Basic hacking tutorial vol.1

By Con

Contents

I. Introduction

This is a short hacking tutorial to get started for absolute beginners. It is a learning by doing guide and I hope you are able to understand the basics of assembly and snes rom hacking afterwards.

II. Basics

II.1 The rom

Download http://www.bszelda.zeldalegends.net/zips/thirdquest.zip (Third Quest). This is a **hirom** and this guide refers to this rom format. If you want to hack AST, remember that this rom is *lorom*.

II.2 Programms

- Geiger http://geigercount.net/crypt/snes9x1.43.ep9r8.7z Disassembler (you won't get far without this tool). (Download 7zip at http://www.7-zip.org/)
- a hex editor (I use http://www.ultraedit.com/ Ultraedit)
- Zsnes 1.14 (w) http://www.bszelda.zeldalegends.net/zips/zsneswv1.14r0.80.zip Needed to make savestates for Geiger
- Lunar IPS http://www.zophar.net/utilities/download/lips100.zip (this programs creates patches and applies ips to a rom)

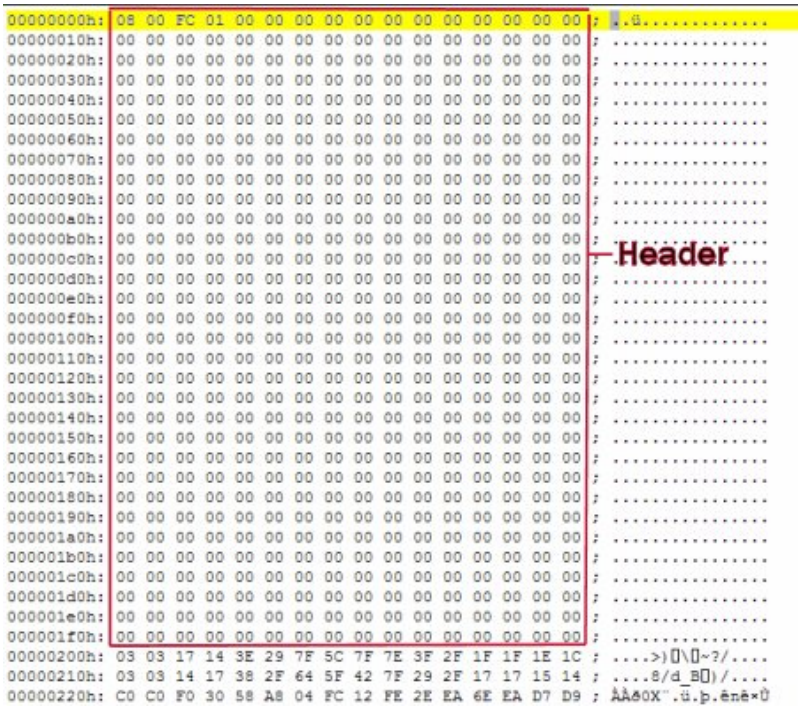
Not necessarily needed programs
- Lunar expand http://www.smwcentral.net/download.php?id=37&type=tools to expand roms e.g. if you want to insert screens.
- Lunar address http://www.smwcentral.net/download.php?id=43&type=tools if you want to hack loroms download this program to convert addresses.

II.3 Getting started

1. Make a **security copy** of your rom.
2. Change the rom's name to e.g. patch1.smc
3. Open patch1.smc with zsnes
4. wait until the title screen appears, than hit "F2" ->save. Exit zsnes.
5. Open the rom with Geiger
6. Click "yes" to remove the header. **Run** the game (button on debug console).
7. Click "F1" to load your zsnes savestate. You can register your name now.
8. Press "Shift + F1" to create a Geiger savestate
9. Exit Geiger.

Information: Why **zsnes**
BS Zelda cannot be loaded by snes9x 1.43. The problem is first fixed in later versions. Geiger's snes9x debugger is built upon snes9x 1.43 and thus can't load bszelda. Means a trick is needed. And since snes9x can load zsnes savestates… problem solved ☺
If there occur graphical problems with the title screen, don't mind! Just press enter.

Information: the **header** thing
Headers are the first $200 bytes and may contain some informations. The header is not necessary for emulation but somehow every rom has it. If you want to hack, **always remove the header!!!!**
Otherwise you can get in mathematic problems, because the header bytes are not calculated by addressing.



