)
:02000DF8 D1D201DB ldrlesb r0,[r2, #+0x1b]
;r2+0x1db=*(020dcb9b)=#1342177762(0x500001e2)
:02000DFC D5C2001C strleb r0,[r2, #+0x1c]        ;r2+0x1c=*(020dc9dc)=#-535752702(0xe0111002)
:02000E00 E59F0070 ldr r0,[r15, #+0x70]        ;r15+0x70=*(02000e78)=#34458048(0x020dc9c0)
:02000E04 E3A02000 mov  r2,#0x0        ;r2=0(0x0)
:02000E08 E1C022B0 strh r2,[r0, #+0x20]        ;r0+0x2b0=*(020dcc70)=#-477846367(0xe384a4a1)
```

```
:02000E0C E1C021BE strh r2,[r0, #+0x1e]       ;r0+0x1be=*(020dcb7e)=#503309727(0x1dffe59f)
:02000E10 E1D012D4 ldrsb r1,[r0, #+0x24]      ;r0+0x2d4=*(020dcc94)=#-
1567764476(0xa28dd004)
:02000E14 E3510000 cmp r1,#0x0
:02000E18 DA000004 ble 02000E30
:02000E1C E2411001 sub  r1,r1,#0x1
:02000E20 E5C01024 strb r1,[r0, #+0x24]       ;r0+0x24=*(020dc9e4)=#-476040958(0xe3a03102)
:02000E24 E1D012D4 ldrsb r1,[r0, #+0x24]      ;r0+0x2d4=*(020dcc94)=#-
1567764476(0xa28dd004)
:02000E28 E3510000 cmp r1,#0x0
:02000E2C D1C022B2 strleh r2,[r0, #+0x22]     ;r0+0x2b2=*(020dcc72)=#-
1878989948(0x9000e384)
:02000E30 E59F1040 ldr r1,[r15, #+0x40]       ;r15+0x40=*(02000e78)=#34458048(0x020dc9c0)
:02000E34 E1D1E1B6 ldrh r14,[r1, #+0x16]      ;r1+0x1b6=*(020dcb76)=#-
1039604832(0xc208e3a0)
:02000E38 E31E00F0 tst r14,#0xF0
:02000E3C 08BD8008 ldmeqia  r13!,{r3,r15}
:02000E40 E1D1C2B2 ldrh r12,[r1, #+0x22]      ;r1+0x2b2=*(020dcc72)=#-
1878989948(0x9000e384)
:02000E44 E3A0200C mov  r2,#0xC       ;r2=12(0xc)
:02000E48 E20C00A0 and  r0,r12,#0xA0
:02000E4C E20C3050 and  r3,r12,#0x50
:02000E50 E1A000C0 mov  r0,r0,asr #0x1
:02000E54 E1800083 orr  r0,r0,r3,lsl #0x1
:02000E58 E00E300C and  r3,r14,r12
:02000E5C E1A00800 mov  r0,r0,lsl #0x10       ;r0=-910163968(0xc9c00000)
:02000E60 E1C131BE strh r3,[r1, #+0x1e]       ;r1+0x1be=*(020dcb7e)=#503309727(0x1dffe59f)
:02000E64 E00E0820 and  r0,r14,r0,lsr #0x10
:02000E68 E1C102B0 strh r0,[r1, #+0x20]       ;r1+0x2b0=*(020dcc70)=#-477846367(0xe384a4a1)
:02000E6C E1C1E2B2 strh r14,[r1, #+0x22]      ;r1+0x2b2=*(020dcc72)=#-
1878989948(0x9000e384)
:02000E70 E5C12024 strb r2,[r1, #+0x24]       ;r1+0x24=*(020dc9e4)=#-476040958(0xe3a03102)
:02000E74 E8BD8008 ldmia  r13!,{r3,r15}
:02000E78 020DC9C0 andeq  r12,r13,#0x300000
:02000E7C 00000CF3 unknown
:02000E80 E59F1024 ldr r1,[r15, #+0x24]       ;r15+0x24=*(02000eac)=#67109168(0x04000130)
:02000E84 E59F0024 ldr r0,[r15, #+0x24]       ;r15+0x24=*(02000eb0)=#41942952(0x027fffa8)
:02000E88 E1D120B0 ldrh r2,[r1, #+0x0]       ;r1+0xb0=*(040001e0)=#0(0x00000000)
:02000E8C E1D010B0 ldrh r1,[r0, #+0x0]        ;r0+0xb0=*(02800058)=#-1398275892(0xaca800cc)
:02000E90 E59F001C ldr r0,[r15, #+0x1c]       ;r15+0x1c=*(02000eb4)=#12287(0x00002fff)
:02000E94 E1821001 orr  r1,r2,r1
:02000E98 E0211000 eor  r1,r1,r0
:02000E9C E0010000 and  r0,r1,r0
:02000EA0 E1A00800 mov  r0,r0,lsl #0x10       ;r0=805240832(0x2fff0000)
:02000EA4 E1A00820 mov  r0,r0,lsr #0x10       ;r0=12287(0x2fff)
:02000EA8 E12FFF1E bx r14 (Jump to addr_9000E384?)
```

To show you what I mean take a look at 02000E00. It loads an address into r0 that will be used a few times right below it. If we were to overwrite the command where r0 loads the address but we left:

```
:02000E08 E1C022B0 strh r2,[r0, #+0x20]       ;r0+0x2b0=*(020dcc70)=#-477846367(0xe384a4a1)
:02000E0C E1C021BE strh r2,[r0, #+0x1e]       ;r0+0x1be=*(020dcb7e)=#503309727(0x1dffe59f)
```

chances are good that we would have a crash since r0 would be a "wild pointer" meaning that it could very well point to code in the game and it would corrupt itself when it stored the value. But if we leave 02000E00 alone and move down two instructions it looks like we might have a possible place to put our trainer:

```
:02000E00 E59F0070 ldr r0,[r15, #+0x70]       ;r15+0x70=*(02000e78)=#34458048(0x020dc9c0)
:02000E04 E3A02000 mov  r2,#0x0       ;r2=0(0x0)
:02000E08 E1C022B0 strh r2,[r0, #+0x20]       ;r0+0x2b0=*(020dcc70)=#-
477846367(0xe384a4a1)
:02000E0C E1C021BE strh r2,[r0, #+0x1e]       ;r0+0x1be=*(020dcb7e)=#503309727(0x1dffe59f)
:02000E10 E1D012D4 ldrsb r1,[r0, #+0x24]      ;r0+0x2d4=*(020dcc94)=#-
1567764476(0xa28dd004)
:02000E14 E3510000 cmp r1,#0x0
:02000E18 DA000004 ble 02000E30
:02000E1C E2411001 sub  r1,r1,#0x1
:02000E20 E5C01024 strb r1,[r0, #+0x24]       ;r0+0x24=*(020dc9e4)=#-476040958(0xe3a03102)
:02000E24 E1D012D4 ldrsb r1,[r0, #+0x24]      ;r0+0x2d4=*(020dcc94)=#-
1567764476(0xa28dd004)
:02000E28 E3510000 cmp r1,#0x0
:02000E2C D1C022B2 strleh r2,[r0, #+0x22]     ;r0+0x2b2=*(020dcc72)=#-
1878989948(0x9000e384)
:02000E30 E59F1040 ldr r1,[r15, #+0x40]       ;r15+0x40=*(02000e78)=#34458048(0x020dc9c0)
:02000E34 E1D1E1B6 ldrh r14,[r1, #+0x16]      ;r1+0x1b6=*(020dcb76)=#-
1039604832(0xc208e3a0)
```

It's possible that what is being stored from r2 @ 2000E08 and 2000E0C is vital to the button handler (which is what this area of the game is) but there's no hurt in seeing what happens. Also note that we can see right away that r1 can be used by our routine since the next command after our routine will be to load a new value into r1 so we know we won't harm anything by using it ourselves. Next we have to find another register that is free for our use. We search a little down at 2000E34 we see that r14 is getting loaded with a new value and since there are no branches in between the end of our routine and the address that r14 gets loaded with a new value we should be fine using it. So now we have our new addresses and we can finish our assembly code. First create a new text file in c:\devkitARM\bin and name it fma.asm. Paste the source code from above into the text file. Go to Edit -> Replace. For Find what put in rX and for Replace with put in r1. Click Replace All. Next follow the same steps to replace rY with r14. Now you should have what looks like:

```
mov r1,#0xFF
mov r14,#0x2200000
add r14,r14,#0x3b000
strb r1,[r14,#+0xab4]
strb r1,[r14,#+0xab6]
strb r1,[r14,#+0xb84]
strb r1,[r14,#+0xb86]
```

Once again leave a blank line at the bottom or the assembler will complain about there not being one. Save the .asm file then drag and drop it onto assemble.bat. Open up XVI32 and drag and drop the .cut file onto it. Use the same steps as above in the easy method to copy the code we need. Don't forget to put the copied code into your hacking text file either.

Now we open up another instance of XVI32 and drag and drop our .nds onto it. Now let's find out where to place our trainer inside the .nds file. We will try the same thing as before by subtracting 0x1FFC000 from 02000E08. We jump to 0x4E08 in XVI32 and see that the bytes (in reverse order) match what it shows in the disassembly listing so it looks like we are set. Overwrite the string (not insert) like we did before. If you did it right the next byte will be D4, if you did it wrong it will be B0. Now go to Save As and save your creation.

Start it up on your DS and let's see how we did. It looks like the trainer is running stable and we have the infinite health we wanted.

Now all that's left is to create an ips patch like we did before and distribute it to all your friends on the online communities.

I hope this will help those of you who have been wanting to create trainers and I would love to see a surge of them in the scene. If you have any questions or comments you can email me at comfortably_numb_@hotmail.com or message me on gbatemp.net or ezflash.sosuke.net with the user name cracker.