III. Basic romhacking

III.1 Hexadecimalsystem

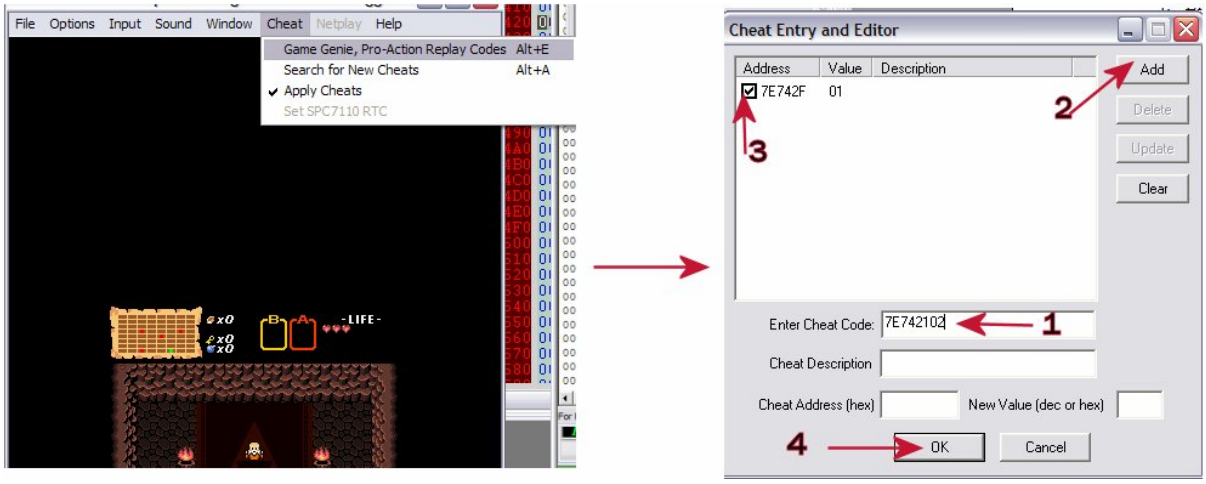
1 byte consists of two numbers and reach from 00 till FF (look at the upper pic).
Hexadecimal counts from 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

III.2 Ram addressing

Remember the PAR codes? E.g. if you want to cheat for the letter, you use the code **7E742F-01:** The address in the ram is 7E742F and the value there is „1“. Another example is the magic boomerang:
7E7421-02: The address in the ram is 7E7421 and the value there is “2”. If it was a “1” there, you’d have the wooden one.

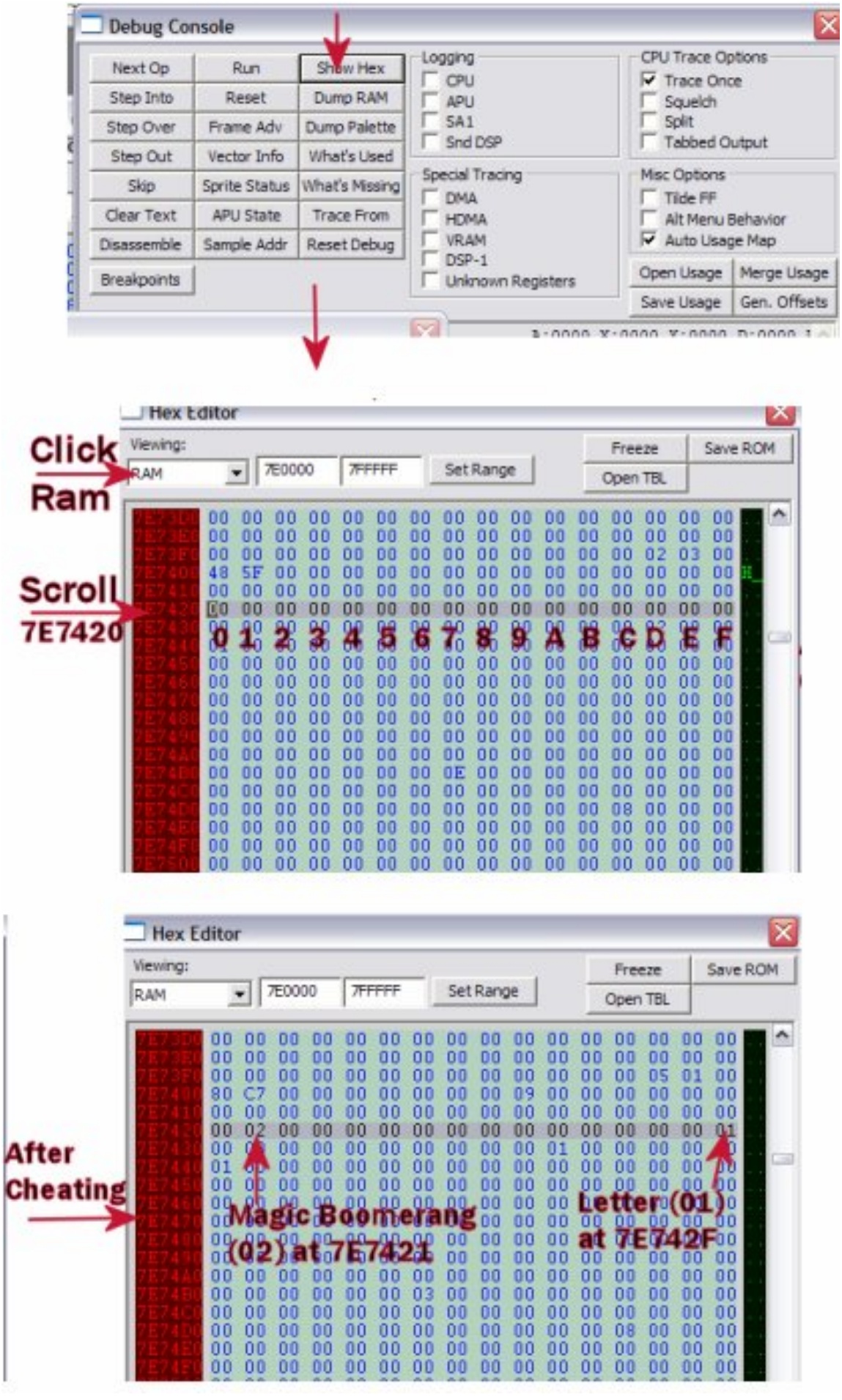
Check this out in Geiger.

1. Open patch1.smc with Geiger
2. hit “F1” hit load your savegame
3. Press “Cheat” -> “Game Genie/Pro Action Reply Cheats”
4. Enter 7e742f01 and 7e742102



Now

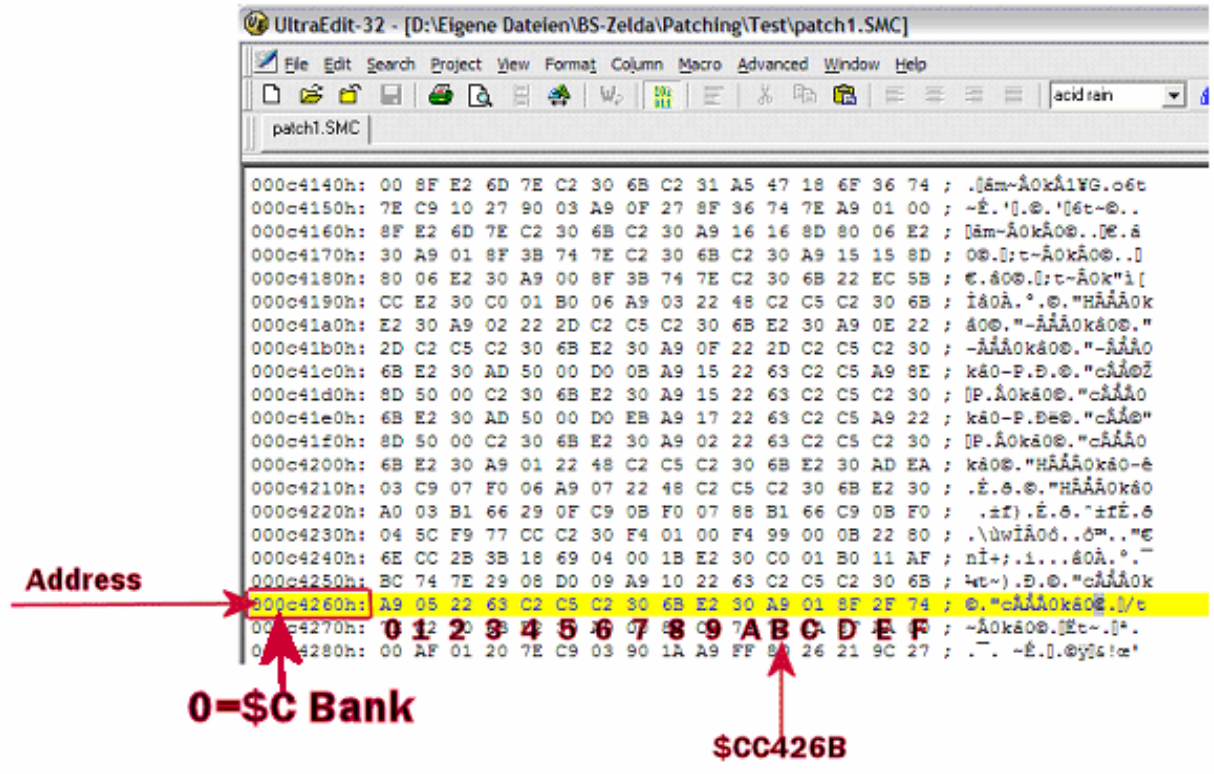
1. Click on the debug console **Show Hex**
2. Viewing RAM
3. Scroll to 7E7420
4. Check at place 1 for the magic boomerang “value 02” and place F for the letter “value 01”


Debug Console
Next Op
Run
Show Hex
Step Into
Reset
Dump RAM
Step Over
Frame Adv
Dump Palette
Step Out
Vector Info
What's Used
Skip
Sprite Status
What's Missing
Clear Text
APU State
Trace From
Disassemble
Sample Addr
Reset Debug
Breakpoints
Logging
CPU
APU
SA1
Snd DSP
Special Tracing
DMA
HDMA
VRAM
DSP-1
Unknown Registers
CPU Trace Options
Trace Once
Squelch
Split
Tabbed Output
Misc Options
Tilde FF
Alt Menu Behavior
Auto Usage Map
Open Usage
Merge Usage
Save Usage
Gen. Offsets
Click Ram
Scroll 7E7420
Hex Editor
Viewing:
RAM
7E0000
7FFFFF
Set Range
Freeze
Save ROM
Open TBL
0 1 2 3 4 5 6 7 8 9 A B C D E F
After Cheating
Magic Boomerang (02) at 7E7421
Letter (01) at 7E742F

III.3 Rom Addressing

Be sure that your rom has no header!
Now open your rom with the hex editor.

You will see something like this:

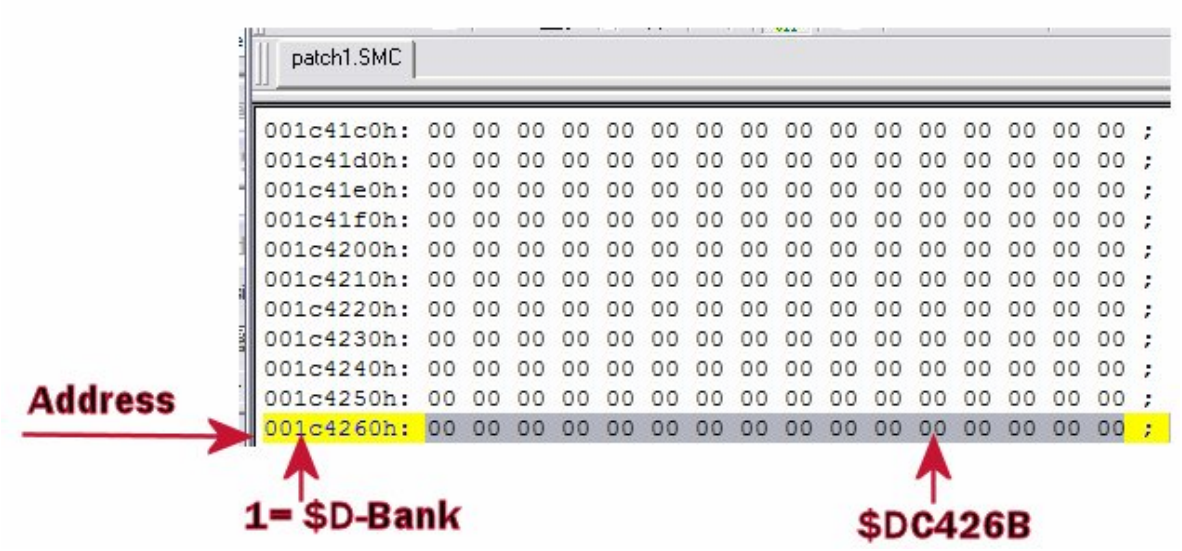


In this pic we have the value **A9** at the address **$CC426B**!
Left at "Address" you see the number 000C4260. That's the "pc-address".
It's easy to get from the pc address to the snes address:

- The byte A9 is at 000C426B (just count as seen in the pic)
- Than just put another C instead of the 3rd "0"

That's it!

Information: **Banks**
But why a "C" – Now that's because we are in the $C – Bank of the rom. Every MB has it's letter. A 1 MB rom has only a $C Bank, a 2 MB rom a $C and a $D bank. Since Third Quest is a 2 MB rom, we also have a $D-Bank. The following pic shows the byte at **$DC426B**!



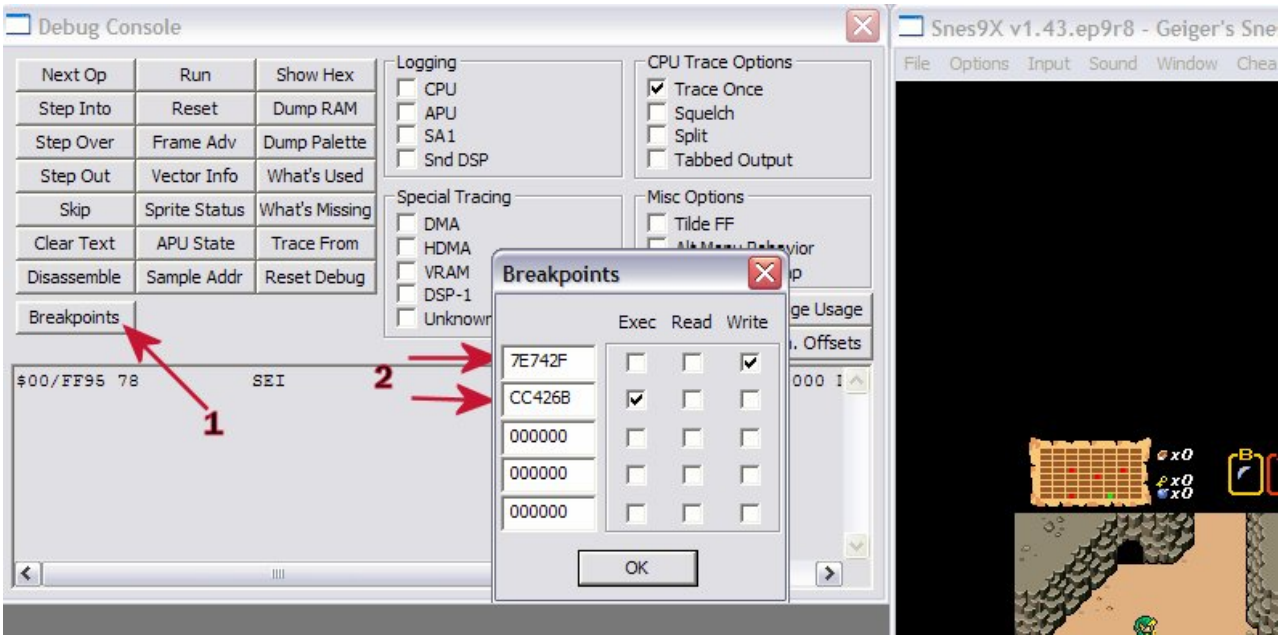
Ok… now you know how the ram & rom addressing works!

IV. Geiger and Assembly basics

Now, why is the addressing that important and how are ram and rom connected? In point III, I gave two important addresses for your first patch:
7E742F-01: ram address for the letter
$CC426B: this rom address that is telling to save the letter into the ram

How is this working?

1. Open Geiger and load the rom, run (debug console) and load your savestate “F1” (be sure cheats are off)
2. Start the game
3. Press “Breakpoints” at the debug console

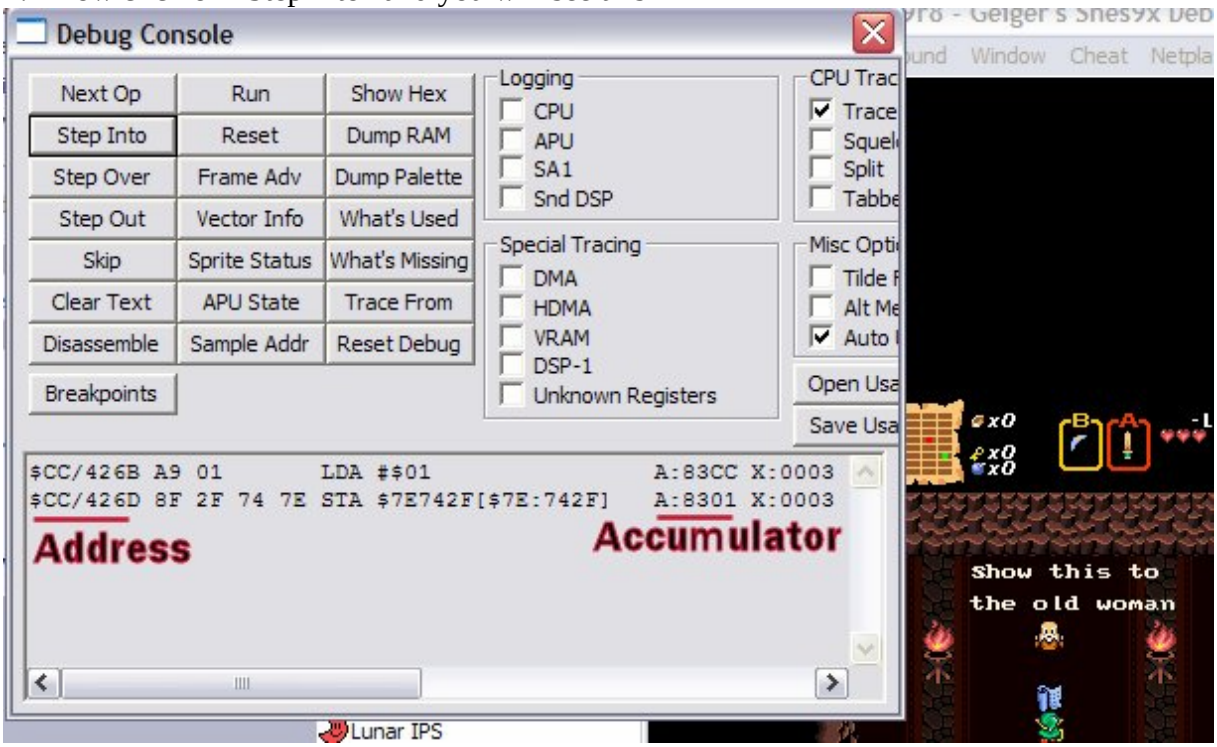


Information: **Breakpoints**
Breakpoints are an indispensable tool for you. There are 2 options

- insert a ram address and click the checkboxes “read” or “write”. Write is needed if a value is stored into ram, like in our case a 01 will be stored at 7E742F. So we click write. (If this value is read (e.g. when you show the letter to the old woman, you must click “read”))
- If you know the address already like I know it is $CC/426B and you want a break at this point you must insert the rom address and click execute!

Let’s check it out.

1. Insert the rom address cc426b and click “execute”
2. Go into the cave and collect the letter
3. Geiger will now happily stop as soon you try to collect the letter
4. Now click on “Step into” and you will see this



Fine, but what is this here telling me?

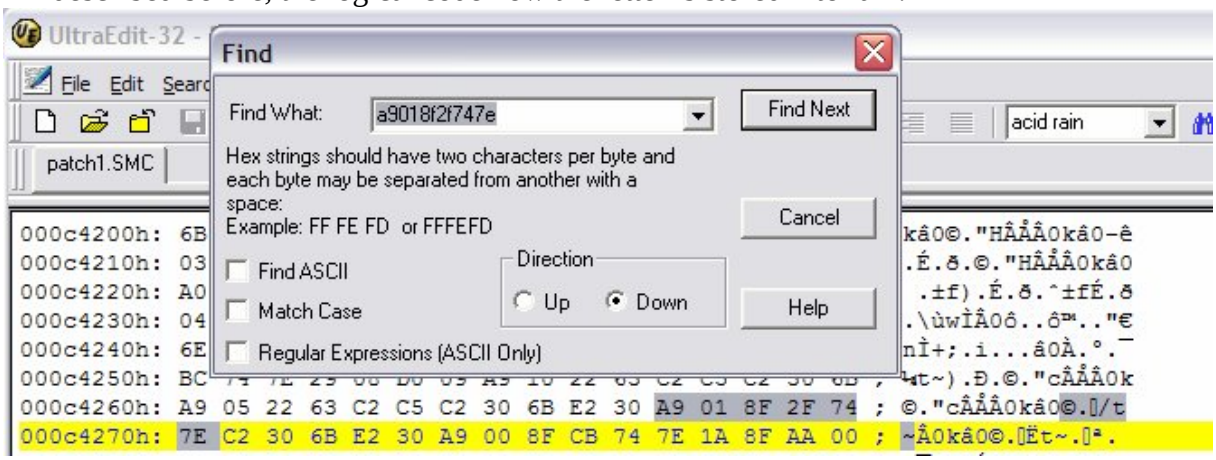
To answer these questions, I must give you some basics in assembly. Let’s start with **opcodes** (operating codes)!
Here you will have two opcodes, the first is A9. A9 means LDA (check the pic) and loads a value into the **Accumulator**. In this case the value is “01”.
After A9 01 (load value 01 into the Accumulator) you see that A:83*CC* changes to A:83*01*!
The second opcode is 8F. 8F means STA and stores the value from the accumulator into the ram. Means the value 01 will be stored to 7E742F. This is exactly the same as if you use the PAR Cheat 7E742F-01!

IMPORTANT NOTE!!!
For some reasons in SNES programming, the bytes are **reversed**. The ram address 7E 74 2F will be in the rom code 2F 74 7E. The same is valid when you use rom addresses.

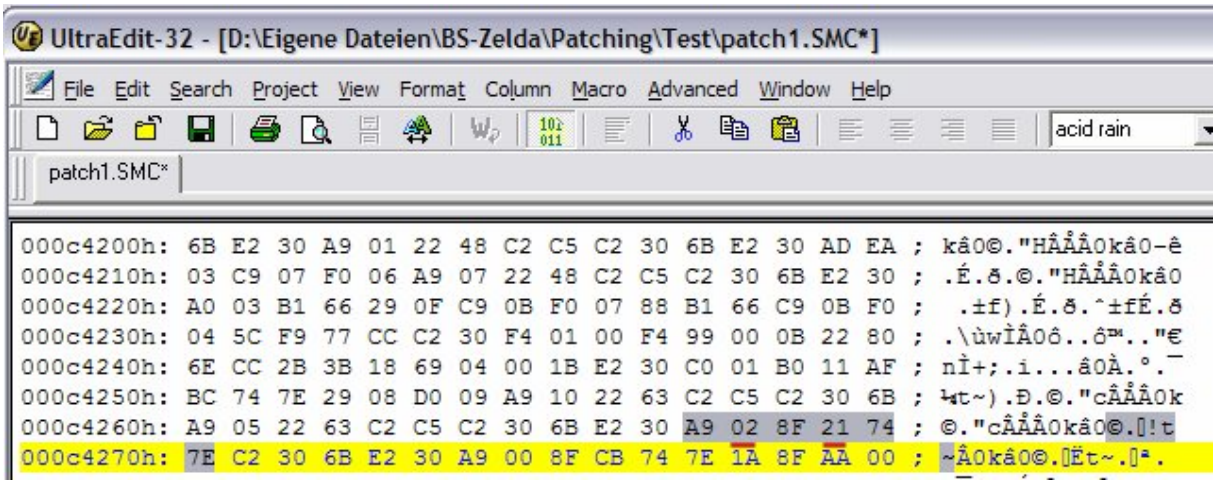
V. Your first patch

You know now the basics how rom and ram are connected. Now we want to hack!
Let's make a stupid patch. You collect the letter, but want the magic boomerang instead

1. Open the rom with the hex editor and search for a9 01 8f 2f 74 7e – that is, as described before, the logical code how the letter is stored into ram.

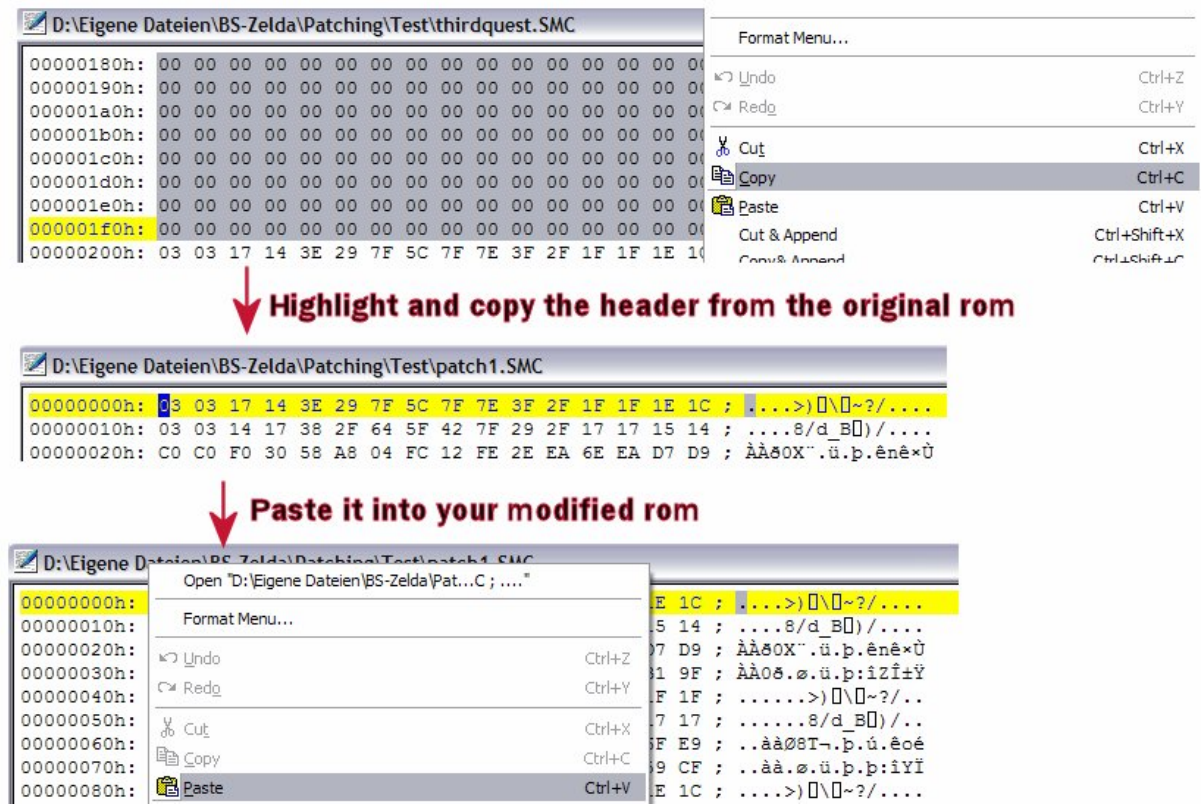


2. Now change it to the magic boomerang PAR Code, which is
A9 02 (load 02 into accumulator)
8F 21 74 7E (save to ram address 7E7421)

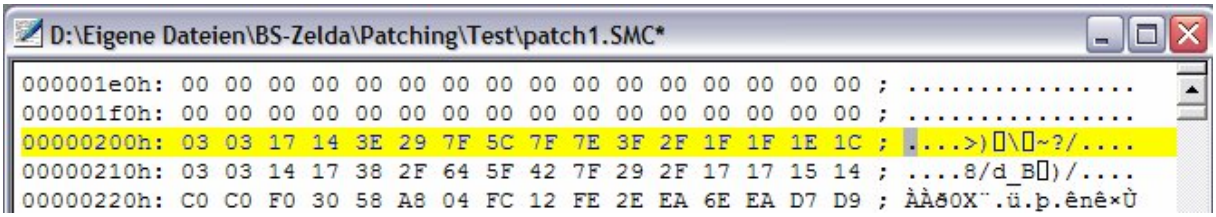


3. Save and play the rom. Collect the letter and you'll receive the magic boomerang instead of the letter

4. Create an ips file of your patch. To do this open the original rom, copy the header and paste it into the beginning of patch1.smc.



Including the header again is rather important. Otherwise, your patch will be corrupted! It will look like this afterwards:



5. *Save* the rom and Run Lunar ips. Click “create ips file”!
Open “thirdquest.smc” as original, unmodified rom.
Now select as new modified rom “patch1.smc” and click on save.
Lunar ips now created an ips patch file that works on all bszelda roms.

Congratulations, you’ve done your first hacking work!!!!!!!